



Panduan Developer

Amazon Rekognition



Amazon Rekognition: Panduan Developer

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan antara para pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan properti dari masing-masing pemilik, yang mungkin berafiliasi, terkait dengan, atau disponsori oleh Amazon, atau tidak.

Table of Contents

Apa itu Amazon Rekognition?	1
Kemampuan kunci	1
Kasus penggunaan	2
Manfaat	4
Kelayakan Amazon Rekognition dan HIPAA	4
Apakah Anda baru pertama kali menggunakan Amazon Rekognition?	5
Cara kerjanya	6
Tipe analisis	7
Label	9
Label kustom	10
Deteksi Keaktifan Wajah	11
Deteksi dan analisis wajah	11
Pencarian wajah	11
Jalur orang	12
Peralatan Pelindung Diri Pribadi	12
Selebriti	12
Pendeteksi teks	12
Konten yang tidak pantas atau menyinggung	13
Kustomisasi	13
Analisis massal	14
Operasi citra dan video	14
Operasi Amazon Rekognition Image	14
Operasi Amazon Rekognition Video	14
Operasi berbasis penyimpanan dan bukan penyimpanan	15
Menggunakan AWS SDK atau HTTP untuk memanggil operasi Amazon Rekognition API	15
Operasi API penyimpanan dan bukan penyimpanan	16
Operasi bukan penyimpanan	16
Operasi API berbasis penyimpanan	18
Versioning model	20
Memulai	22
Langkah 1: Siapkan akun AWS dan buat Pengguna	22
Buat AWS Akun dan Pengguna	23
Langkah 2: Siapkan AWS CLI dan AWS SDK	25
Memberikan akses programatis	27

Bekerja dengan AWS SDK	31
Langkah 3: Memulai menggunakan API AWS CLI dan AWS SDK	32
Memformat contoh AWS CLI	32
Langkah selanjutnya	33
Langkah 4: Mulai menggunakan konsol	33
Siapkan izin konsol	34
Latihan 1: Deteksi objek dan adegan (konsol)	37
Latihan 2: Analisis wajah (konsol)	44
Latihan 3: Bandingkan wajah (konsol)	47
Latihan 4: Lihat metrik gabungan (konsol)	50
Bekerja dengan gambar dan video	52
Bekerja dengan citra	52
Spesifikasi citra	53
Menganalisis citra dalam bucket Amazon S3	55
Menggunakan sistem file lokal	71
Menampilkan kotak pembatas	86
Mendapatkan orientasi citra dan koordinat kotak pembatas	99
Bekerja dengan analisis video tersimpan	110
Tipe analisis	111
Gambaran umum Amazon Rekognition Video API	111
Memanggil operasi Amazon Rekognition Video	114
Mengonfigurasi Amazon Rekognition Video	120
Menganalisis video yang tersimpan (SDK)	124
Menganalisis video (AWS CLI)	153
Referensi: Notifikasi hasil analisis video	157
Pemecahan masalah Amazon Rekognition Video	159
Bekerja dengan acara video streaming	161
Ikhtisar operasi prosesor aliran Video Rekognition Amazon	162
Menandai pemroses aliran Amazon Rekognition Video	163
Penanganan kesalahan	165
Komponen kesalahan	165
Pesan dan kode kesalahan	166
Penanganan kesalahan dalam aplikasi Anda	171
Menggunakan Amazon Rekognition dengan FedRAMP	172
Praktik terbaik untuk sensor, citra input, dan video	177
Latensi operasi Amazon Rekognition Image	177

Rekomendasi untuk citra input perbandingan wajah	177
Rekomendasi umum untuk gambar masukan untuk operasi wajah	178
Rekomendasi untuk mencari wajah dalam koleksi	178
Rekomendasi untuk pengaturan kamera (citra dan video)	179
Rekomendasi untuk pengaturan kamera (video tersimpan dan streaming)	182
Rekomendasi untuk pengaturan kamera (video streaming)	182
Rekomendasi untuk Penggunaan Face Liveness	183
Mendeteksi objek dan konsep	184
Objek Respons Label	185
Kotak Pembatas	185
Skor Keyakinan	186
Induk	186
Kategori	187
Alias	187
Properti Gambar	187
Versi Model	189
Filter Inklusi dan Pengecualian	189
Menyortir dan Mengagregasikan Hasil	190
Mendeteksi label dalam citra	190
DetectLabels permintaan operasi	201
DetectLabels respon	202
Mengubah respon DetectLabels	206
Mendeteksi label dalam video	210
StartLabelPermintaan Deteksi	211
GetLabelDetection Respon Operasi	212
Mengubah Respon GetLabelDetection	219
Mendeteksi label dalam streaming acara video	227
Mempersiapkan Amazon Rekognition Video Amazon dan sumber daya Amazon Kinesis	228
Operasi deteksi label untuk streaming acara video	233
Mendeteksi label kustom	240
Mendeteksi dan menganalisis wajah	241
Gambaran umum deteksi wajah dan perbandingan wajah	242
Pedoman atribut wajah	244
Mendeteksi wajah dalam citra	245
DetectFaces permintaan operasi	256
DetectFaces respon operasi	257

Membandingkan wajah dalam citra	264
CompareFaces permintaan operasi	277
CompareFaces respon operasi	277
Mendeteksi wajah dalam video yang tersimpan	280
GetFaceDetection respon operasi	289
Mencari wajah dalam koleksi	295
Mengelola koleksi	298
Mengelola wajah dalam koleksi	299
Mengelola pengguna dalam koleksi	299
Menggunakan ambang kesamaan untuk mengaitkan wajah	300
Panduan untuk menggunakan IndexFaces	300
Aplikasi keamanan kritis atau publik	300
Aplikasi berbagi foto dan media sosial	300
Penggunaan umum	300
Mencari wajah dan pengguna dalam koleksi	301
Menggunakan ambang batas kemiripan untuk mencocokkan wajah	302
Kasus penggunaan yang melibatkan keselamatan publik	302
Menggunakan Amazon Rekognition untuk membantu keselamatan publik	304
Membuat koleksi	305
CreateCollection permintaan operasi	312
CreateCollection respon operasi	312
Penandaan koleksi	312
Tambahkan tanda ke koleksi baru	312
Tambahkan tanda ke koleksi yang sudah ada	314
Daftar tanda dalam koleksi	315
Hapus tanda dari koleksi	316
Mencantumkan koleksi	317
ListCollections permintaan operasi	323
ListCollections respon operasi	324
Menjelaskan koleksi	324
DescribeCollection permintaan operasi	331
DescribeCollectionrespon operasi	332
Menghapus koleksi	332
DeleteCollection permintaan operasi	339
DeleteCollection respon operasi	339
Menambahkan wajah ke koleksi	339

Memfilter wajah	340
IndexFaces permintaan operasi	350
IndexFaces respon operasi	350
Daftar wajah dan pengguna terkait dalam koleksi	359
ListFaces permintaan operasi	365
ListFaces respon operasi	365
Menghapus wajah dari koleksi	367
DeleteFaces permintaan operasi	372
DeleteFaces respon operasi	372
Membuat pengguna	373
Menghapus pengguna	376
Mengaitkan wajah dengan pengguna	378
AssociateFaces respon operasi	381
Memutus wajah dari pengguna	382
DisassociateFaces respon operasi	385
Daftar pengguna dalam koleksi	386
ListUsers respon operasi	389
Mencari wajah (ID wajah)	390
SearchFaces permintaan operasi	396
SearchFaces respon operasi	396
Mencari wajah (citra)	398
SearchFacesByImage permintaan operasi	405
SearchFacesByImage respon operasi	406
Mencari pengguna (ID wajah/ID pengguna)	407
SearchUsers permintaan operasi	411
SearchUsers respon operasi	411
Mencari pengguna (gambar)	413
SearchUsersByImage permintaan operasi	416
SearchUsersByImage respon operasi	417
Mencari video yang disimpan untuk wajah	418
GetFaceSearch respon operasi	427
Mencari wajah dalam koleksi dalam streaming video	432
Mempersiapkan Amazon Rekognition Video Amazon dan sumber daya Amazon Kinesis	433
Mencari wajah dalam video streaming	436
Streaming menggunakan plugin GStreamer	460
Mengatasi masalah video streaming	463

Lintasan orang	470
GetPersonTracking respon operasi	479
Mendeteksi alat pelindung diri	484
Tipe APD	485
Penutup wajah	485
Penutup tangan	485
Penutup kepala	485
kepercayaan deteksi APD	486
Meringkas APD yang terdeteksi dalam sebuah citra	486
Tutorial: Membuat AWS Lambda fungsi yang mendeteksi gambar dengan APD	487
Memahami API deteksi APD	487
Menyuplai citra	487
Memahami DetectProtectiveEquipment tanggapannya	489
Mendeteksi APD dalam citra	494
Contoh: kotak pembatas dan penutup wajah	506
Mengenal selebriti	522
Pengenalan selebriti dibandingkan dengan pencarian wajah	522
Mengenal selebriti dalam sebuah citra	523
Memanggil RecognizeCelebrities	524
RecognizeCelebrities permintaan operasi	533
RecognizeCelebrities respon operasi	534
Mengenal selebriti dalam video yang tersimpan	537
GetCelebrityRecognition respon operasi	552
Mendapatkan informasi selebriti	554
Memanggil GetCelebrityInfo	555
GetCelebrityInfo permintaan operasi	560
GetCelebrityInfo respon operasi	560
Memoderasi konten	561
Menggunakan API moderasi citra dan video	563
Kategori Label	563
Jenis konten	575
Kepercayaan	576
Penentuan Versi	576
Menyortir dan Mengagregasi	576
Status adaptor Moderasi Kustom	577
Menguji Moderasi Konten versi 7 dan mengubah respons API	577

AWS SDK dan Panduan Penggunaan untuk Moderasi Konten versi 7	578
Pemetaan label untuk Versi 6.1 hingga 7	578
Mendeteksi citra yang tidak pantas	584
Mendeteksi konten yang tidak pantas dalam gambar	584
.....	584
DetectModerationLabels permintaan operasi	591
DetectModerationLabels respon operasi	592
Mendeteksi video tersimpan yang tidak pantas	593
GetContentModeration respon operasi	601
Meningkatkan akurasi dengan Custom Moderation	604
Membuat dan menggunakan adaptor	605
Mempersiapkan kumpulan data Anda	608
Mengelola adaptor dengan AWS CLI dan SDK	610
Tutorial adaptor Moderasi Kustom	617
Mengevaluasi dan meningkatkan adaptor Anda	635
Format file manifes	637
Praktik terbaik untuk adaptor pelatihan	642
Menyiapkan AutoUpdate izin	643
AWS Notifikasi Dasbor Kesehatan untuk Rekognition	645
Meninjau konten yang tidak sesuai dengan Amazon A2I	647
Mendeteksi teks	653
Mendeteksi teks dalam sebuah citra	655
DetectText permintaan operasi	663
DetectText respon operasi	664
Mendeteksi teks dalam video yang tersimpan	670
Filter	680
GetTextDetection respon	680
Mendeteksi segmen video	686
Isyarat teknis	687
Frame hitam	687
Kredit	687
Diagram warna	688
Papan tulis	688
Logo studio	688
Daftar isi	688
Deteksi sorotan	689

Tentang API deteksi Segmen Video Rekognition Amazon	690
Menggunakan API Segmen Amazon Rekognition	690
Memulai analisis segmen	691
Mendapatkan hasil analisis segmen	692
Contoh: Mendeteksi segmen dalam video yang tersimpan	697
Mendeteksi keaktifan wajah	710
Persyaratan Keaktifan Wajah Sisi Pengguna	712
Arsitektur dan Diagram Urutan	713
Prasyarat	715
Langkah 1: Siapkan AWS akun	715
Langkah 2: Siapkan SDK Face Liveness AWS	715
Langkah 3: Siapkan AWS Amplify Resources	716
Praktik Terbaik untuk Mendeteksi Keaktifan Wajah	716
Memprogram Amazon Rekognition Face Liveness API	716
Langkah 1: CreateFaceLivenessSession	717
Langkah 2: StartFaceLivenessSession	718
Langkah 3: GetFaceLivenessSessionResults	718
Langkah 4: Menanggapi hasil	719
Memanggil Face Liveness API	719
Mengkonfigurasi dan Menyesuaikan Aplikasi Anda	726
Mengkonfigurasi Aplikasi Anda	726
Sesuaikan Aplikasi Anda	726
Model Tanggung Jawab Bersama Face Liveness	726
Panduan pembaruan Face Liveness	731
Pembuatan versi dan kerangka waktu	731
Rilis versi dan matriks kompatibilitas	732
Komunikasi rilis baru	733
FAQ Face Liveness	733
Analisis massal	737
Memproses gambar dalam jumlah besar	737
Untuk membuat pekerjaan analisis massal (CLI)	737
StartMediaAnalysisJob manifestasi keluaran	739
Jenis konten	740
Verifikasi prediksi dan pelatihan adaptor	740
Tutorial	741
Menyimpan Data Amazon Rekognition dengan Amazon RDS dan DynamoDB	741

Prasyarat	742
Mendapatkan Label untuk Gambar di Amazon S3 Bucket	742
Membuat Tabel Amazon DynamoDB	744
Mengunggah Data ke DynamoDB	745
Membuat Database MySQL di Amazon RDS	747
Mengunggah Data ke Tabel MySQL Amazon RDS	748
Menggunakan Amazon Rekognition dan Lambda untuk menandai aset di bucket Amazon S3 ..	750
Prasyarat	752
Konfigurasi peran Lambda IAM	752
Buat proyek	753
Tulis kode	756
Kemas proyek	766
Deploy fungsi Lambda	767
Uji metode Lambda	768
Membuat AWS aplikasi penganalisis video	769
Prasyarat	770
Prosedur	770
Membuat fungsi Amazon Rekognition Lambda	771
Prasyarat	773
Buat topik SNS	773
Buat fungsi Lambda	773
Konfigurasi fungsi Lambda	774
Konfigurasi peran IAM Lambda	775
Buat proyek AWS Toolkit for Eclipse Lambda	776
Uji fungsi Lambda	779
Menggunakan Amazon Rekognition untuk Verifikasi Identitas	781
Prasyarat	782
Membuat Koleksi	782
Registrasi Pengguna Baru	783
Login Pengguna yang Ada	792
Mendeteksi Label dalam Gambar Menggunakan Lambda dan Python	794
Buat fungsi Lambda (konsol)	795
(Opsional) Buat layer (konsol)	796
Tambahkan kode Python (konsol)	798
Untuk menambahkan kode Python (konsol)	800
Contoh kode	804

Tindakan	808
CompareFaces	809
CreateCollection	819
DeleteCollection	825
DeleteFaces	831
DescribeCollection	837
DetectFaces	843
DetectLabels	859
DetectModerationLabels	881
DetectText	889
DisassociateFaces	899
GetCelebrityInfo	901
IndexFaces	903
ListCollections	916
ListFaces	922
RecognizeCelebrities	931
SearchFaces	945
SearchFacesByImage	954
Skenario	964
Bangun koleksi dan temukan wajah di dalamnya	964
Mendeteksi dan menampilkan elemen dalam gambar	977
Mendeteksi informasi dalam video	993
Contoh lintas layanan	1033
Membuat aplikasi nirserver untuk mengelola foto	1033
Mendeteksi APD dalam gambar	1037
Mendeteksi wajah dalam gambar	1039
Mendeteksi objek dalam gambar	1040
Mendeteksi orang dan objek dalam video	1043
Menyimpan EXIF dan informasi gambar lainnya	1045
Referensi API	1047
Keamanan	1048
Identity and access management	1048
Audiens	1049
Mengautentikasi dengan identitas	1049
Mengelola akses menggunakan kebijakan	1052
Cara kerja Amazon Rekognition dengan IAM	1055

Kebijakan terkelola AWS	1060
Contoh kebijakan berbasis identitas	1068
Contoh kebijakan berbasis sumber daya	1072
Pemecahan Masalah	1073
Perlindungan data	1075
Enkripsi data	1076
Privasi lalu lintas jaringan internet	1079
Menggunakan Amazon Rekognition dengan Amazon VPC endpoint	1079
Membuat titik akhir Amazon VPC untuk Amazon Rekognition	1080
Buat Kebijakan VPC endpoint untuk Amazon Rekognition	1080
Validasi kepatuhan	1082
Ketahanan	1082
Konfigurasi dan analisis kelemahan	1083
Pencegahan wakil bingung lintas layanan	1083
Keamanan infrastruktur	1085
Memantau	1087
Memantau Rekognition dengan Amazon CloudWatch	1087
Menggunakan CloudWatch metrik untuk Rekognition	1088
Akses metrik Rekognition	1089
Buat alarm	1090
CloudWatchmetrik untuk Rekognition	1092
Mencatat panggilan API Amazon Rekognition dengan AWS CloudTrail	1096
Informasi Amazon Rekognition diCloudTrail	1097
Memahami entri berkas log Amazon Rekognition	1098
Pedoman dan kuota	1101
Wilayah yang didukung	1101
Kuota Set	1101
Citra Amazon Rekognition	1101
Analisis Massal Gambar Rekognition Amazon	1101
Video yang disimpan Amazon Rekognition Video	1102
Video streaming Amazon Rekognition Video	1103
Kuota default	1103
Hitung perubahan kuota TPS	1104
Praktik terbaik untuk kuota TPS	1104
Buat kasus untuk mengubah kuota TPS	1104
Riwayat dokumen	1107

AWSGlosarium	1121
.....	mcxxii

Apa itu Amazon Rekognition?

Amazon Rekognition adalah layanan analisis gambar dan video berbasis cloud yang memudahkan untuk menambahkan kemampuan visi komputer canggih ke aplikasi Anda. Layanan ini didukung oleh teknologi pembelajaran mendalam yang telah terbukti dan tidak memerlukan keahlian pembelajaran mesin untuk digunakan. Amazon Rekognition menyertakan API sederhana easy-to-use yang dapat dengan cepat menganalisis file gambar atau video apa pun yang disimpan di Amazon S3.

Anda dapat menambahkan fitur yang mendeteksi objek, teks, konten tidak aman, menganalisis gambar/video, dan membandingkan wajah dengan aplikasi Anda menggunakan API Rekognition. Dengan API pengenalan wajah Amazon Rekognition, Anda dapat mendeteksi, menganalisis, dan membandingkan wajah untuk berbagai kasus penggunaan, termasuk verifikasi pengguna, katalog, penghitungan orang, dan keselamatan publik.

Layanan ini didasarkan pada teknologi pembelajaran mendalam yang terbukti, sangat skalabel, yang dikembangkan oleh ilmuwan visi komputer Amazon, teknologi yang dapat menganalisis miliaran gambar dan video setiap hari. Rekognition secara rutin belajar dari data baru, dan kami sering menambahkan label dan fitur baru ke layanan.

Untuk informasi selengkapnya, lihat [FAQ Amazon Rekognition](#).

Kemampuan kunci

Analisis Gambar:

- Objek, Pemandangan, dan Deteksi Konsep - Mendeteksi dan mengklasifikasikan objek, adegan, konsep, dan selebriti dalam gambar.
- Deteksi Teks - Mendeteksi dan mengenali teks cetak dan tulisan tangan dalam gambar dalam berbagai bahasa.
- Konten Tidak Aman - Mendeteksi dan memfilter konten dan gambar yang eksplisit, tidak pantas, dan penuh kekerasan. Mendeteksi label konten granular yang tidak aman.
- Pengakuan selebriti - Kenali puluhan ribu selebriti dalam gambar Anda di berbagai kategori, seperti politisi, atlet, aktor, dan musisi.
- Analisis Wajah - Mendeteksi, menganalisis, dan membandingkan wajah, bersama dengan atribut wajah, seperti jenis kelamin, usia, dan emosi. Kasus penggunaan dapat mencakup verifikasi pengguna, katalog, penghitungan orang, dan keselamatan publik.

- Label Kustom - Bangun pengklasifikasi khusus untuk mendeteksi objek khusus untuk kasus penggunaan Anda, seperti logo, produk, karakter.
- Properti Gambar - Analisis properti gambar seperti kualitas, warna, ketajaman, kontras.

Analisis Video:

- Objek, Pemandangan, dan Deteksi Konsep - Mendeteksi dan mengklasifikasikan objek, adegan, konsep, dan selebriti dalam video.
- Deteksi Teks - Mendeteksi dan mengenali teks cetak dan tulisan tangan dalam video dalam berbagai bahasa.
- People pathing - Lacak orang yang diidentifikasi saat mereka bergerak melintasi bingkai video.
- Analisis Wajah - Mendeteksi, menganalisis, dan membandingkan wajah dalam streaming atau video yang disimpan.
- Pengakuan selebriti - Kenali puluhan ribu selebriti dalam video yang Anda simpan di berbagai kategori, seperti politisi, atlet, aktor, dan musisi.
- Deteksi Konten Tidak Aman - Mendeteksi konten eksplisit, tidak pantas, dan kekerasan dalam video.
- Segmentasi video - Secara otomatis mengidentifikasi segmen video yang berguna, seperti bingkai hitam dan kredit akhir.
- Face liveness - Deteksi jika pengguna langsung hadir selama verifikasi wajah.

Kasus penggunaan

Perpustakaan Media yang Dapat Dicari - Rekognition mendeteksi label, objek, konsep, dan adegan dalam gambar dan video. Anda dapat membuat label ini dapat dicari berdasarkan analisis konten visual ini. Berguna untuk membangun perpustakaan gambar dan video yang dapat dicari.

Verifikasi Identitas Pengguna Berbasis Wajah - Konfirmasikan identitas pengguna dengan membandingkan wajah dalam gambar untuk referensi gambar wajah. Berguna untuk verifikasi identitas dalam aplikasi.

Face Liveness Detection - Rekognition Face Liveness adalah fitur machine learning (ML) yang dikelola sepenuhnya yang dirancang untuk membantu pengembang mencegah penipuan selama verifikasi identitas berbasis wajah. Fitur ini membantu Anda memverifikasi bahwa pengguna secara fisik hadir di depan kamera dan bukan aktor yang buruk menipu wajah pengguna. Menggunakan

Rekognition Face Liveness dapat membantu Anda mendeteksi serangan spoof yang disajikan ke kamera, seperti foto cetak, foto/video digital, atau topeng 3D. Ini juga membantu mendeteksi serangan spoof yang melewati kamera, seperti video pra-rekaman atau deepfake yang disuntikkan langsung ke subsistem pengambilan video.

Pencarian Wajah - Dengan Rekognition, Anda dapat mencari gambar, video yang disimpan, dan streaming video untuk wajah yang cocok dengan yang disimpan dalam wadah yang dikenal sebagai koleksi wajah. Koleksi wajah adalah indeks wajah yang Anda miliki dan Anda kelola. Mencari orang berdasarkan wajah mereka mengharuskan Anda Mengindeks wajah dan kemudian Cari wajah.

Deteksi Konten Tidak Aman - Mendeteksi dan memfilter konten eksplisit, tidak pantas, dan kekerasan dalam gambar dan video. Menggunakan label untuk penyaringan granular berdasarkan kebutuhan bisnis. Content Moderation API juga menampilkan daftar hierarkis dari setiap label yang terdeteksi (objek dan konsep), bersama dengan skor kepercayaan. Objek/label ini menunjukkan kategori spesifik dari konten tidak aman, yang memungkinkan pemfilteran granular dan pengelolaan volume besar konten buatan pengguna (UGC). Anda dapat menyesuaikan output API Moderasi Konten dengan adaptor, yang meningkatkan kinerja untuk gambar seperti yang Anda berikan sebagai data pelatihan.

Deteksi Alat Pelindung Diri - Mendeteksi alat pelindung diri dalam gambar untuk memantau kepatuhan keselamatan di berbagai industri. Anda dapat secara otomatis menandai kondisi yang tidak aman dengan mendeteksi peralatan yang tidak tepat dan menerima peringatan tentang kondisi ini, yang dapat meningkatkan kepatuhan dan pelatihan.

Pengakuan Selebriti - Kenali selebriti dalam gambar dan video Anda di seluruh kategori, seperti politisi, atlet, aktor, dan musisi. Anda dapat mengidentifikasi penampilan selebriti tanpa perlu memberikan nama.

Deteksi Teks - Mendeteksi dan mengekstrak teks dalam gambar untuk pencarian visual atau mengekstrak metadata. Ini bekerja pada font dan gaya yang berbeda. Mendeteksi orientasi untuk menangani teks pada tanda dan spanduk.

Label Kustom - Identifikasi objek, konsep, dan adegan khusus untuk kasus penggunaan bisnis, seperti deteksi logo. Anda dapat melatih pengklasifikasi khusus untuk menangani objek khusus atau berpemilik, yang meningkatkan akurasi pada objek utama versus pengklasifikasi umum. Untuk informasi selengkapnya, lihat [Apa itu Custom Label Amazon Rekognition?](#) di Panduan Developer Custom Label Amazon Rekognition.

Manfaat

Mengintegrasikan analisis gambar dan video yang kuat ke dalam aplikasi Anda - Tambahkan analisis gambar dan video yang akurat ke aplikasi tanpa keahlian. Amazon Rekognition API memungkinkan analisis melalui pembelajaran mendalam tanpa memerlukan pengetahuan pembelajaran mesin apa pun. Anda dapat dengan cepat membangun visi komputer ke dalam aplikasi web, seluler, dan perangkat.

Analisis gambar dan video berbasis pembelajaran mendalam - Menganalisis gambar dan video menggunakan pembelajaran mendalam untuk akurasi tinggi. Amazon Rekognition dapat mendeteksi label, objek, adegan, wajah, selebriti. Filter hasil untuk menyertakan/mengecualikan label tertentu.

Analisis gambar yang dapat diskalakan - Menganalisis jutaan gambar untuk mengatur kumpulan data visual besar-besaran. Timbangan untuk menangani pustaka gambar dan lalu lintas yang berkembang. Anda tidak perlu merencanakan kapasitas, dan Anda hanya membayar untuk apa yang Anda gunakan.

Menganalisis dan memfilter gambar berdasarkan properti - Menganalisis dan memfilter gambar berdasarkan properti, seperti kualitas, warna, dan konten visual, dan mendeteksi ketajaman gambar, kecerahan, kontras.

Integrasi dengan AWS layanan lain - Amazon Rekognition terintegrasi di luar kotak dengan S3 dan Lambda. Anda dapat memanggil API Amazon Rekognition dari Lambda dan memproses gambar di Amazon S3 tanpa memindahkan data. Rekognition memiliki skalabilitas dan keamanan bawaan menggunakan IAM. AWS

Biaya rendah - ay-as-you-go harga P, tidak ada minimum atau komitmen. Tingkat gratis tersedia untuk memulai. Hemat lebih banyak saat skala penggunaan melalui harga berjenjang. Hemat biaya relatif terhadap solusi in-house.

Kustomisasi sederhana - Sesuaikan akurasi untuk kasus penggunaan Anda dengan adaptor. Berikan contoh gambar untuk melatih adaptor. Meningkatkan deteksi objek dan label untuk domain tertentu. Cara mudah untuk menyesuaikan analisis tanpa keahlian ML.

Untuk informasi selengkapnya, lihat [FAQ Amazon Rekognition](#).

Kelayakan Amazon Rekognition dan HIPAA

Ini adalah Layanan yang Memenuhi Syarat HIPAA. [Untuk informasi lebih lanjut tentang AWS, Undang-Undang Portabilitas dan Akuntabilitas Asuransi Kesehatan AS tahun 1996 \(HIPAA\), dan](#)

[menggunakan AWS layanan untuk memproses, menyimpan, dan mengirimkan informasi kesehatan yang dilindungi \(PHI\), lihat Ikhtisar HIPAA.](#)

Apakah Anda baru pertama kali menggunakan Amazon Rekognition?

Jika Anda baru pertama kali menggunakan Amazon Rekognition, kami menyarankan agar Anda membaca bagian-bagian berikut secara berurutan:

1. [Bagaimana Amazon Rekognition bekerja](#)— Bagian ini memperkenalkan berbagai komponen Amazon Rekognition yang Anda gunakan untuk menciptakan pengalaman. end-to-end
2. [Memulai dengan Amazon Rekognition](#)— Di bagian ini, Anda mengatur akun, menginstal SDK yang mencerminkan bahasa pilihan Anda, dan menguji API Rekognition Amazon. Untuk daftar bahasa pemrograman yang didukung oleh Amazon Rekognition, lihat. [Menggunakan Rekognition dengan SDK AWS](#)
3. [Bekerja dengan citra](#) — Bagian ini menyediakan informasi tentang menggunakan Amazon Rekognition dengan citra yang disimpan dalam bucket Amazon S3 dan citra yang diambil dari sistem file lokal.
4. [Bekerja dengan analisis video tersimpan](#) — Bagian ini menyediakan informasi tentang menggunakan Amazon Rekognition dengan video yang disimpan dalam bucket Amazon S3.
5. [Bekerja dengan acara video streaming](#) — Bagian ini menyediakan informasi tentang menggunakan Amazon Rekognition dengan video streaming.

Bagaimana Amazon Rekognition bekerja

Amazon Rekognition menyediakan dua set API untuk analisis visual:

- Gambar Rekognition Amazon untuk analisis gambar
- Amazon Rekognition Video untuk analisis video

Analisis gambar

Dengan Amazon Rekognition Image, aplikasi Anda dapat:

- Mendeteksi objek, adegan, dan konsep dalam gambar
- Kenali selebriti
- Mendeteksi teks dalam berbagai bahasa
- Mendeteksi konten atau gambar yang eksplisit, tidak pantas, atau penuh kekerasan
- Mendeteksi, menganalisis, dan membandingkan wajah dan atribut wajah seperti usia dan emosi
- Mendeteksi keberadaan APD

Kasus penggunaan termasuk menyempurnakan aplikasi foto, membuat katalog gambar, dan memoderasi konten.

Analisis video

Dengan Amazon Rekognition Video, aplikasi Anda dapat:

- Lacak orang dan objek di seluruh bingkai video
- Kenali objek
- Kenali selebriti
- Cari video yang disimpan dan streaming untuk orang yang diminati
- Analisis wajah untuk atribut seperti usia dan emosi
- Mendeteksi konten atau gambar yang eksplisit, tidak pantas, atau penuh kekerasan
- Agregat dan urutkan hasil analisis berdasarkan stempel waktu dan segmen
- Mendeteksi orang, hewan peliharaan, dan paket dalam streaming video

Kasus penggunaan termasuk analisis video, membuat katalog video, dan memfilter konten yang tidak pantas.

Fitur utama

- Analisis pembelajaran mendalam yang kuat
- Deteksi akurasi tinggi untuk objek, adegan, wajah, teks
- API yang mudah digunakan untuk mengintegrasikan ke dalam aplikasi
- Model yang dapat disesuaikan disetel ke data Anda
- Analisis pustaka media yang dapat diskalakan

Amazon Rekognition memungkinkan Anda meningkatkan akurasi model pembelajaran mendalam tertentu dengan melatih adaptor khusus. Misalnya, dengan Moderasi Kustom Amazon Rekognition, Anda dapat mengadaptasi model analisis gambar dasar Amazon Rekognition dengan melatih adaptor khusus dengan gambar Anda. Lihat [Meningkatkan akurasi dengan Moderasi Kustom](#) untuk informasi selengkapnya.

Bagian berikut mencakup jenis analisis yang disediakan Amazon Rekognition dan ikhtisar operasi Gambar Rekognition Amazon dan Amazon Rekognition Video. Juga mencakup perbedaan antara operasi penyimpanan dan bukan penyimpanan.

Untuk mendemonstrasikan API Amazon Rekognition, Anda dapat [melihat Langkah 3: Memulai menggunakan AWS CLI dan AWS SDK API](#), yang mencakup mencoba Rekognition di konsol. AWS

Topik

- [Tipe analisis](#)
- [Operasi citra dan video](#)
- [Operasi API penyimpanan dan bukan penyimpanan](#)
- [Versioning model](#)

Tipe analisis

Berikut ini merupakan tipe analisis yang dapat dilakukan oleh API Amazon Rekognition Image serta API Amazon Rekognition Video. Untuk informasi tentang API, lihat [Operasi citra dan video](#).

Tabel berikut mencantumkan operasi yang perlu Anda gunakan sehubungan dengan jenis media yang Anda gunakan dan kasus penggunaan Anda:

Kasus Penggunaan	Tipe Media	Operasi
Memoderasi konten	Citra	DetectModerationLabels , StartMediaAnalysisJob , GetMediaAnalysisJob , ListMediaAnalysisJobs
	Video yang disimpan	StartContentModeration , GetContentModeration
Verifikasi identitas	Gambar	CreateCollection , CreateUser , IndexFaces , AssociateFaces , SearchFacesByImage , SearchUsersByImage
	Video yang disimpan	CreateCollection , IndexFaces , StartFaceSearch , GetFaceSearch
	Streaming video (Mendeteksi keaktifan wajah)	CreateFaceLivenessSession , StartFaceLivenessSession , GetFaceLivenessSessionResults ,
Analisis wajah	Citra	DetectFaces , CompareFaces
	Video yang disimpan	StartFaceDetection , GetFaceDetection
	Streaming video	CreateStreamProcessor , StartStreamProcessor
Pengenalan objek dan aktivitas	Citra	DetectLabels

Kasus Penggunaan	Tipe Media	Operasi
	Video yang disimpan	StartLabelDetection , GetLabelDetection
Rumah Terhubung	Streaming Video	StartStreamProcessor
Analisis Media	Video yang disimpan	StartSegmentDetection , GetSegmentDetection
Keamanan di tempat kerja	Citra	DetectProtectiveEquipment
Deteksi teks	Citra	DetectText
	Video	StartTextDetection , GetTextDetection
Jalur orang	Video	StartPersonTracking , GetPersonTracking
Pengakuan selebriti	Citra	RecognizeCelebrities
	Video	StartCelebrityRecognition , GetCelebrityRecognition
Deteksi label khusus	Citra	DetectCustomLabels
	Pelatihan model	Lihat panduan pengembang Custom Labels

Label

Label mengacu pada salah satu dari berikut ini: objek (misalnya, bunga, pohon, atau meja), acara (misalnya, pernikahan, kelulusan, atau pesta ulang tahun), konsep (misalnya, lanskap, malam, dan alam) atau kegiatan (misalnya, berlari atau bermain bola basket). Amazon Rekognition dapat mendeteksi label dalam citra dan video. Untuk informasi selengkapnya, lihat [Mendeteksi objek dan konsep](#).

Rekognition dapat mendeteksi daftar besar label dalam gambar dan video yang disimpan. Rekognition juga dapat mendeteksi sejumlah kecil label dalam streaming video.

Gunakan operasi berikut untuk mendeteksi label berdasarkan kasus penggunaan Anda:

- Untuk mendeteksi label dalam gambar: Gunakan [DetectLabels](#). Anda dapat mengidentifikasi properti gambar seperti warna gambar dominan dan kualitas gambar. Untuk mencapai ini, gunakan [DetectLabels](#) dengan IMAGE_PROPERTIES sebagai parameter input.
- Untuk mendeteksi label dalam video yang disimpan: Gunakan [StartLabelDetection](#). Deteksi warna gambar dominan dan kualitas gambar tidak didukung untuk video yang disimpan.
- Untuk mendeteksi label dalam streaming video: Gunakan [CreateStreamProcessor](#). Deteksi warna gambar dominan dan kualitas gambar tidak didukung untuk streaming video.

Anda dapat menentukan jenis label yang ingin Anda kembalikan untuk deteksi label gambar dan video tersimpan dengan menggunakan opsi pemfilteran inklusif dan eksklusif.

Label kustom

Label Kustom Amazon Rekognition dapat mengidentifikasi objek dan adegan dalam citra yang spesifik untuk kebutuhan bisnis Anda dengan melatih sebuah model machine learning. Misalnya, Anda dapat melatih model untuk mendeteksi logo atau bagian-bagian mesin rekayasa pada jalur perakitan.

Note

Untuk informasi tentang Label Kustom Amazon Rekognition, lihat [Panduan Developer Label Kustom Amazon Rekognition](#).

Amazon Rekognition menyediakan sebuah konsol yang dapat Anda gunakan untuk membuat, melatih, mengevaluasi, serta menjalankan model machine learning. Untuk informasi selengkapnya, lihat [Memulai Label Kustom Amazon Rekognition](#) di Panduan Developer Label Kustom Amazon Rekognition. Anda juga dapat menggunakan API Label Kustom Amazon Rekognition untuk melatih dan menjalankan sebuah model. Untuk informasi selengkapnya, lihat [Memulai SDK Label Kustom Rekognition Amazon](#) di Panduan Pengembang Rekognition Amazon. CustomLabels

Untuk menganalisis gambar menggunakan model terlatih, gunakan [DetectCustomLabels](#).

Deteksi Keaktifan Wajah

Amazon Rekognition Face Liveness dapat membantu Anda memverifikasi bahwa pengguna yang melalui verifikasi identitas berbasis wajah secara fisik hadir di depan kamera dan bukan aktor buruk yang menipu wajah pengguna. Ini mendeteksi serangan spoof yang disajikan ke kamera dan serangan yang melewati kamera. Seorang pengguna dapat menyelesaikan pemeriksaan Face Liveness dengan mengambil selfie video pendek, dan skor Liveness dikembalikan untuk cek. Face Liveness ditentukan dengan perhitungan probabilistik dan skor kepercayaan (antara 0-100) dikembalikan setelah cek. Semakin tinggi skor, semakin besar kepercayaan diri bahwa orang yang mengambil cek itu hidup.

Untuk informasi lebih lanjut tentang Face Liveness, lihat [Mendeteksi keaktifan wajah](#).

Deteksi dan analisis wajah

Amazon Rekognition dapat mendeteksi wajah dalam citra dan video yang tersimpan. Dengan Amazon Rekognition, Anda bisa mendapatkan informasi tentang:

- Di mana wajah terdeteksi dalam gambar atau video
- Landmark wajah seperti posisi mata
- Adanya oklusi wajah dalam gambar
- Emosi yang terdeteksi, seperti bahagia atau sedih
- Arah tatapan mata dari tatapan seseorang dalam gambar

Anda juga dapat menafsirkan dan informasi demografis seperti jenis kelamin atau usia. Anda dapat membandingkan wajah dalam gambar dengan wajah yang terdeteksi di gambar lain. Informasi tentang wajah juga dapat disimpan untuk pengambilan informasi nantinya. Untuk informasi selengkapnya, lihat [Mendeteksi dan menganalisis wajah](#).

Untuk mendeteksi wajah dalam citra, gunakan [DetectFaces](#). Untuk mendeteksi wajah dalam video yang tersimpan, gunakan [StartFaceDetection](#).

Pencarian wajah

Amazon Rekognition dapat mencari wajah. Informasi wajah diindeks ke dalam kontainer yang disebut sebagai koleksi. Informasi wajah dalam koleksi kemudian dapat dicocokkan dengan wajah yang terdeteksi dalam citra, video yang tersimpan, dan video streaming. Untuk informasi selengkapnya, [Mencari wajah dalam koleksi](#).

Untuk mencari wajah yang dikenal dalam citra, gunakan [DetectFaces](#). Untuk mencari wajah yang dikenal di video yang tersimpan, gunakan [StartFaceDetection](#). Untuk mencari wajah yang dikenal di video streaming, gunakan [CreateStreamProcessor](#).

Jalur orang

Amazon Rekognition dapat melacak jalur orang yang terdeteksi dalam video yang tersimpan. Amazon Rekognition Video menyediakan pelacakan jalur, detail pada wajah, dan informasi pada lokasi dalam bingkai untuk orang yang terdeteksi dalam video. Untuk informasi selengkapnya, lihat [Lintasan orang](#).

Untuk mendeteksi orang dalam video yang tersimpan, gunakan [StartPersonTracking](#).

Peralatan Pelindung Diri Pribadi

Amazon Rekognition dapat mendeteksi Alat Pelindung Diri (APD) yang dikenakan oleh orang-orang yang terdeteksi dalam citra. Amazon Rekognition mendeteksi penutup wajah, penutup tangan, dan penutup kepala. Amazon Rekognition memprediksi apakah sebuah item APD menutupi bagian tubuh yang tepat. Anda juga bisa mendapatkan kotak pembatas untuk orang yang terdeteksi dan item APD. Untuk informasi selengkapnya, lihat [Mendeteksi alat pelindung diri](#).

Untuk mendeteksi APD dalam citra, gunakan [DetectProtectiveEquipment](#).

Selebriti

Amazon Rekognition dapat mengenali ribuan selebriti dalam citra dan video yang disimpan. Anda bisa mendapatkan informasi tentang letak wajah selebriti berada pada citra, penanda wajah, dan ekspresi dari wajah selebriti. Anda bisa mendapatkan informasi pelacakan untuk selebriti saat selebriti tersebut muncul di seluruh video yang tersimpan. Anda juga bisa mendapatkan informasi lebih lanjut tentang selebriti yang diakui, seperti emosi yang diungkapkan, dan presentasi gender. Untuk informasi selengkapnya, lihat [Mengenali selebriti](#).

Untuk mengenali selebriti dalam citra, gunakan [RecognizeCelebrities](#). Untuk mengenali selebriti dalam video yang disimpan, gunakan [StartCelebrityRecognition](#).

Pendeteksi teks

Amazon Rekognition Text dalam Citra dapat mendeteksi teks dalam citra dan mengubahnya menjadi teks yang dapat terbaca oleh mesin. Untuk informasi selengkapnya, lihat [Mendeteksi teks](#).

Untuk mendeteksi teks dalam citra, gunakan [DetectText](#).

Konten yang tidak pantas atau menyinggung

Amazon Rekognition dapat menganalisis citra dan video yang disimpan untuk konten dewasa dan kekerasan. Untuk informasi selengkapnya, lihat [Memoderasi konten](#).

Untuk mendeteksi citra yang tidak aman, gunakan [DetectModerationLabels](#). Untuk mendeteksi video tersimpan yang tidak aman, gunakan [StartContentModeration](#).

Kustomisasi

API analisis gambar tertentu yang ditawarkan oleh Rekognition memungkinkan Anda meningkatkan akurasi model pembelajaran mendalam dengan membuat adaptor khusus yang dilatih pada data Anda sendiri. Adaptor adalah komponen yang plug-in ke model pembelajaran mendalam Rekognition yang telah dilatih sebelumnya, meningkatkan akurasi dengan pengetahuan domain berdasarkan gambar Anda. Anda melatih adaptor untuk memenuhi kebutuhan Anda dengan menyediakan dan membuat anotasi gambar sampel.

Setelah Anda membuat adaptor, Anda diberikan file AdapterId. Anda dapat memberikan ini AdapterId ke operasi untuk menentukan bahwa Anda ingin menggunakan adaptor yang telah Anda buat. Misalnya, Anda memberikan [DetectModerationLabels](#) API AdapterId untuk analisis gambar sinkron. Menyediakan AdapterId sebagai bagian dari permintaan dan Rekognition akan secara otomatis menggunakannya untuk meningkatkan prediksi untuk gambar Anda. Ini memungkinkan Anda untuk memanfaatkan kemampuan Rekognition sambil menyesuaikannya agar sesuai dengan kebutuhan Anda.

Anda juga memiliki opsi untuk mendapatkan prediksi gambar secara massal dengan [StartMediaAnalysisJob](#) API. Lihat [Analisis massal](#) untuk informasi lebih lanjut.

Anda dapat menilai keakuratan operasi Rekognition dengan mengunggah gambar ke konsol Rekognition dan menjalankan analisis pada gambar-gambar ini. Rekognition akan membuat anotasi gambar Anda menggunakan fitur yang dipilih, dan Anda kemudian dapat meninjau prediksi, menggunakan prediksi terverifikasi untuk menentukan label mana yang akan mendapat manfaat dari membuat adaptor.

Saat ini Anda dapat menggunakan adaptor dengan [DetectModerationLabels](#). Untuk informasi selengkapnya tentang membuat dan menggunakan adaptor, lihat [Meningkatkan akurasi dengan Custom Moderation](#).

Analisis massal

Rekognition Bulk Analysis memungkinkan Anda memproses koleksi besar gambar secara asinkron dengan menggunakan file manifes bersama dengan operasi. [StartMediaAnalysisJob](#) Lihat [Analisis massal](#) untuk informasi lebih lanjut.

Operasi citra dan video

Amazon Rekognition menawarkan dua set API utama untuk analisis gambar dan video:

- Amazon Rekognition Image: API ini dirancang untuk menganalisis gambar.
- Amazon Rekognition Video: API ini berfokus pada analisis video yang disimpan dan streaming.

Kedua API dapat mendeteksi berbagai entitas seperti wajah dan objek. Untuk pemahaman komprehensif tentang jenis perbandingan dan deteksi yang didukung, lihat bagian di [Tipe analisis](#).

Operasi Amazon Rekognition Image

Operasi Gambar Rekognition Amazon sinkron. Input dan respons dalam format JSON. Operasi Amazon Rekognition Image menganalisis citra input yang berada dalam format citra .jpg atau .png. Citra diteruskan ke operasi Amazon Rekognition Image yang dapat disimpan dalam bucket Amazon S3. Jika Anda tidak menggunakan AWS CLI, Anda juga dapat meneruskan byte gambar yang dikodekan Base64 langsung ke operasi Rekognition Amazon. Untuk informasi selengkapnya, lihat [Bekerja dengan gambar](#).

Operasi Amazon Rekognition Video

Amazon Rekognition Video API memfasilitasi analisis video baik yang disimpan dalam bucket Amazon S3 atau dialirkan melalui Amazon Kinesis Video Streams.

Untuk operasi video yang disimpan, perhatikan hal berikut:

- Operasi tidak sinkron.
- Analisis harus dimulai dengan operasi “Mulai” (misalnya, [StartFaceDetection](#) untuk deteksi wajah dalam video yang disimpan).
- Status penyelesaian analisis dipublikasikan ke topik Amazon SNS.
- Untuk mengambil hasil analisis, gunakan operasi “Dapatkan” yang sesuai (misalnya, [GetFaceDetection](#)).

- Untuk informasi selengkapnya, lihat [Bekerja dengan analisis video tersimpan](#).

Untuk analisis video streaming:

- Kemampuan termasuk pencarian wajah dalam koleksi Video Rekognition dan deteksi label (objek atau konsep).
- Hasil analisis untuk label dikirim sebagai notifikasi Amazon SNS dan Amazon S3.
- Hasil pencarian wajah adalah output ke aliran data Kinesis.
- Manajemen analisis video streaming dilakukan melalui prosesor aliran Amazon Rekognition Video (misalnya, membuat prosesor menggunakan). [CreateStreamProcessor](#)
- Untuk informasi selengkapnya, lihat [Bekerja dengan acara video streaming](#).

Setiap operasi analisis video mengembalikan metadata tentang video yang sedang dianalisis, serta ID pekerjaan dan tag pekerjaan. Operasi seperti Deteksi Label dan Moderasi Konten untuk video memungkinkan pengurutan berdasarkan stempel waktu atau nama label, dan menggabungkan hasil berdasarkan stempel waktu atau segmen.

Operasi berbasis penyimpanan dan bukan penyimpanan

Operasi Amazon Rekognition dikelompokkan ke dalam kategori berikut.

- Operasi API bukan penyimpanan — Dalam operasi ini, Amazon Rekognition tidak mendesakkan informasi apa pun. Anda memberikan input citra dan video, operasi melakukan analisis, dan mengembalikan hasilnya, tetapi Amazon Rekognition tidak menyimpan apa pun. Untuk informasi selengkapnya, lihat [Operasi bukan penyimpanan](#).
- Operasi API berbasis penyimpanan - server Amazon Rekognition dapat menyimpan informasi wajah yang terdeteksi dalam kontainer yang dapat disebut dengan koleksi. Amazon Rekognition menyediakan operasi API tambahan yang dapat Anda gunakan untuk mencari informasi sama yang berulang pada wajah untuk pencocokan wajah. Untuk informasi selengkapnya, lihat [Operasi API berbasis penyimpanan](#).

Menggunakan AWS SDK atau HTTP untuk memanggil operasi Amazon Rekognition API

Anda dapat memanggil operasi Amazon Rekognition API menggunakan AWS SDK maupun secara langsung dengan menggunakan HTTP. Kecuali Anda tidak menghadapi permasalahan apa pun,

Anda harus selalu menggunakan AWS SDK. Contoh Java dalam bagian ini menggunakan [AWS SDK](#). File proyek Java tidak disediakan, tetapi Anda dapat menggunakan [AWS Toolkit for Eclipse](#) untuk mengembangkan aplikasi AWS menggunakan Java.

Contoh NET tersebut pada bagian ini menggunakan [AWS SDK for .NET](#). Anda dapat menggunakan [AWS Toolkit for Visual Studio](#) untuk mengembangkan aplikasi AWS menggunakan NET. Termasuk templat dan Penjelajah AWS yang bermanfaat untuk men-deploy aplikasi dan mengelola layanan.

[Referensi API](#) dalam panduan ini mencakup panggilan operasi Amazon Rekognition menggunakan HTTP. Untuk informasi referensi Java, lihat [AWS SDK for Java](#).

Titik akhir layanan Amazon Rekognition yang dapat Anda gunakan didokumentasikan di [Wilayah AWS dan titik akhir](#).

Saat memanggil Amazon Rekognition dengan HTTP, gunakan operasi POST HTTP.

Operasi API penyimpanan dan bukan penyimpanan

Amazon Rekognition menyediakan dua tipe operasi API. Yaitu operasi bukan penyimpanan yang membuat Amazon Rekognition tidak menyimpan informasi apa pun, serta operasi penyimpanan yang membuat Amazon Rekognition menyimpan informasi wajah tertentu.

Operasi bukan penyimpanan

Amazon Rekognition menyediakan operasi API bukan penyimpanan berikut untuk citra:

- [DetectLabels](#)
- [DetectFaces](#)
- [CompareFaces](#)
- [DetectModerationLabels](#)
- [DetectProtectiveEquipment](#)
- [RecognizeCelebrities](#)
- [DetectText](#)
- [GetCelebrityInfo](#)

Amazon Rekognition menyediakan operasi API bukan penyimpanan berikut untuk video:

- [StartLabelDetection](#)

- [StartFaceDetection](#)
- [StartPersonTracking](#)
- [StartCelebrityRecognition](#)
- [StartContentModeration](#)

Disebut sebagai operasi API bukan penyimpanan karena ketika Anda membuat panggilan operasi, Amazon Rekognition tidak menahan informasi apa pun yang ditemukan tentang citra input. Seperti semua operasi API Amazon Rekognition lainnya, tidak ada input bit citra yang ditahan oleh operasi API bukan penyimpanan.

Contoh skenario berikut menunjukkan tempat Anda mungkin mengintegrasikan operasi API bukan penyimpanan dalam aplikasi Anda. Skenario ini mengasumsikan bahwa Anda memiliki repositori citra lokal.

Example 1: Sebuah aplikasi yang menemukan citra di repositori lokal Anda yang berisi label tertentu

Pertama, Anda mendeteksi label (objek dan konsep) menggunakan operasi Amazon DetectLabels Rekognition di setiap gambar di repositori Anda dan membuat indeks sisi klien, seperti yang ditunjukkan berikut:

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

Kemudian, aplikasi Anda dapat mencari indeks ini untuk menemukan citra dalam repositori lokal Anda yang berisi label tertentu. Misalnya, tampilkan citra yang berisi pohon.

Setiap label yang Amazon Rekognition deteksi memiliki nilai kepercayaan yang terkait. Hal ini menunjukkan bahwa tingkat kepercayaan pada citra input berisi label itu. Secara opsional Anda dapat menggunakan nilai kepercayaan ini untuk melakukan pemfilteran sisi klien tambahan pada label tergantung pada persyaratan aplikasi Anda tentang tingkat kepercayaan pada deteksi. Misalnya, jika Anda memerlukan label yang tepat, Anda dapat memfilter dan hanya memilih label

dengan kepercayaan lebih tinggi (dengan ukuran 95% atau lebih tinggi). Jika aplikasi Anda tidak memerlukan nilai kepercayaan yang lebih tinggi, Anda dapat memilih untuk memfilter label dengan nilai kepercayaan yang lebih rendah (mendekati 50%).

Example 2: Sebuah aplikasi untuk menampilkan citra wajah yang kualitasnya disempurnakan

Pertama, Anda dapat mendeteksi wajah di setiap citra di repositori lokal Anda menggunakan operasi DetectFaces Amazon Rekognition dan membangun indeks sisi klien. Untuk setiap wajah, operasi mengembalikan metadata yang mencakup kotak pembatas, penanda wajah (misalnya, posisi mulut dan telinga), dan atribut wajah (misalnya, tipe kelamin). Anda dapat menyimpan metadata ini dalam indeks lokal sisi klien, seperti yang ditunjukkan berikut:

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

Dalam indeks ini, kunci primer adalah kombinasi dari kedua ImageID dan FaceID.

Kemudian, Anda dapat menggunakan informasi yang ada di indeks untuk meningkatkan kualitas citra ketika aplikasi Anda menampilkan citra tersebut dari repositori lokal Anda. Misalnya, Anda mungkin menambahkan kotak pembatas di sekitar wajah atau menyorotkan fitur wajah.

Operasi API berbasis penyimpanan

Amazon Rekognition Image mendukung operasi [IndexFaces](#), yang dapat Anda gunakan untuk mendeteksi wajah dalam citra dan menahan informasi tentang fitur wajah yang terdeteksi dalam koleksi Amazon Rekognition. Operasi ini adalah contoh dari operasi API berbasis–penyimpanan karena layanan tersebut menyimpan informasi pada server.

Amazon Rekognition Image menyediakan operasi penyimpanan API berikut:

- [IndexFaces](#)
- [ListFaces](#)
- [SearchFacesByImage](#)

- [SearchFaces](#)
- [DeleteFaces](#)
- [DescribeCollection](#)
- [DeleteCollection](#)
- [ListCollections](#)
- [CreateCollection](#)

Amazon Rekognition Video menyediakan operasi penyimpanan API berikut:

- [StartFaceSearch](#)
- [CreateStreamProcessor](#)

Untuk menyimpan informasi wajah, Anda harus terlebih dahulu membuat koleksi wajah di salah satu Wilayah AWS di akun Anda. Anda menentukan koleksi wajah ini ketika Anda memanggil operasi `IndexFaces`. Setelah membuat koleksi wajah dan menyimpan informasi fitur wajah untuk semua wajah, Anda dapat mencari kecocokan wajah pada koleksi tersebut. Misalnya, Anda dapat mendeteksi wajah terbesar dalam sebuah citra dan mencari wajah yang cocok di dalam koleksi dengan memanggil `searchFacesByImage`.

Informasi wajah disimpan dalam koleksi oleh `IndexFaces` yang dapat diakses oleh operasi Amazon Rekognition Video. Misalnya, Anda dapat menelusuri video untuk seseorang yang wajahnya sesuai dengan yang ada dalam koleksi dengan menghubungi [StartFaceSearch](#).

Untuk informasi selengkapnya tentang membuat dan mengelola koleksi, lihat [Mencari wajah dalam koleksi](#).

Note

Koleksi menyimpan vektor wajah, yang merupakan representasi matematis dari wajah. Koleksi tidak menyimpan gambar wajah.

Example 1: Aplikasi yang mengautentikasi akses ke gedung

Anda memulainya dengan membuat koleksi wajah untuk menyimpan lencana citra yang dipindai menggunakan operasi `IndexFaces`, yang mengekstrak wajah dan menyimpannya sebagai vektor citra yang dapat dicari. Kemudian, ketika seorang karyawan memasuki gedung, citra wajah karyawan

ditangkap dan dikirim ke operasi `SearchFacesByImage`. Jika kecocokan wajah menghasilkan skor kesamaan yang cukup tinggi (katakanlah 99%), Anda dapat mengautentikasi karyawan tersebut.

Versioning model

Amazon Rekognition menggunakan model deep learning untuk melakukan deteksi wajah dan mencari wajah dalam koleksi. Amazon Rekognition akan terus meningkatkan keakuratan modelnya berdasarkan umpan balik pelanggan dan kemajuan dalam penelitian deep learning. Perbaikan ini dikirim sebagai pembaruan pada model. Contohnya, dengan model versi 1.0, [IndexFaces](#) dapat mengindeks 15 wajah terbesar dalam sebuah citra. Versi yang lebih baru dari model ini memungkinkan `IndexFaces` untuk mengindeks 100 wajah terbesar dalam sebuah citra.

Ketika Anda membuat koleksi baru, koleksi tersebut terkait dengan versi terbaru dari model. Untuk meningkatkan akurasi, model ini terkadang perlu diperbarui.

Ketika versi baru dari model sudah dirilis, hal berikut akan terjadi:

- Koleksi baru yang Anda buat terkait dengan model terbaru. Wajah yang Anda tambahkan ke koleksi baru dengan menggunakan [IndexFaces](#) terdeteksi dengan menggunakan model terbaru.
- Koleksi Anda yang ada terus menggunakan versi model yang dengannya mereka dibuat. Vektor wajah yang disimpan dalam koleksi ini tidak diperbarui secara otomatis ke versi model terbaru.
- Wajah baru yang ditambahkan ke koleksi yang ada terdeteksi dengan menggunakan model yang sudah terkait dengan koleksi.

Versi model yang berbeda tidak kompatibel satu sama lainnya. Spesifiknya, jika citra diindeks ke beberapa koleksi yang menggunakan versi model berbeda, hasil dari pengenalan wajah untuk wajah yang terdeteksi sama adalah berbeda. Jika citra diindeks ke beberapa koleksi yang terkait dengan model yang sama, hasil dari pengenalan wajah adalah sama.

Aplikasi Anda mungkin menghadapi masalah kompatibilitas jika manajemen koleksi Anda tidak memperhitungkan pembaruan model. Anda dapat menentukan versi model koleksi yang digunakan dengan menggunakan bidang `FaceModelVersion` yang dikembalikan dalam bentuk respons dari operasi koleksi (misalnya, `CreateCollection`). Anda bisa mendapatkan versi model koleksi yang ada dengan memanggil [DescribeCollection](#). Untuk informasi selengkapnya, lihat [Menjelaskan koleksi](#).

Vektor wajah yang ada dalam koleksi tidak dapat diperbarui ke versi model yang lebih baru. Karena Amazon Rekognition tidak menyimpan bit citra sumber, Amazon Rekognition tidak dapat secara otomatis mengindeks ulang citra dengan menggunakan versi model.

Untuk menggunakan model terbaru pada wajah yang disimpan dalam koleksi yang ada, buat koleksi baru ([CreateCollection](#)) dan indeks ulang citra sumber ke dalam koleksi baru ([IndexFaces](#)). Anda perlu memperbarui setiap pengenalan wajah yang disimpan oleh aplikasi Anda karena pengenalan wajah dalam koleksi baru berbeda dari pengenalan wajah di dalam koleksi lama. Jika Anda tidak lagi memerlukan koleksi lama, Anda dapat menghapusnya dengan menggunakan [DeleteCollection](#).

Operasi stateless, seperti [DetectFaces](#), menggunakan model versi model terbaru.

Memulai dengan Amazon Rekognition

Bagian ini menyediakan topik untuk membantu Anda memulai menggunakan Amazon Rekognition. Jika Anda baru mengenal Amazon Rekognition, kami sarankan Anda terlebih dahulu meninjau konsep dan terminologi yang disajikan dalam [Bagaimana Amazon Rekognition bekerja](#).

Sebelum Anda dapat menggunakan Rekognition, Anda harus membuat akun AWS dan mendapatkan ID akun AWS. Anda juga ingin membuat pengguna, yang memungkinkan sistem Rekognition Amazon untuk menentukan apakah Anda memiliki izin yang diperlukan untuk mengakses sumber dayanya.

Setelah membuat akun, Anda akan ingin menginstal dan mengkonfigurasi AWS CLI dan AWS SDK. Ini AWS CLI memungkinkan Anda berinteraksi dengan Amazon Rekognition dan layanan lainnya melalui baris perintah, sedangkan AWS SDK memungkinkan Anda menggunakan bahasa pemrograman seperti Java dan Python untuk berinteraksi dengan Amazon Rekognition.

Setelah Anda mengatur AWS CLI dan AWS SDK, Anda dapat melihat beberapa contoh cara menggunakan keduanya. Anda juga dapat melihat beberapa contoh bagaimana berinteraksi dengan Amazon Rekognition menggunakan konsol tersebut.

Topik

- [Langkah 1: Siapkan akun AWS dan buat Pengguna](#)
- [Langkah 2: Siapkan AWS CLI dan AWS SDK](#)
- [Langkah 3: Memulai menggunakan API AWS CLI dan AWS SDK](#)
- [Langkah 4: Mulai menggunakan konsol Amazon Rekognition](#)

Langkah 1: Siapkan akun AWS dan buat Pengguna

Sebelum Anda menggunakan Amazon Rekognition untuk pertama kalinya, Anda harus menyelesaikan tugas-tugas berikut:

1. Mendaftar untuk sebuah AWS akun.
2. Buat Pengguna.

Bagian panduan pengembang ini menjelaskan mengapa dan bagaimana Anda akan membuat AWS akun dan pengguna.

Topik

- [Buat AWS Akun dan Pengguna](#)

Buat AWS Akun dan Pengguna

Akun AWS

Saat Anda mendaftar untuk Amazon Web Services (AWS), akun AWS Anda secara otomatis terdaftar untuk semua layanan di AWS, termasuk Amazon Rekognition. Anda hanya akan dikenakan biaya untuk layanan yang digunakan.

Dengan Amazon Rekognition, Anda hanya membayar untuk sumber daya yang Anda gunakan.

Jika Anda adalah AWS pelanggan baru, Anda dapat memulai dengan Amazon Rekognition secara gratis. Untuk informasi selengkapnya, lihat [Tingkatan Gratis AWS](#).

Lihat [Mendaftar untuk Akun AWS](#) bagian yang akan datang untuk instruksi pembuatan akun.

Jika Anda sudah memiliki AWS akun, lewati pengaturan akun dan buat pengguna administratif.

Pengguna

Layanan di AWS, seperti Amazon Rekognition, mewajibkan Anda memberikan kredensial saat Anda mengaksesnya. Layanan kemudian dapat menentukan apakah Anda memiliki izin untuk mengakses sumber daya yang dimiliki oleh layanan tersebut.

Anda dapat membuat kunci akses untuk AWS akun Anda untuk mengakses AWS CLI atau API saat menggunakan konsol memerlukan kata sandi Anda. Namun, kami tidak merekomendasikan Anda untuk mengakses AWS menggunakan kredensial untuk pengguna root akun AWS Anda. Sebagai gantinya, kami menyarankan Anda menggunakan AWS Identity and Access Management (IAM) untuk membuat pengguna administratif.

Anda kemudian dapat mengakses AWS dengan menggunakan URL khusus dan kredensi pengguna administratif tersebut.

Jika Anda mendaftar AWS, tetapi Anda belum membuat pengguna sendiri, Anda dapat membuatnya dengan menggunakan konsol IAM. Lihat [Buat pengguna dengan akses administratif](#) bagian yang akan datang untuk petunjuk tentang cara membuat pengguna administratif.

Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka <https://portal.aws.amazon.com/billing/signup>.
2. Ikuti petunjuk online.

Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWS dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan [tugas yang memerlukan akses pengguna root](#).

AWS mengirim Anda email konfirmasi setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan mengunjungi <https://aws.amazon.com/> dan memilih Akun Saya.

Buat pengguna dengan akses administratif

Setelah Anda mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke [AWS Management Console](#) sebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

Untuk bantuan masuk dengan menggunakan pengguna root, lihat [Masuk sebagai pengguna root](#) di AWS Sign-In Panduan Pengguna.

2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

Untuk petunjuk, lihat [Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root \(konsol\) Anda](#) di Panduan Pengguna IAM.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

Untuk mendapatkan petunjuk, silakan lihat [Mengaktifkan AWS IAM Identity Center](#) di Panduan Pengguna AWS IAM Identity Center .

2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna.

Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat [Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM](#) di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat [Masuk ke portal AWS akses](#) di Panduan AWS Sign-In Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

Untuk petunjuknya, lihat [Membuat set izin](#) di Panduan AWS IAM Identity Center Pengguna.

2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

Untuk petunjuk, lihat [Menambahkan grup](#) di Panduan AWS IAM Identity Center Pengguna.

Langkah 2: Siapkan AWS CLI dan AWS SDK

Topik

- [Memberikan akses programatis](#)
- [Menggunakan Rekognition dengan SDK AWS](#)

Langkah-langkah berikut menunjukkan cara menginstal AWS Command Line Interface (AWS CLI) dan AWS SDK yang digunakan contoh dalam dokumentasi ini. Ada sejumlah cara berbeda untuk mengautentikasi panggilan AWS SDK. Contoh dalam panduan ini mengasumsikan bahwa Anda menggunakan profil kredensial default untuk memanggil AWS CLI perintah dan operasi AWS SDK API.

Untuk daftar AWS Wilayah yang tersedia, lihat [Wilayah dan Titik Akhir](#) di Referensi Umum Amazon Web Services

Ikuti langkah-langkah untuk mengunduh dan mengonfigurasi AWS SDK.

Untuk mengatur AWS CLI dan AWS SDK

1. Unduh dan instal [AWS CLI](#) dan AWS SDK yang ingin Anda gunakan. Panduan ini memberikan contoh untuk AWS CLI, Java, Python, Ruby, Node.js, PHP, .NET, dan JavaScript Untuk informasi tentang menginstal AWS SDK, lihat [Alat untuk Amazon Web Services](#).
2. Buat access key untuk pengguna yang Anda buat dalam [Buat AWS Akun dan Pengguna](#).
 - a. Masuk ke AWS Management Console dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.
 - b. Di panel navigasi, pilih Pengguna.
 - c. Pilih nama pengguna yang Anda buat di [Buat AWS Akun dan Pengguna](#).
 - d. Pilih tab Kredensial keamanan.
 - e. Pilih Buat access key. Lalu pilih Unduh file .csv untuk menyimpan access key ID dan secret access key ke file CSV dalam komputer Anda. Simpan file di lokasi aman. Anda tidak akan memiliki akses ke secret access key lagi setelah menutup kotak dialog ini. Setelah mengunduh file CSV, pilih Tutup.
3. Jika Anda telah menginstal AWS CLI, Anda dapat [mengonfigurasi kredensial dan wilayah untuk sebagian besar AWS SDK dengan memasukkan `aws configure` pada prompt perintah](#). Jika tidak, gunakan arahan berikut.
4. Pada komputer Anda, navigasi ke direktori beranda Anda, dan buat direktori `.aws`. Pada sistem berbasis Unix, seperti Linux atau macOS, ini berada di lokasi berikut:

```
~/ .aws
```

Di Windows, ini ada di lokasi berikut:


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

- Di direktori `.aws`, buat file baru bernama `credentials`.
- Buka file CSV kredensial yang Anda buat di langkah 2 dan salin isinya ke file `credentials` menggunakan format berikut:

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

Ganti access key ID dan secret access key Anda dengan `your_access_key_id` dan `your_secret_access_key`.

- Simpan file `Credentials` dan hapus file CSV.
- Di direktori `.aws`, buat file baru bernama `config`.
- Buka file `config` dan masukkan wilayah Anda dalam format berikut.

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

Ganti Wilayah AWS yang Anda inginkan (misalnya, `us-west-2`) dengan `your_aws_region`.

Note

Jika Anda tidak memilih wilayah, maka `us-east-1` akan digunakan secara default.

- Simpan file `config`.

Memberikan akses programatis

Anda dapat menjalankan contoh AWS CLI dan kode dalam panduan ini di komputer lokal atau AWS lingkungan lain, seperti instans Amazon Elastic Compute Cloud. Untuk menjalankan contoh, Anda perlu memberikan akses ke operasi AWS SDK yang digunakan contoh.

Topik

- [Menjalankan kode di komputer lokal Anda](#)
- [Menjalankan kode di AWS lingkungan](#)

Menjalankan kode di komputer lokal Anda

Untuk menjalankan kode di komputer lokal, sebaiknya gunakan kredensial jangka pendek untuk memberikan akses pengguna ke operasi AWS SDK. Untuk informasi spesifik tentang menjalankan contoh kode AWS CLI dan pada komputer lokal, lihat [Menggunakan profil di komputer lokal Anda](#).

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar. AWS Management Console Cara untuk memberikan akses terprogram tergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses programatis, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. <ul style="list-style-type: none"> Untuk AWS CLI, lihat Mengkonfigurasi yang akan AWS CLI digunakan AWS IAM Identity Center dalam Panduan AWS Command Line Interface Pengguna. Untuk AWS SDK, alat, dan AWS API, lihat otentikasi Pusat Identitas IAM di Panduan Referensi AWS SDK dan Alat.
IAM	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	Mengikuti petunjuk dalam Menggunakan kredensial sementara dengan AWS sumber daya di Panduan Pengguna IAM.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
IAM	(Tidak direkomendasikan) Gunakan kredensial jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	<p>Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.</p> <ul style="list-style-type: none"> • Untuk mengetahui AWS CLI, lihat Mengautentikasi menggunakan kredensial pengguna IAM di Panduan Pengguna. AWS Command Line Interface • Untuk AWS SDK dan alat bantu, lihat Mengautentikasi menggunakan kredensial jangka panjang di Panduan Referensi AWS SDK dan Alat. • Untuk AWS API, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Menggunakan profil di komputer lokal Anda

Anda dapat menjalankan contoh AWS CLI dan kode dalam panduan ini dengan kredensi jangka pendek yang Anda buat. [Menjalankan kode di komputer lokal Anda](#) Untuk mendapatkan kredensi dan informasi pengaturan lainnya, contoh menggunakan profil bernama `profile-name` Misalnya:

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

Pengguna yang diwakili profil harus memiliki izin untuk memanggil operasi SDK Rekognition dan operasi SDK AWS lainnya yang diperlukan oleh contoh.

Untuk membuat profil yang sesuai dengan contoh kode AWS CLI dan, pilih salah satu dari berikut ini. Pastikan nama profil yang Anda buat adalah `profile-name`.

- Pengguna yang dikelola oleh IAM — Ikuti petunjuk di [Beralih ke peran IAM \(AWS CLI\)](#).
- Identitas tenaga kerja (Pengguna dikelola oleh AWS IAM Identity Center) — Ikuti petunjuk di [Mengonfigurasi AWS CLI](#) untuk digunakan. AWS IAM Identity Center Untuk contoh kode, sebaiknya gunakan Integrated Development Environment (IDE), yang mendukung AWS Toolkit yang mengaktifkan otentikasi melalui IAM Identity Center. Untuk contoh Java, lihat [Mulai membangun dengan Java](#). Untuk contoh Python, lihat [Mulai membangun dengan Python](#). Untuk informasi selengkapnya, lihat [kredensial Pusat Identitas IAM](#).

Note

Anda dapat menggunakan kode untuk mendapatkan kredensi jangka pendek. Untuk informasi selengkapnya, lihat [Beralih ke peran IAM \(AWS API\)](#). Untuk Pusat Identitas IAM, dapatkan kredensi jangka pendek untuk suatu peran dengan mengikuti instruksi di [Mendapatkan kredensial peran IAM](#) untuk akses CLI.

Menjalankan kode di AWS lingkungan

Anda tidak boleh menggunakan kredensial pengguna untuk menandatangani panggilan AWS SDK di AWS lingkungan, seperti kode produksi yang berjalan dalam suatu fungsi. AWS Lambda Sebagai gantinya, Anda mengonfigurasi peran yang menentukan izin yang dibutuhkan kode Anda. Anda kemudian melampirkan peran ke lingkungan tempat kode Anda berjalan. Cara Anda melampirkan peran dan membuat kredensial sementara tersedia bervariasi tergantung pada lingkungan tempat kode Anda berjalan:

- AWS Lambda fungsi — Gunakan kredensial sementara yang secara otomatis disediakan Lambda ke fungsi Anda saat mengasumsikan peran eksekusi fungsi Lambda. Kredensialnya tersedia di variabel lingkungan Lambda. Anda tidak perlu menentukan profil. Untuk informasi selengkapnya, silakan lihat [Peran eksekusi Lambda](#).
- Amazon EC2 — Gunakan penyedia kredensial titik akhir metadata instans Amazon EC2. Penyedia secara otomatis membuat dan menyegarkan kredensial untuk Anda menggunakan profil instans Amazon EC2 yang Anda lampirkan ke instans Amazon EC2. Untuk informasi selengkapnya, lihat [Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2](#)

- Amazon Elastic Container Service — Gunakan penyedia kredensi Container. Amazon ECS mengirim dan menyegarkan kredensial ke titik akhir metadata. Peran IAM tugas yang Anda tentukan menyediakan strategi untuk mengelola kredensial yang digunakan aplikasi Anda. Untuk informasi selengkapnya, lihat [Berinteraksi dengan layanan AWS](#).

Untuk informasi selengkapnya tentang penyedia kredensi, lihat Penyedia kredensi [terstandarisasi](#).

Menggunakan Rekognition dengan SDK AWS

AWS kit pengembangan perangkat lunak (SDK) tersedia untuk banyak bahasa pemrograman populer. Setiap SDK menyediakan API, contoh kode, dan dokumentasi yang memudahkan developer untuk membangun aplikasi dalam bahasa pilihan mereka.

Dokumentasi SDK	Contoh kode
AWS SDK for C++	AWS SDK for C++ contoh kode
AWS CLI	AWS CLI contoh kode
AWS SDK for Go	AWS SDK for Go contoh kode
AWS SDK for Java	AWS SDK for Java contoh kode
AWS SDK for JavaScript	AWS SDK for JavaScript contoh kode
AWS SDK for Kotlin	AWS SDK for Kotlin contoh kode
AWS SDK for .NET	AWS SDK for .NET contoh kode
AWS SDK for PHP	AWS SDK for PHP contoh kode
AWS Tools for PowerShell	Alat untuk contoh PowerShell kode
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) contoh kode
AWS SDK for Ruby	AWS SDK for Ruby contoh kode
AWS SDK for Rust	AWS SDK for Rust contoh kode
AWS SDK untuk SAP ABAP	AWS SDK untuk SAP ABAP contoh kode

Dokumentasi SDK	Contoh kode
AWS SDK for Swift	AWS SDK for Swift contoh kode

Untuk contoh khusus untuk Rekognition, lihat. [Contoh kode untuk Amazon AWS Rekognition menggunakan SDK](#)

Ketersediaan contoh

Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik pada bagian bawah halaman ini.

Langkah 3: Memulai menggunakan API AWS CLI dan AWS SDK

Setelah menyiapkan AWS CLI dan AWS SDK yang ingin Anda gunakan, Anda dapat membangun aplikasi yang menggunakan Amazon Rekognition. Topik berikut menunjukkan kepada Anda cara memulai dengan Amazon Rekognition Image dan Amazon Rekognition Video.

- [Bekerja dengan citra](#)
- [Bekerja dengan analisis video tersimpan](#)
- [Bekerja dengan acara video streaming](#)

Memformat contoh AWS CLI

AWS CLI Contoh dalam panduan ini diformat untuk sistem operasi Linux. Untuk menggunakan sampel dengan Microsoft Windows, Anda perlu mengubah format parameter `--image` JSON, dan mengubah jeda baris dari garis miring terbalik (`\`) menjadi tanda sisipan (`^`). Untuk informasi selengkapnya tentang pemformatan JSON, lihat [Penentuan Nilai Parameter untuk Antarmuka Baris Perintah AWS](#).

Berikut ini adalah contoh AWS CLI perintah yang diformat untuk Microsoft Windows (perhatikan bahwa perintah ini tidak akan berjalan sebagaimana adanya, mereka hanya memformat contoh):

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

```
--region region-name
```

Anda juga dapat memberikan versi singkat dari JSON yang bekerja pada Microsoft Windows dan Linux.

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

Untuk informasi selengkapnya, lihat, [Menggunakan Singkatan Sintaksis dengan Antarmuka Baris Perintah AWS](#).

Langkah selanjutnya

[Langkah 4: Mulai menggunakan konsol Amazon Rekognition](#)

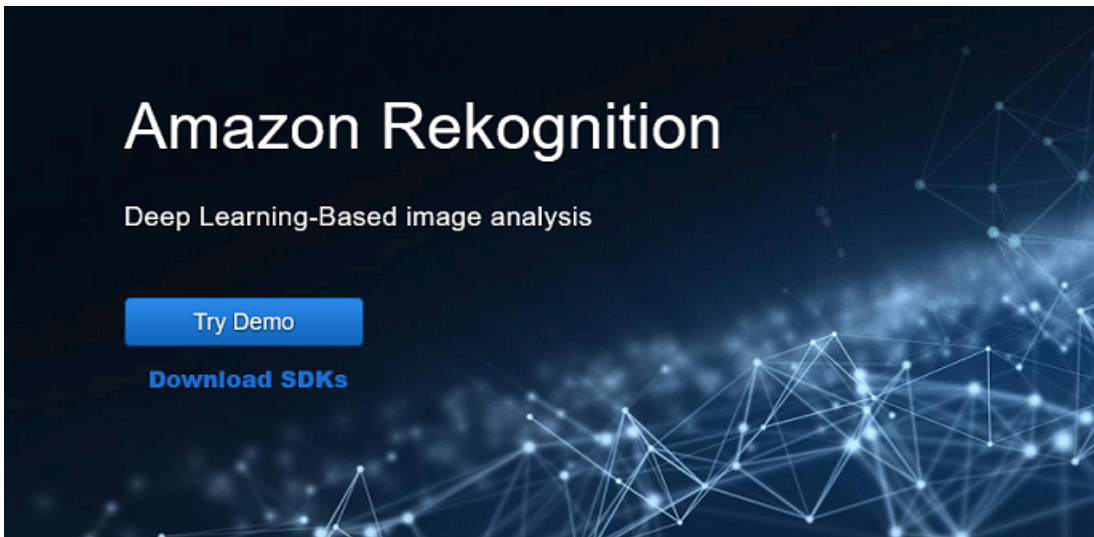
Langkah 4: Mulai menggunakan konsol Amazon Rekognition

Bagian ini menunjukkan cara menggunakan subset dari kemampuan Amazon Rekognition seperti deteksi objek dan adegan, analisis wajah, dan perbandingan wajah dalam satu set citra. Untuk informasi selengkapnya, lihat [Bagaimana Amazon Rekognition bekerja](#). Anda juga dapat menggunakan Amazon Rekognition API AWS CLI atau untuk mendeteksi objek dan pemandangan, mendeteksi wajah, dan membandingkan dan mencari wajah. Untuk informasi selengkapnya, lihat [Langkah 3: Memulai menggunakan API AWS CLI dan AWS SDK](#).

Bagian ini juga menunjukkan cara melihat CloudWatch metrik Amazon agregat untuk Rekognition dengan menggunakan konsol Rekognition.

Topik

- [Siapkan izin konsol](#)
- [Latihan 1: Mendeteksi objek dan adegan \(Konsol\)](#)
- [Latihan 2: Analisis wajah pada citra \(konsol\)](#)
- [Latihan 3: Bandingkan wajah dalam citra \(konsol\)](#)
- [Latihan 4: Lihat metrik agregat \(konsol\)](#)



Siapkan izin konsol

Untuk menggunakan konsol Rekognition, Anda harus memiliki izin yang sesuai untuk peran atau akun yang mengakses konsol. Untuk beberapa operasi, Rekognition akan secara otomatis membuat bucket Amazon S3 untuk menyimpan file yang ditangani selama operasi. Jika Anda ingin menyimpan file latihan di bucket selain bucket konsol ini, Anda akan memerlukan izin tambahan.

Mengizinkan akses konsol

Untuk menggunakan konsol Rekognition, Anda dapat menggunakan kebijakan IAM seperti berikut ini, yang mencakup Amazon S3 dan konsol Rekognition. Untuk informasi tentang menetapkan izin, lihat Menetapkan izin.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RekognitionFullAccess",
      "Effect": "Allow",
      "Action": [
        "rekognition:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "RekognitionConsoleS3BucketSearchAccess",
      "Effect": "Allow",
```



```
    "Action": [
      "s3:ListAllMyBuckets",
      "s3:ListBucket",
      "s3:GetBucketAcl",
      "s3:GetBucketLocation"
    ],
    "Resource": "*"
  },
  {
    "Sid": "RekognitionConsoleS3BucketFirstUseSetupAccess",
    "Effect": "Allow",
    "Action": [
      "s3:CreateBucket",
      "s3:PutBucketVersioning",
      "s3:PutLifecycleConfiguration",
      "s3:PutEncryptionConfiguration",
      "s3:PutBucketPublicAccessBlock",
      "s3:PutCors",
      "s3:GetCors"
    ],
    "Resource": "arn:aws:s3::rekognition-custom-projects-*"
  },
  {
    "Sid": "RekognitionConsoleS3BucketAccess",
    "Effect": "Allow",
    "Action": [
      "s3:ListBucket",
      "s3:GetBucketLocation",
      "s3:GetBucketVersioning"
    ],
    "Resource": "arn:aws:s3::rekognition-custom-projects-*"
  },
  {
    "Sid": "RekognitionConsoleS3ObjectAccess",
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:HeadObject",
      "s3:DeleteObject",
      "s3:GetObjectAcl",
      "s3:GetObjectTagging",
      "s3:GetObjectVersion",
      "s3:PutObject"
    ],
  },
```

```

    "Resource": "arn:aws:s3::rekognition-custom-projects-*/*"
  },
  {
    "Sid": "RekognitionConsoleManifestAccess",
    "Effect": "Allow",
    "Action": [
      "groundtruthlabeling:*",
    ],
    "Resource": "*"
  },
  {
    "Sid": "RekognitionConsoleTagSelectorAccess",
    "Effect": "Allow",
    "Action": [
      "tag:GetTagKeys",
      "tag:GetTagValues"
    ],
    "Resource": "*"
  },
  {
    "Sid": "RekognitionConsoleKmsKeySelectorAccess",
    "Effect": "Allow",
    "Action": [
      "kms:ListAliases"
    ],
    "Resource": "*"
  }
]
}

```

Mengakses bucket Amazon S3 eksternal

Saat pertama kali membuka konsol Rekognition di AWS Region baru, Rekognition membuat bucket (bucket konsol) yang digunakan untuk menyimpan file proyek. Atau, Anda dapat menggunakan bucket Amazon S3 (bucket eksternal) Anda sendiri untuk mengunggah gambar atau file manifes ke konsol. Untuk menggunakan bucket eksternal, tambahkan blok kebijakan berikut ke kebijakan sebelumnya. Ganti ember saya dengan nama ember.

```

{
    "Sid": "s3ExternalBucketPolicies",
    "Effect": "Allow",

```

```
    "Action": [
      "s3:GetBucketAcl",
      "s3:GetBucketLocation",
      "s3:GetObject",
      "s3:GetObjectAcl",
      "s3:GetObjectVersion",
      "s3:GetObjectTagging",
      "s3:ListBucket",
      "s3:PutObject"
    ],
    "Resource": [
      "arn:aws:s3:::my-bucket*"
    ]
  }
```

Menetapkan izin

Untuk memberikan akses, menambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center (penerus AWS Single Sign-On):

Buat rangkaian izin. Ikuti petunjuk di [Membuat izin yang ditetapkan](#) di Panduan Pengguna AWS IAM Identity Center (penerus AWS Single Sign-On).

- Pengguna yang dikelola di IAM melalui penyedia identitas:

Buat peran untuk federasi identitas. Ikuti instruksi dalam [Membuat peran untuk penyedia identitas pihak ketiga \(federasi\)](#) di Panduan Pengguna IAM.

- Pengguna IAM:

- Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam [Membuat peran untuk pengguna IAM](#) dalam Panduan Pengguna IAM.
- (Tidak disarankan) Pasang kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti instruksi dalam [Menambahkan izin ke pengguna \(konsol\)](#) dalam Panduan Pengguna IAM.

Latihan 1: Mendeteksi objek dan adegan (Konsol)

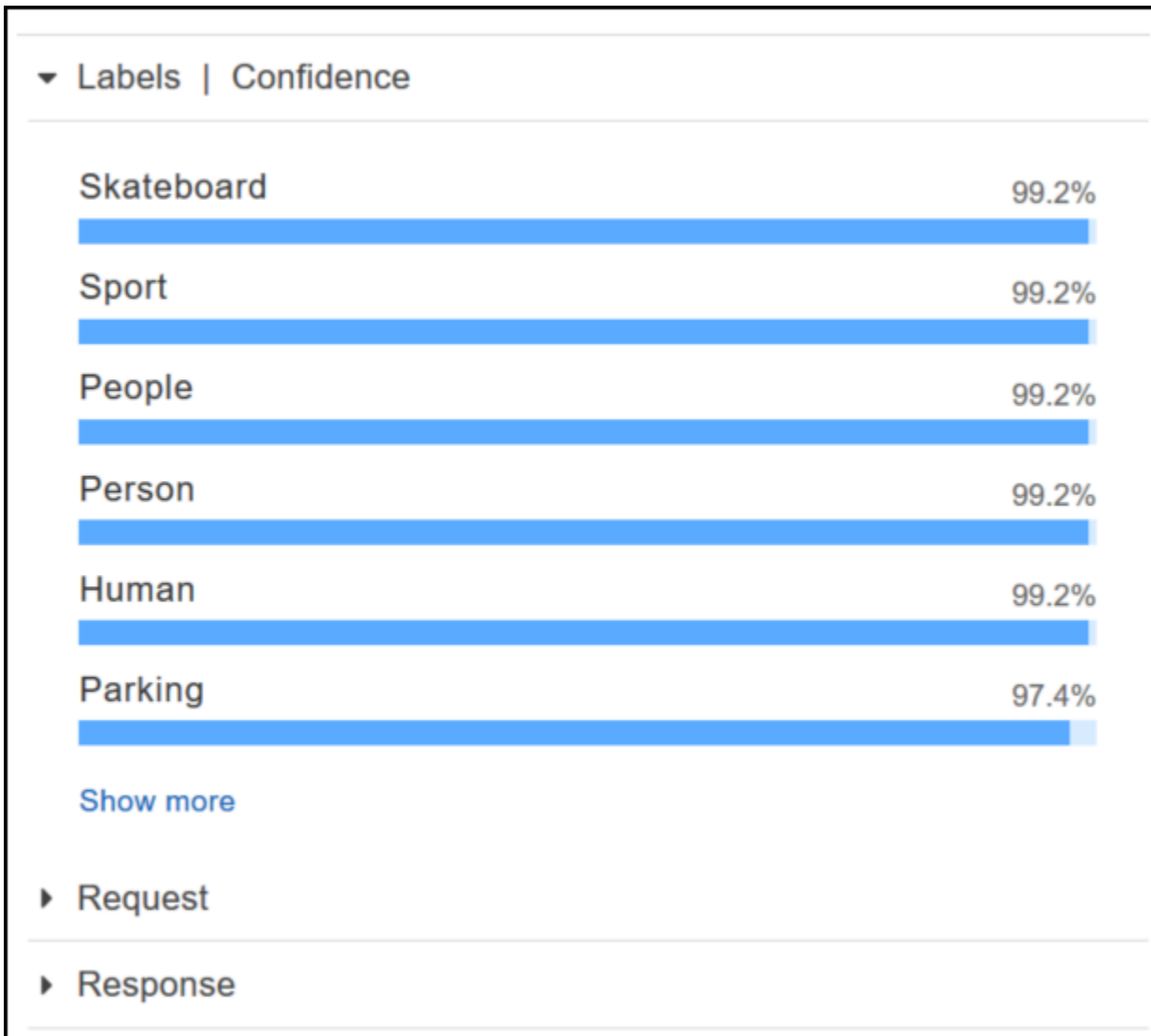
Bagian ini menunjukkan cara, pada tingkat yang sangat tinggi, kemampuan deteksi objek dan adegan Amazon Rekognition bekerja. Ketika Anda menentukan citra sebagai input, layanan mendeteksi objek

dan adegan dalam citra dan mengembalikannya bersama dengan persentase skor kepercayaan untuk setiap objek dan adegan.

Misalnya, Amazon Rekognition mendeteksi objek dan adegan berikut dalam citra sampel: papan luncur, olahraga, orang, oto, mobil dan kendaraan.



Amazon Rekognition juga mengembalikan skor kepercayaan untuk setiap objek yang terdeteksi dalam citra sampel, seperti yang ditunjukkan dalam respons sampel berikut.



Untuk melihat semua nilai kepercayaan yang ditampilkan dalam respons, pilih Tampilkan lebih banyak di panel Label | Kepercayaan.

Anda juga dapat melihat permintaan ke API dan respons dari API sebagai referensi.

Permintaan

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

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

Respons

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

Untuk informasi selengkapnya, lihat [Bagaimana Amazon Rekognition bekerja](#).

Deteksi objek dan adegan dalam citra yang Anda berikan

Anda dapat mengunggah citra yang Anda miliki atau memberikan URL ke citra sebagai input di konsol Amazon Rekognition. Amazon Rekognition mengembalikan objek dan adegan, skor kepercayaan untuk setiap objek, dan adegan mendeteksi dalam citra yang Anda berikan.

Note

citra harus berukuran kurang dari 5MB dan harus dalam format JPEG atau PNG.

Untuk mendeteksi objek dan adegan dalam citra yang Anda berikan

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>.

2. Pilih Deteksi label.
3. Lakukan salah satu dari hal berikut ini:
 - Unggah citra – Pilih Unggah, buka lokasi tempat Anda menyimpan citra, lalu pilih citra.
 - Gunakan URL – Ketik URL di kotak teks, lalu pilih Buka.
4. Lihat skor kepercayaan setiap label yang terdeteksi di panel Label | Kepercayaan.

Untuk opsi analisis gambar lainnya, lihat [the section called “Bekerja dengan citra”](#).

Mendeteksi objek dan orang dalam video yang Anda berikan

Anda dapat mengunggah video yang Anda berikan sebagai masukan di konsol Amazon Rekognition. Amazon Rekognition mengembalikan orang, objek, dan label yang terdeteksi dalam video.

Note

Video demo tidak boleh lebih dari satu menit atau lebih besar dari 30 MB. Itu harus dalam format file MP4 dan dikodekan menggunakan codec H.264.

Untuk mendeteksi objek dan orang dalam video yang Anda berikan

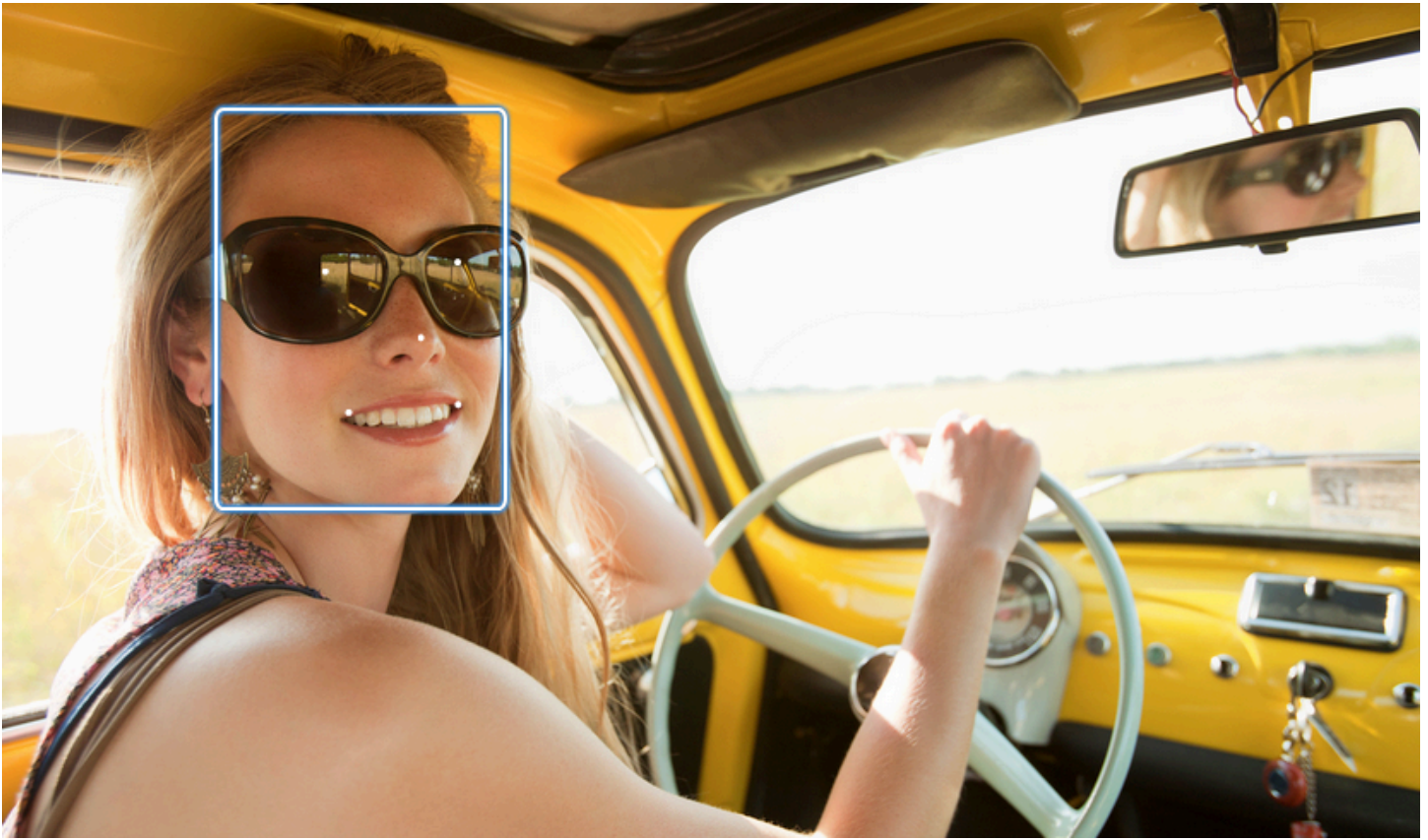
1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>.
2. Pilih Analisis Video Tersimpan dari bilah navigasi.
3. Di bawah Pilih sampel atau unggah milik Anda, pilih Video Anda sendiri dari menu tarik-turun.
4. Seret dan lepas video Anda atau pilih video Anda dari lokasi tempat Anda menyimpannya.

Untuk opsi analisis video lainnya, lihat [the section called “Bekerja dengan analisis video tersimpan”](#) atau [the section called “Bekerja dengan acara video streaming”](#).

Latihan 2: Analisis wajah pada citra (konsol)

Bagian ini menunjukkan cara menggunakan konsol Amazon Rekognition untuk mendeteksi wajah dan menganalisis atribut wajah dalam citra. Ketika Anda memberikan citra yang berisi wajah sebagai input, layanan mendeteksi wajah dalam citra, menganalisis atribut wajah, lalu mengembalikan persentase skor kepercayaan untuk wajah dan atribut wajah yang terdeteksi dalam citra. Untuk informasi selengkapnya, lihat [Bagaimana Amazon Rekognition bekerja](#).

Misalnya, jika Anda memilih citra sampel berikut sebagai input, Amazon Rekognition mendeteksinya sebagai wajah dan mengembalikan skor kepercayaan untuk wajah dan atribut wajah yang terdeteksi.



Berikut ini menunjukkan respons sampel.

▼ 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](#)

Jika ada beberapa wajah dalam citra input, Rekognition mendeteksi hingga 100 wajah dalam citra. Setiap wajah yang terdeteksi ditandai dengan kotak. Ketika Anda klik area yang ditandai dengan kotak di wajah, Rekognition menampilkan skor kepercayaan wajah tersebut dan atributnya yang terdeteksi di panel Wajah | Kepercayaan.

Analisis wajah dalam citra yang Anda berikan

Anda dapat mengunggah citra Anda sendiri atau memberikan URL untuk citra di konsol Amazon Rekognition.

Note

Citra harus berukuran kurang dari 5MB dan harus dalam format JPEG atau PNG.

Untuk menganalisis wajah dalam citra yang Anda berikan

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>.
2. Pilih Analisis wajah.
3. Lakukan salah satu dari hal berikut ini:
 - Unggah citra – Pilih Unggah, buka lokasi tempat Anda menyimpan citra, lalu pilih citra.
 - Gunakan URL – Ketik URL di kotak teks, lalu pilih Buka.
4. Lihat skor kepercayaan dari salah satu wajah yang terdeteksi dan atribut wajahnya di panel Wajah | Kepercayaan.
5. Jika ada beberapa wajah dalam citra, pilih salah satu wajah lain untuk melihat atribut dan skornya.

Latihan 3: Bandingkan wajah dalam citra (konsol)

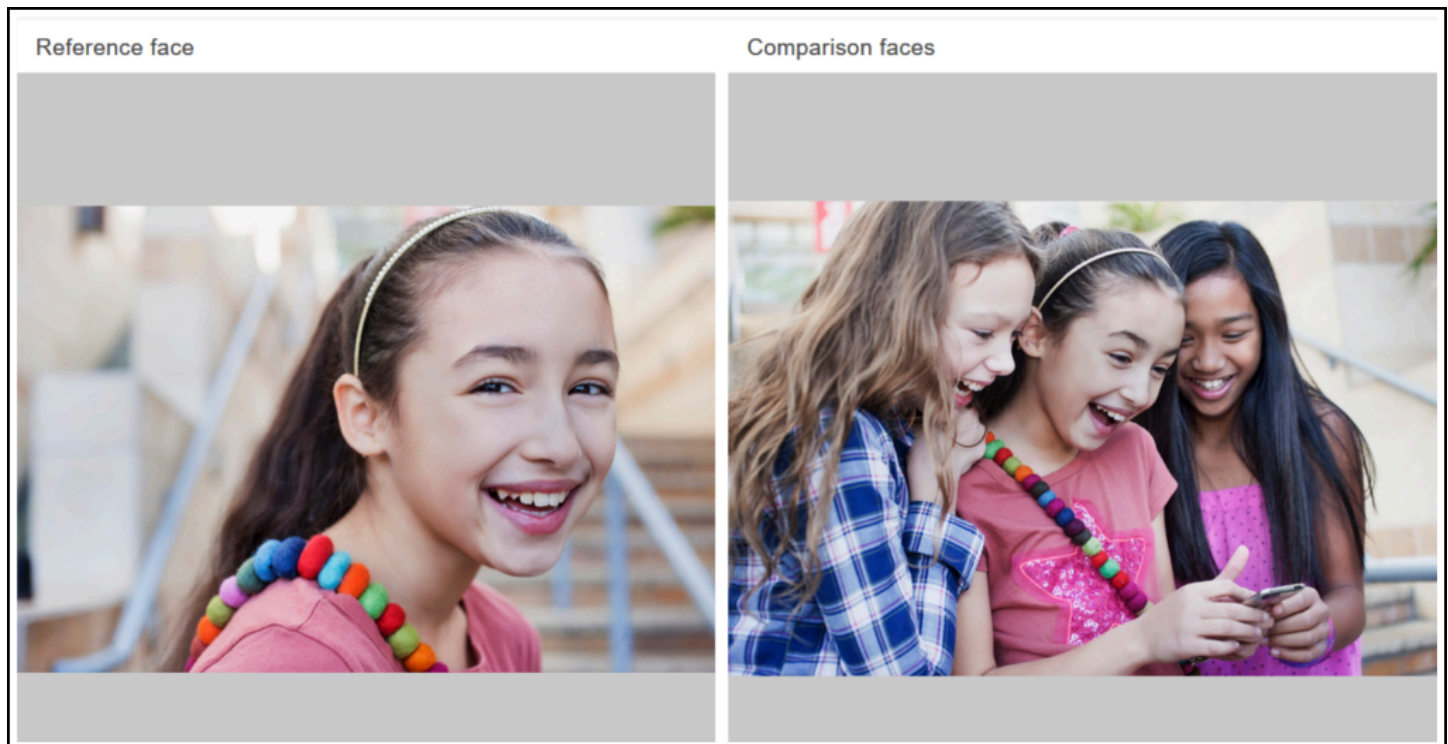
Bagian ini menunjukkan cara menggunakan konsol Amazon Rekognition untuk membandingkan wajah dalam satu set citra dengan beberapa wajah di dalamnya. Jika Anda menentukan citra Wajah referensi (sumber) dan Wajah Perbandingan (target), Rekognition membandingkan wajah terbesar dalam citra sumber (yaitu, wajah referensi) dengan jumlah wajah yang terdeteksi hingga 100 wajah dalam citra target (yaitu, wajah perbandingan), lalu temukan seberapa dekat wajah dalam sumber yang cocok dengan wajah dalam citra target. Skor kesamaan untuk setiap perbandingan ditampilkan dalam panel Hasil.

Jika citra target berisi beberapa wajah, Rekognition cocok dengan wajah dalam citra sumber dengan jumlah wajah yang terdeteksi hingga 100 wajah dalam citra target, lalu menetapkan skor kesamaan untuk setiap kecocokan.

Jika citra sumber berisi beberapa wajah, layanan mendeteksi wajah terbesar dalam citra sumber dan menggunakannya untuk membandingkan dengan setiap wajah yang terdeteksi dalam citra target.



Untuk informasi selengkapnya, lihat [Membandingkan wajah dalam citra](#).

Misalnya, dengan citra sampel yang ditampilkan di sebelah kiri sebagai citra sumber dan citra sampel di sebelah kanan sebagai citra target, Rekognition mendeteksi wajah dalam citra sumber, membandingkannya dengan setiap wajah yang terdeteksi pada citra target, dan menampilkan skor kesamaan untuk setiap pasangan.






Berikut ini menunjukkan wajah yang terdeteksi pada citra target dan skor kesamaan untuk setiap wajah.

▼ Results

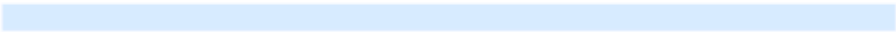
 ↔ 



Similarity 92%



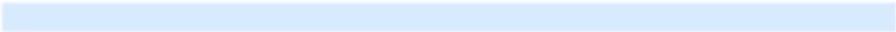
 ↔ 

Similarity 0%



 ↔ 

Similarity 0%



► Request

► Response

Bandingkan wajah dalam citra yang Anda berikan

Anda dapat mengunggah sumber Anda sendiri dan citra target untuk Rekognition untuk membandingkan wajah dalam citra atau Anda dapat menentukan URL untuk lokasi citra.

Note

Citra harus berukuran kurang dari 5MB dan harus dalam format JPEG atau PNG.

Untuk membandingkan wajah dalam citra Anda

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>.
2. Pilih Perbandingan wajah.
3. Untuk citra sumber, lakukan salah satu hal berikut ini:
 - Unggah citra – Pilih Unggah di sebelah kiri, buka lokasi di tempat Anda menyimpan citra sumber Anda, lalu pilih citra.
 - Gunakan URL – Ketik URL citra sumber Anda di kotak teks, lalu pilih Buka.
4. Untuk citra target, lakukan salah satu hal berikut:
 - Unggah citra – Pilih Unggah di sebelah kanan, buka lokasi di tempat Anda menyimpan citra sumber Anda, lalu pilih citra.
 - Gunakan URL – Ketik URL citra sumber Anda di kotak teks, lalu pilih Buka.
5. Rekognition cocok dengan wajah terbesar dalam citra sumber Anda dengan jumlah hingga 100 wajah dalam citra target lalu menampilkan skor kesamaan untuk setiap pasangan di panel Hasil.

Latihan 4: Lihat metrik agregat (konsol)

Panel metrik Amazon Rekognition menunjukkan grafik aktivitas untuk gabungan dari metrik Rekognition individual selama periode waktu tertentu. Misalnya, metrik gabungan `SuccessfulRequestCount` menunjukkan jumlah total permintaan yang berhasil ke semua operasi API Rekognition selama tujuh hari terakhir.

Tabel berikut mencantumkan grafik yang ditampilkan di panel metrik Rekognition dan metrik Rekognition yang sesuai. Untuk informasi selengkapnya, lihat [CloudWatchmetrik untuk Rekognition](#).

Grafik	Metrik Gabungan
Panggilan berhasil	<code>SuccessfulRequestCount</code>
Kesalahan klien	<code>UserErrorCount</code>

Grafik	Metrik Gabungan
Kesalahan server	ServerErrorCount
Di-throttling	ThrottledCount
Label yang terdeteksi	DetectedLabelCount
i	
Wajah yang terdeteksi	DetectedFaceCount

Setiap grafik menampilkan data metrik gabungan yang dikumpulkan selama periode waktu tertentu. Jumlah total data metrik gabungan untuk jangka waktu juga ditampilkan. Untuk melihat metrik panggilan API individual, pilih tautan di bawah setiap grafik.

Untuk memungkinkan pengguna mengakses panel metrik Rekognition, pastikan bahwa pengguna memiliki izin Rekognition yang sesuai. CloudWatch Misalnya, pengguna dengan izin kebijakan terkelola `AmazonRekognitionReadOnlyAccess` dan `CloudWatchReadOnlyAccess` dapat melihat panel metrik. Jika pengguna tidak memiliki izin yang diperlukan, saat pengguna membuka panel metrik, tidak ada grafik yang muncul. Untuk informasi selengkapnya, lihat [Identity and Access Management untuk Amazon Rekognition](#).

Untuk informasi lebih lanjut tentang pemantauan CloudWatch Rekognition dengan lihat. [Memantau Rekognition dengan Amazon CloudWatch](#)

Untuk melihat metrik gabungan (konsol)

1. Buka konsol Amazon Rekognition di <https://console.aws.amazon.com/rekognition/>.
2. Di panel navigasi, pilih Metrik.
3. Pada dropdown, pilih periode waktu yang diinginkan metrik.
4. Untuk memperbarui grafik, pilih tombol Segarkan kembali.
5. Untuk melihat CloudWatch metrik terperinci untuk metrik agregat tertentu, pilih Lihat detail di CloudWatch bawah grafik metrik.

Bekerja dengan gambar dan video

Anda dapat menggunakan operasi Amazon Rekognition API dengan tiga jenis media yang berbeda: gambar, video yang disimpan, dan video streaming. Bagian ini memberikan informasi umum tentang menulis kode yang mengakses Amazon Rekognition untuk memproses berbagai jenis media. Untuk panduan tentang praktik dan pertimbangan terbaik, lihat bagian masing-masing yang tercantum di bawah ini, tergantung pada jenis media yang Anda proses.

Bagian lain dalam panduan ini memberikan informasi tentang tipe analisis citra dan video tertentu, seperti pendeteksi wajah.

Topik

- [Bekerja dengan citra](#)
- [Bekerja dengan analisis video tersimpan](#)
- [Bekerja dengan acara video streaming](#)
- [Penanganan kesalahan](#)
- [Menggunakan Amazon Rekognition sebagai layanan resmi FedRAMP](#)

Bekerja dengan citra

Bagian ini mencakup tipe analisis yang dapat dilakukan Amazon Rekognition Image pada citra.

- [Deteksi objek dan adegan](#)
- [Deteksi dan perbandingan wajah](#)
- [Mencari wajah dalam koleksi](#)
- [Pengakuan selebriti](#)
- [Moderasi gambar](#)
- [Teks dalam deteksi gambar](#)

Ini dilakukan oleh operasi API bukan penyimpanan tempat Amazon Rekognition Image tidak mempertahankan informasi apa pun yang ditemukan oleh operasi. Tidak ada bit citra input yang bertahan oleh operasi API bukan penyimpanan. Untuk informasi selengkapnya, lihat [Operasi API penyimpanan dan bukan penyimpanan](#).

Amazon Rekognition Image juga dapat menyimpan metadata wajah dalam koleksi untuk pengambilan nanti. Untuk informasi selengkapnya, lihat [Mencari wajah dalam koleksi](#).

Pada bagian ini, Anda menggunakan operasi API Amazon Rekognition Image untuk menganalisis citra yang disimpan dalam bucket Amazon S3 dan bit citra yang dimuat dari sistem file lokal. Bagian ini juga mencakup tentang mendapatkan informasi orientasi citra dari citra .jpg.

Rekognition hanya menggunakan saluran RGB untuk melakukan inferensi. AWS merekomendasikan pengguna menghapus Saluran Alpha sebelum menggunakan Display untuk memeriksa perbandingan secara visual (secara manual oleh manusia).

Topik

- [Spesifikasi citra](#)
- [Menganalisis citra yang disimpan di bucket Amazon S3](#)
- [Menganalisis citra yang dimuat dari sistem file lokal](#)
- [Menampilkan kotak pembatas](#)
- [Mendapatkan orientasi citra dan koordinat kotak pembatas](#)

Spesifikasi citra

Operasi Amazon Rekognition Image dapat menganalisis citra dalam format .jpg atau .png.

Anda meneruskan bit citra ke operasi Amazon Rekognition Image sebagai bagian dari panggilan atau Anda mereferensikan objek Amazon S3 yang ada. Sebagai contoh menganalisis citra yang disimpan di bucket Amazon S3, lihat [Menganalisis citra yang disimpan di bucket Amazon S3](#). Untuk contoh meneruskan bit citra ke operasi API Amazon Rekognition Image, lihat [Menganalisis citra yang dimuat dari sistem file lokal](#).

Jika Anda menggunakan HTTP dan meneruskan bit citra sebagai bagian dari operasi Amazon Rekognition Image, bit citra harus berupa string yang dikodekan base64. Jika Anda menggunakan AWS SDK dan meneruskan bit citra sebagai bagian dari panggilan operasi API, kebutuhan untuk base64-encode byte citra tergantung pada bahasa yang Anda gunakan.

AWS SDK umum berikut secara otomatis menyandikan gambar dengan base64, dan Anda tidak perlu menyandikan byte gambar sebelum memanggil operasi Amazon Rekognition Image API.

- Java

- JavaScript
- Python
- PHP

Jika Anda menggunakan AWS SDK lain dan mendapatkan kesalahan format citra ketika memanggil operasi API Rekognition, coba mengodekan base64 bit citra sebelum meneruskan mereka ke operasi API Rekognition.

Jika Anda menggunakan operasi AWS CLI untuk memanggil Amazon Rekognition Image, meneruskan byte gambar sebagai bagian dari panggilan tidak didukung. Anda harus mengunggah citra terlebih dahulu ke bucket Amazon S3, lalu memanggil operasi yang mereferensikan citra yang diunggah.

Note

Citra tidak perlu dikodekan base64 jika Anda meneruskan sebuah citra yang disimpan dalam `S3Object` bukan bit citra.

Untuk informasi tentang memastikan latensi serendah mungkin untuk operasi Amazon Rekognition Image, lihat [Latensi operasi Amazon Rekognition Image](#).

Memperbaiki orientasi citra

Dalam beberapa operasi API Rekognition, orientasi citra yang dianalisis dikembalikan. Mengetahui orientasi citra merupakan hal penting karena itu memungkinkan Anda untuk mereorientasi citra untuk ditampilkan. Operasi API Rekognition yang menganalisis wajah juga mengembalikan kotak pembatas untuk lokasi wajah dalam citra. Anda dapat menggunakan kotak pembatas untuk menampilkan kotak di sekitar wajah pada citra. Koordinat kotak pembatas yang dikembalikan dipengaruhi oleh orientasi citra dan Anda mungkin perlu menerjemahkan koordinat kotak pembatas untuk menampilkan kotak di sekitar wajah dengan benar. Untuk informasi selengkapnya, lihat [Mendapatkan orientasi citra dan koordinat kotak pembatas](#).

Pengubahan ukuran citra

Selama melakukan analisis, Amazon Rekognition secara internal mengubah ukuran citra menggunakan set rentang yang telah ditetapkan yang paling sesuai dengan model atau algoritme tertentu. Karena itu, Amazon Rekognition mungkin mendeteksi jumlah yang berbeda dari objek,

atau memberikan hasil yang berbeda, tergantung pada resolusi citra input. Misalnya, anggap Anda memiliki dua citra. Citra pertama memiliki resolusi 1024x768 piksel. Citra kedua, versi yang diubah ukurannya dari citra pertama, memiliki resolusi 640x480 piksel. Jika Anda mengirimkan gambar ke [DetectLabels](#), tanggapan dari dua panggilan ke DetectLabels mungkin sedikit berbeda.

Menganalisis citra yang disimpan di bucket Amazon S3

Amazon Rekognition Image dapat menganalisis citra yang disimpan dalam bucket Amazon S3 atau citra yang disediakan sebagai bit citra.

Dalam topik ini, Anda menggunakan operasi [DetectLabels](#) API untuk mendeteksi objek, konsep, dan adegan dalam gambar (JPEG atau PNG) yang disimpan dalam bucket Amazon S3. Anda meneruskan gambar ke operasi Amazon Rekognition Image API dengan menggunakan parameter input `Image`. Di dalamnya `Image`, Anda menentukan properti objek [S3Object](#) untuk mereferensikan gambar yang disimpan dalam bucket S3. Bit citra untuk citra yang disimpan dalam bucket Amazon S3 tidak perlu dikodekan ke base64. Untuk informasi selengkapnya, lihat [Spesifikasi citra](#).

Contoh Permintaan

Dalam contoh permintaan JSON ini untuk `DetectLabels`, citra sumber (`input.jpg`) dimuat dari bucket Amazon S3 bernama `MyBucket`. Ingat bahwa wilayah untuk bucket S3 yang berisi objek S3 harus sesuai dengan wilayah yang Anda gunakan untuk operasi Amazon Rekognition Image.

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

Contoh berikut menggunakan berbagai AWS SDK dan AWS CLI to call `DetectLabels`. Untuk informasi tentang respons operasi `DetectLabels`, lihat [DetectLabels respon](#).

Untuk mendeteksi label dalam citra

1. Jika belum:

- a. Buat atau perbarui pengguna dengan AmazonRekognitionFullAccess dan AmazonS3ReadOnlyAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#). Pastikan bahwa Anda telah memberi pengguna yang memanggil operasi API izin yang tepat untuk akses terprogram, lihat [Memberikan akses programatis](#) petunjuk tentang cara melakukannya.
2. Unggah citra yang berisi satu atau beberapa objek—seperti pohon, rumah, dan perahu—ke bucket S3 Anda. Citra harus dalam format .jpg atau .png.

Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

3. Gunakan contoh berikut untuk memanggil operasi DetectLabels.

Java

Contoh ini menampilkan daftar label yang terdeteksi pada citra input. Ganti nilai-nilai bucket dan photo dengan nama bucket Amazon S3 dan citra yang Anda gunakan di langkah 2.

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

package com.amazonaws.samples;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.DetectLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.Label;
import com.amazonaws.services.rekognition.model.S3Object;
import java.util.List;

public class DetectLabels {

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

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

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

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

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

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

AWS CLI

Contoh ini menampilkan output JSON dari operasi CLI `detect-labels`. Ganti nilai-nilai bucket dan photo dengan nama bucket Amazon S3 dan citra yang Anda gunakan di Langkah 2. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

Jika Anda menggunakan Windows, Anda mungkin perlu menghindari tanda qutasi seperti yang terlihat pada contoh di bawah ini.

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

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

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

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

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "  <bucket> <image>\n\n" +
            "Where:\n" +
```



```
        " bucket - The name of the Amazon S3 bucket that contains the
image (for example, ,ImageBucket)." +
        " image - The name of the image located in the Amazon S3 bucket
(for example, Lake.png). \n\n";

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

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

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

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

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

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

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

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

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

Python

Contoh ini menampilkan label yang terdeteksi dalam citra input. Ganti nilai-nilai bucket dan photo dengan nama bucket Amazon S3 dan citra yang Anda gunakan di Langkah 2. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import boto3

def detect_labels(photo, bucket):

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

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

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

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

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

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

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

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

return len(response['Labels'])
```

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

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

Node.Js

Contoh ini menampilkan informasi tentang label yang terdeteksi dalam gambar.

Ubah nilai photo dengan nama jalur dan file dari sebuah file citra yang berisi satu wajah selebriti atau lebih. Ubah nilai bucket ke nama bucket S3 yang berisi file gambar yang disediakan. Ubah nilai REGION ke nama wilayah yang terkait dengan akun Anda. Ganti nilai profile_name di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

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

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

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

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

// Set params
const params = {For example, to grant
  Image: {
```

```
    S3Object: {
      Bucket: bucket,
      Name: photo
    },
  },
}

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

detect_labels();
```

.NET

Contoh ini menampilkan daftar label yang terdeteksi pada citra input. Ganti nilai-nilai bucket dan photo dengan nama bucket Amazon S3 dan citra yang Anda gunakan di Langkah 2.

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

using System;
```

```
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

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

        AmazonRekognitionClient rekognitionClient = new
AmazonRekognitionClient();

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

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

Ruby

Contoh ini menampilkan daftar label yang terdeteksi pada citra input. Ganti nilai-nilai bucket dan photo dengan nama bucket Amazon S3 dan citra yang Anda gunakan di Langkah 2.

```
# Add to your Gemfile
# gem 'aws-sdk-rekognition'
require 'aws-sdk-rekognition'
credentials = Aws::Credentials.new(
  ENV['AWS_ACCESS_KEY_ID'],
  ENV['AWS_SECRET_ACCESS_KEY']
)
bucket = 'bucket' # the bucket name without s3://
photo = 'photo' # the name of file
client = Aws::Rekognition::Client.new credentials: credentials
attrs = {
  image: {
    s3_object: {
      bucket: bucket,
      name: photo
    },
  },
  max_labels: 10
}
response = client.detect_labels attrs
puts "Detected labels for: #{photo}"
response.labels.each do |label|
  puts "Label:      #{label.name}"
  puts "Confidence: #{label.confidence}"
  puts "Instances:"
  label['instances'].each do |instance|
    box = instance['bounding_box']
    puts "  Bounding box:"
    puts "    Top:      #{box.top}"
    puts "    Left:     #{box.left}"
    puts "    Width:    #{box.width}"
    puts "    Height:   #{box.height}"
    puts "    Confidence: #{instance.confidence}"
  end
  puts "Parents:"
  label.parents.each do |parent|
```

```
    puts "  #{parent.name}"
  end
  puts "-----"
  puts ""
end
```

Contoh respons

Respons dari `DetectLabels` adalah array label yang terdeteksi dalam citra dan tingkat kepercayaan yang mereka deteksi.

Ketika Anda melakukan operasi `DetectLabels` pada citra, Amazon Rekognition mengembalikan output yang mirip dengan contoh respons berikut.

Respons menunjukkan bahwa operasi mendeteksi beberapa label termasuk Orang, Kendaraan, dan Mobil. Setiap label memiliki tingkat kepercayaan yang terkait. Misalnya, algoritme deteksi adalah 98.991432% kepercayaan bahwa citra berisi seseorang.

Respons juga mencakup label leluhur untuk label di array `Parents`. Misalnya, label Otomobil memiliki dua label induk bernama Kendaraan dan Transportasi.

Respons untuk label objek umum mencakup informasi kotak pembatas untuk lokasi label pada citra input. Misalnya, label Orang memiliki array instans yang berisi dua kotak batas. Ini adalah lokasi dari dua orang yang terdeteksi dalam citra.

Bidang `LabelModelVersion` berisi nomor versi model deteksi yang digunakan oleh `DetectLabels`.

Untuk informasi selengkapnya tentang menggunakan operasi `DetectLabels`, lihat [Mendeteksi objek dan konsep](#).

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



```
        "Name": "Transportation"
      }
    ]
  },
  {
    "Name": "Transportation",
    "Confidence": 99.15271759033203,
    "Instances": [],
    "Parents": []
  },
  {
    "Name": "Automobile",
    "Confidence": 99.15271759033203,
    "Instances": [],
    "Parents": [
      {
        "Name": "Vehicle"
      },
      {
        "Name": "Transportation"
      }
    ]
  },
  {
    "Name": "Car",
    "Confidence": 99.15271759033203,
    "Instances": [
      {
        "BoundingBox": {
          "Width": 0.10616336017847061,
          "Height": 0.18528179824352264,
          "Left": 0.0037978808395564556,
          "Top": 0.5039216876029968
        },
        "Confidence": 99.15271759033203
      },
      {
        "BoundingBox": {
          "Width": 0.2429988533258438,
          "Height": 0.21577216684818268,
          "Left": 0.7309805154800415,
          "Top": 0.5251884460449219
        },
        "Confidence": 99.1286392211914
      }
    ]
  }
}
```

```
    },
    {
      "BoundingBox": {
        "Width": 0.14233611524105072,
        "Height": 0.15528248250484467,
        "Left": 0.6494812965393066,
        "Top": 0.5333095788955688
      },
      "Confidence": 98.48368072509766
    },
    {
      "BoundingBox": {
        "Width": 0.11086395382881165,
        "Height": 0.10271988064050674,
        "Left": 0.10355594009160995,
        "Top": 0.5354844927787781
      },
      "Confidence": 96.45606231689453
    },
    {
      "BoundingBox": {
        "Width": 0.06254628300666809,
        "Height": 0.053911514580249786,
        "Left": 0.46083059906959534,
        "Top": 0.5573825240135193
      },
      "Confidence": 93.65448760986328
    },
    {
      "BoundingBox": {
        "Width": 0.10105438530445099,
        "Height": 0.12226245552301407,
        "Left": 0.5743985772132874,
        "Top": 0.534368634223938
      },
      "Confidence": 93.06217193603516
    },
    {
      "BoundingBox": {
        "Width": 0.056389667093753815,
        "Height": 0.17163699865341187,
        "Left": 0.9427769780158997,
        "Top": 0.5235804319381714
      },
    },
```

```
    "Confidence": 92.6864013671875
  },
  {
    "BoundingBox": {
      "Width": 0.06003860384225845,
      "Height": 0.06737709045410156,
      "Left": 0.22409997880458832,
      "Top": 0.5441341400146484
    },
    "Confidence": 90.4227066040039
  },
  {
    "BoundingBox": {
      "Width": 0.02848697081208229,
      "Height": 0.19150497019290924,
      "Left": 0.0,
      "Top": 0.5107086896896362
    },
    "Confidence": 86.65286254882812
  },
  {
    "BoundingBox": {
      "Width": 0.04067881405353546,
      "Height": 0.03428703173995018,
      "Left": 0.316415935754776,
      "Top": 0.5566273927688599
    },
    "Confidence": 85.36471557617188
  },
  {
    "BoundingBox": {
      "Width": 0.043411049991846085,
      "Height": 0.0893595889210701,
      "Left": 0.18293385207653046,
      "Top": 0.5394920110702515
    },
    "Confidence": 82.21705627441406
  },
  {
    "BoundingBox": {
      "Width": 0.031183116137981415,
      "Height": 0.03989990055561066,
      "Left": 0.2853088080883026,
      "Top": 0.5579366683959961
    }
  }
}
```

```
    },
    "Confidence": 81.0157470703125
  },
  {
    "BoundingBox": {
      "Width": 0.031113790348172188,
      "Height": 0.056484755128622055,
      "Left": 0.2580395042896271,
      "Top": 0.5504819750785828
    },
    "Confidence": 56.13441467285156
  },
  {
    "BoundingBox": {
      "Width": 0.08586374670267105,
      "Height": 0.08550430089235306,
      "Left": 0.5128012895584106,
      "Top": 0.5438792705535889
    },
    "Confidence": 52.37760925292969
  }
],
"Parents": [
  {
    "Name": "Vehicle"
  },
  {
    "Name": "Transportation"
  }
]
},
{
  "Name": "Human",
  "Confidence": 98.9914321899414,
  "Instances": [],
  "Parents": []
},
{
  "Name": "Person",
  "Confidence": 98.9914321899414,
  "Instances": [
    {
      "BoundingBox": {
        "Width": 0.19360728561878204,
```

```
        "Height": 0.2742200493812561,  
        "Left": 0.43734854459762573,  
        "Top": 0.35072067379951477  
    },  
    "Confidence": 98.9914321899414  
},  
{  
    "BoundingBox": {  
        "Width": 0.03801717236638069,  
        "Height": 0.06597328186035156,  
        "Left": 0.9155802130699158,  
        "Top": 0.5010883808135986  
    },  
    "Confidence": 85.02790832519531  
}  
],  
"Parents": []  
}  
],  
"LabelModelVersion": "2.0"  
}  
  
}
```

Menganalisis citra yang dimuat dari sistem file lokal

Operasi Amazon Rekognition Image dapat menganalisis citra yang disediakan sebagai bit citra atau citra yang disimpan dalam bucket Amazon S3.

Topik ini memberikan contoh menyediakan bit citra untuk operasi API Amazon Rekognition Image dengan menggunakan file yang dimuat dari sistem file lokal. Anda meneruskan byte gambar ke operasi Amazon Rekognition API dengan [menggunakan](#) parameter input Image. Dalam Image, Anda menentukan properti Bytes untuk meneruskan bit citra yang dikodekan base64.

Bit citra diteruskan ke operasi API Amazon Rekognition dengan menggunakan parameter input Bytes yang harus dikodekan ke base64. AWS SDK yang digunakan contoh ini secara otomatis citranya dikodekan ke base64. Anda tidak perlu mengodekan bit citra sebelum memanggil operasi API Amazon Rekognition. Untuk informasi selengkapnya, lihat [Spesifikasi citra](#).

Dalam contoh permintaan JSON ini untuk DetectLabels, bit citra sumber diteruskan dalam parameter input Bytes.

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

Contoh berikut menggunakan berbagai AWS SDK dan AWS CLI to call `DetectLabels`. Untuk informasi tentang respons operasi `DetectLabels`, lihat [DetectLabels respon](#).

Untuk JavaScript contoh sisi klien, lihat [Menggunakan JavaScript](#)

Untuk mendeteksi label dalam citra lokal

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan `AmazonRekognitionFullAccess` dan `AmazonS3ReadOnlyAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi `DetectLabels`.

Java

Contoh Java berikut menunjukkan cara memuat citra dari sistem file lokal dan mendeteksi label dengan menggunakan operasi AWS SDK [detectLabels](#). Ubah nilai `photo` ke nama jalur dan file dari file citra (format `.jpg` atau `.png`).

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

package aws.example.rekognition.image;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.util.List;
import com.amazonaws.services.rekognition.AmazonRekognition;
```

```
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.AmazonClientException;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.DetectLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.Label;
import com.amazonaws.util.IOUtils;

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

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

        AmazonRekognition rekognitionClient =
        AmazonRekognitionClientBuilder.defaultClient();

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

        try {

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

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

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

```
}  
}
```

Python

Contoh [AWS SDK for Python](#) berikut menunjukkan cara untuk memuat citra dari sistem file lokal dan memanggil operasi [detect_labels](#). Ubah nilai photo ke nama jalur dan file dari file citra (format .jpg atau .png).

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/  
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
import boto3  
  
def detect_labels_local_file(photo):  
  
    client=boto3.client('rekognition')  
  
    with open(photo, 'rb') as image:  
        response = client.detect_labels(Image={'Bytes': image.read()})  
  
    print('Detected labels in ' + photo)  
    for label in response['Labels']:  
        print (label['Name'] + ' : ' + str(label['Confidence']))  
  
    return len(response['Labels'])  
  
def main():  
    photo='photo'  
  
    label_count=detect_labels_local_file(photo)  
    print("Labels detected: " + str(label_count))  
  
if __name__ == "__main__":  
    main()
```


.NET

Contoh berikut menunjukkan cara untuk memuat citra dari sistem file lokal dan mendeteksi label dengan menggunakan operasi `DetectLabels`. Ubah nilai `photo` ke nama jalur dan file dari file citra (format `.jpg` atau `.png`).

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

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

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

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

```

        AmazonRekognitionClient rekognitionClient = new
AmazonRekognitionClient();

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

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

```

PHP

Contoh [AWS SDK for PHP](#) berikut menunjukkan cara memuat gambar dari sistem file lokal dan memanggil operasi [DetectFaces](#) API. Ubah nilai photo ke nama jalur dan file dari file citra (format .jpg atau .png).

```

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

require 'vendor/autoload.php';

use Aws\Rekognition\RekognitionClient;

$options = [
    'region' => 'us-west-2',

```

```
        'version'          => 'latest'
    ];

    $rekognition = new RekognitionClient($options);

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

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

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

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

Ruby

Contoh ini menampilkan daftar label yang terdeteksi pada citra input. Ubah nilai photo ke nama jalur dan file dari file citra (format .jpg atau .png).

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)
```

```
# gem 'aws-sdk-rekognition'
require 'aws-sdk-rekognition'
credentials = Aws::Credentials.new(
  ENV['AWS_ACCESS_KEY_ID'],
  ENV['AWS_SECRET_ACCESS_KEY']
)
client = Aws::Rekognition::Client.new credentials: credentials
photo = 'photo.jpg'
path = File.expand_path(photo) # expand path relative to the current
directory
file = File.read(path)
attrs = {
  image: {
    bytes: file
  },
  max_labels: 10
}
response = client.detect_labels attrs
puts "Detected labels for: #{photo}"
response.labels.each do |label|
  puts "Label:      #{label.name}"
  puts "Confidence: #{label.confidence}"
  puts "Instances:"
  label['instances'].each do |instance|
    box = instance['bounding_box']
    puts "  Bounding box:"
    puts "    Top:      #{box.top}"
    puts "    Left:     #{box.left}"
    puts "    Width:    #{box.width}"
    puts "    Height:   #{box.height}"
    puts "    Confidence: #{instance.confidence}"
  end
  puts "Parents:"
  label.parents.each do |parent|
    puts "  #{parent.name}"
  end
  puts "-----"
  puts ""
end
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class DetectLabels {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

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

            """;

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

```
String sourceImage = args[0];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

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

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

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

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

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

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

Menggunakan JavaScript

Contoh JavaScript halaman web berikut memungkinkan pengguna untuk memilih gambar dan melihat perkiraan usia wajah yang terdeteksi dalam gambar. Perkiraan usia dikembalikan dengan panggilan ke [DetectFaces](#).

Gambar yang dipilih dimuat dengan menggunakan JavaScript `FileReader.readAsDataURL` fungsi, yang base64-menyandikan gambar. Hal ini berguna untuk menampilkan citra pada kanvas HTML. Namun, itu berarti bit citra harus tidak dikodekan sebelum mereka diteruskan ke operasi Amazon Rekognition Image. Contoh ini menunjukkan cara tidak mengodekan bit citra yang dimuat. Jika bit citra yang dikodekan tidak berguna bagi Anda, gunakan `FileReader.readAsArrayBuffer` sebagai gantinya karena citra yang dimuat tidak dikodekan. Ini berarti bahwa operasi Amazon Rekognition Image dapat dipanggil tanpa mengodekan bit citra terlebih dahulu. Sebagai contoh, lihat [Menggunakan readAsArray Buffer](#).

Untuk menjalankan JavaScript contoh

1. Muat kode sumber contoh ke editor.
2. Dapatkan pengenal kolom identitas Amazon Cognito. Untuk informasi selengkapnya, lihat [Mencari pengenal kolom identitas Amazon Cognito](#).
3. Di fungsi `AnonLog` dari kode contoh, ubah `IdentityPoolIdToUse` dan `RegionToUse` dengan nilai-nilai yang Anda catat di langkah 9 dari [Mencari pengenal kolom identitas Amazon Cognito](#).
4. Di fungsi `DetectFaces`, ubah `RegionToUse` dengan nilai yang Anda gunakan di langkah sebelumnya.
5. Simpan kode sumber contoh sebagai file `.html`.
6. Muat file ke peramban Anda.
7. Pilih tombol `Cari...`, dan pilih citra yang berisi satu wajah atau lebih. Tabel yang ditampilkan berisi perkiraan usia untuk setiap wajah yang terdeteksi dalam citra.

Note

Contoh kode berikut menggunakan dua skrip yang tidak lagi menjadi bagian dari Amazon Cognito. Untuk mendapatkan file-file ini, ikuti tautan [aws-cognito-sdkuntuk.min.js](#) [amazon-cognito-identitydan.min.js](#), lalu simpan teks dari masing-masing sebagai file terpisah. `.js`

JavaScript kode contoh

Contoh kode berikut menggunakan JavaScript V2. Untuk contoh di JavaScript V3, lihat [contoh di repositori contoh SDK AWS Dokumentasi](#). [GitHub](#)

```
<!--
Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-
rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)
-->
<!DOCTYPE html>
<html>
<head>
  <script src="aws-cognito-sdk.min.js"></script>
  <script src="amazon-cognito-identity.min.js"></script>
  <script src="https://sdk.amazonaws.com/js/aws-sdk-2.16.0.min.js"></script>
  <meta charset="UTF-8">
  <title>Rekognition</title>
</head>

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

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

  //Calls DetectFaces API and shows estimated ages of detected faces
  function DetectFaces(imageData) {
    AWS.region = "RegionToUse";
    var rekognition = new AWS.Rekognition();
    var params = {
      Image: {
        Bytes: imageData
      },
      Attributes: [
        'ALL',
      ]
    };
    rekognition.detectFaces(params, function (err, data) {
```



```
if (err) console.log(err, err.stack); // an error occurred
else {
  var table = "<table><tr><th>Low</th><th>High</th></tr>";
  // show each face and build out estimated age table
  for (var i = 0; i < data.FaceDetails.length; i++) {
    table += '<tr><td>' + data.FaceDetails[i].AgeRange.Low +
      '</td><td>' + data.FaceDetails[i].AgeRange.High + '</td></tr>';
  }
  table += "</table>";
  document.getElementById("opResult").innerHTML = table;
}
});
}
//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(ua);
    };
})(file);
reader.readAsDataURL(file);
}
//Provides anonymous log on to AWS services
function AnonLog() {

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

```

Menggunakan readAsArray Buffer

Cuplikan kode berikut adalah implementasi alternatif dari ProcessImage fungsi dalam kode sampel, menggunakan JavaScript V2. Ini menggunakan readAsArrayBuffer untuk memuat citra dan memanggil DetectFaces. Karena readAsArrayBuffer tidak mengodekan file yang dimuat ke base64, tidak perlu membatalkan pengodean bit citra sebelum memanggil operasi Amazon Rekognition Image.

```

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

function ProcessImage() {
    AnonLog();
}

```

```

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

// Load base64 encoded image for display
var reader = new FileReader();
reader.onload = (function (theFile) {
    return function (e) {
        //Call Rekognition
        AWS.region = "RegionToUse";
        var rekognition = new AWS.Rekognition();
        var params = {
            Image: {
                Bytes: e.target.result
            },
            Attributes: [
                'ALL',
            ]
        };
        rekognition.detectFaces(params, function (err, data) {
            if (err) console.log(err, err.stack); // an error occurred
            else {
                var table = "<table><tr><th>Low</th><th>High</th></tr>";
                // show each face and build out estimated age table
                for (var i = 0; i < data.FaceDetails.length; i++) {
                    table += '<tr><td>' + data.FaceDetails[i].AgeRange.Low +
                        '</td><td>' + data.FaceDetails[i].AgeRange.High + '</td></tr>';
                }
                table += "</table>";
                document.getElementById("opResult").innerHTML = table;
            }
        });
    })(file);
    reader.readAsArrayBuffer(file);
}

```

Mencari pengenal kolom identitas Amazon Cognito

Agar sederhana, contoh menggunakan kolom identitas Amazon Cognito anonim untuk menyediakan akses tanpa autentikasi ke API Amazon Rekognition Image. Mungkin cocok untuk kebutuhan Anda. Misalnya, Anda dapat menggunakan akses tidak sah untuk memberikan akses gratis, atau uji coba, akses ke situs web Anda sebelum pengguna mendaftar. Untuk memberikan akses yang diautentikasi,

gunakan kolom pengguna Amazon Cognito. Untuk informasi selengkapnya, lihat [Kolam Pengguna Amazon Cognito](#).

Prosedur berikut menunjukkan cara membuat kolom identitas yang memungkinkan akses ke identitas tidak terautentikasi, dan cara untuk mendapatkan pengenalan kolom identitas yang diperlukan dalam kode contoh.

Untuk mendapatkan pengenalan kolom identitas

1. Buka Amazon Cognito [konsol](#).
2. Pilih Buat kolom identitas baru.
3. Untuk Nama kolom identitas*, ketikkan nama untuk kolom identitas Anda.
4. Di Identitas tidak terautentikasi, pilih Aktifkan akses ke identitas yang tidak terautentikasi.
5. Pilih Buat kolom.
6. Pilih Lihat detail, dan perhatikan nama peran untuk identitas yang tidak terautentikasi.
7. Pilih Izinkan.
8. Di Platform, pilih JavaScript.
9. Di Cari Kredensial AWS, perhatikan nilai-nilai `AWS.config.region` dan `IdentityPoolId` yang ditunjukkan di potongan kode.
10. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
11. Di panel navigasi, pilih Peran.
12. Pilih nama peran yang Anda catat di langkah 6.
13. Di tab Izin, pilih Lampirkan Kebijakan.
14. Pilih `AmazonRekognitionReadOnlyAkses`.
15. Pilih Pasang Kebijakan.

Menampilkan kotak pembatas

Operasi Amazon Rekognition Image dapat mengembalikan koordinat kotak pembatas untuk item yang terdeteksi dalam citra. Misalnya, [DetectFaces](#) operasi mengembalikan kotak pembatas ([BoundingBox](#)) untuk setiap wajah yang terdeteksi dalam gambar. Anda dapat menggunakan koordinat kotak pembatas untuk menampilkan kotak di sekitar item yang terdeteksi. Misalnya, citra berikut menunjukkan kotak pembatas yang mengelilingi wajah.



BoundingBox memiliki properti berikut:

- Tinggi – Tinggi kotak pembatas sebagai rasio tinggi citra secara keseluruhan.
- Kiri – Koordinat kiri kotak pembatas sebagai rasio lebar citra secara keseluruhan.
- Atas – Koordinat atas kotak pembatas sebagai rasio tinggi citra secara keseluruhan.
- Lebar – Lebar kotak pembatas sebagai rasio lebar citra secara keseluruhan.

Setiap BoundingBox properti memiliki nilai antara 0 dan 1. Setiap nilai properti adalah rasio lebar citra keseluruhan (Left dan Width) atau tinggi (Height dan Top). Misalnya, jika gambar input 700 x 200 piksel, dan koordinat kiri atas kotak pembatas adalah 350 x 50 piksel, API mengembalikan nilai 0,5 (350/700) dan Left nilai 0,25 (50/200). Top

Diagram berikut menunjukkan rentang citra yang dikover oleh tiap properti kotak pembatas.

Untuk menampilkan kotak pembatas dengan lokasi dan ukuran yang benar, Anda harus mengalikan BoundingBox nilai dengan lebar atau tinggi gambar (tergantung pada nilai yang Anda inginkan) untuk mendapatkan nilai piksel. Anda menggunakan nilai-nilai piksel untuk menampilkan kotak pembatas. Misalnya, dimensi piksel dari citra sebelumnya adalah lebar 608 x tinggi 588. Nilai kotak pembatas untuk wajah adalah:

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

Lokasi kotak pembatas wajah dalam hitungan piksel adalah sebagai berikut:

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

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

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

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

Anda menggunakan nilai-nilai ini untuk menampilkan kotak pembatas di sekitar wajah.

Note

Sebuah citra dapat diorientasikan dalam berbagai cara. Aplikasi Anda mungkin perlu memutar citra untuk menampilkannya dengan orientasi koreksi. Koordinat kotak pembatas dipengaruhi oleh orientasi citra. Anda mungkin perlu menerjemahkan koordinat sebelum Anda dapat menampilkan kotak pembatas di lokasi yang tepat. Untuk informasi selengkapnya, lihat [Mendapatkan orientasi citra dan koordinat kotak pembatas](#).

Contoh berikut menunjukkan cara menampilkan kotak pembatas di sekitar wajah yang terdeteksi dengan memanggil [DetectFaces](#). Contoh tersebut menunjukkan citra yang berorientasi pada 0 derajat. Contoh tersebut juga menunjukkan cara mengunduh citra dari bucket Amazon S3.

Untuk menampilkan kotak pembatas

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan `AmazonRekognitionFullAccess` dan `AmazonS3ReadOnlyAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi `DetectFaces`.

Java

Ubah nilai bucket untuk bucket Amazon S3 yang berisi file citra. Ubah nilai `photo` ke nama file dari file citra (format `.jpg` atau `.png`).

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

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

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

import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.DetectFacesRequest;
import com.amazonaws.services.rekognition.model.DetectFacesResult;
import com.amazonaws.services.rekognition.model.FaceDetail;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
```

```
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3ObjectInputStream;

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

    private static final long serialVersionUID = 1L;

    BufferedImage image;
    static int scale;
    DetectFacesResult result;

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

        result = facesResult;
        image = bufImage;
    }

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

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

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

        // Iterate through faces and display bounding boxes.
        List<FaceDetail> faceDetails = result.getFaceDetails();
        for (FaceDetail face : faceDetails) {

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



```
        Math.round((width * box.getWidth()) / scale),
        Math.round((height * box.getHeight()) / scale);

    }
}

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

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

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

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

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

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

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

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

        System.out.println("Left: " + String.valueOf((int) left));
        System.out.println("Top: " + String.valueOf((int) top));
    }
}
```

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

    }

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

}
}
```

Python

Ubah nilai bucket untuk bucket Amazon S3 yang berisi file citra. Ubah nilai photo ke nama file dari file citra (format .jpg atau .png). Ganti nilai profile_name di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

def show_faces(photo, bucket):

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

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

    stream = io.BytesIO(s3_response['Body'].read())
```

```
image = Image.open(stream)

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

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

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

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

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

    points = (
        (left, top),
        (left + width, top),
        (left + width, top + height),
        (left, top + height),
        (left, top)
    )
    draw.line(points, fill='#00d400', width=2)

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

image.show()
```

```
        return len(response['FaceDetails'])

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

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

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

Perhatikan bahwa `s3` mengacu pada klien Amazon S3 dari AWS SDK Amazon dan `rekClient` mengacu pada klien AWS SDK Amazon Rekognition.

```
//snippet-start:[rekognition.java2.detect_labels.import]
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import software.amazon.awssdk.services.rekognition.model.DetectFacesRequest;
import software.amazon.awssdk.services.rekognition.model.DetectFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceDetail;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
```

```
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
//snippet-end:[rekognition.java2.detect_labels.import]

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

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

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

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

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

        String sourceImage = args[0];
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();
```

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

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

// snippet-start:[rekognition.java2.display_faces.main]
public static void displayAllFaces(S3Client s3,
    RekognitionClient rekClient,
    String sourceImage,
    String bucketName) {

    int height;
    int width;
    byte[] data = getObjectBytes (s3, bucketName, sourceImage);
    InputStream is = new ByteArrayInputStream(data);

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

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

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

        result = rekClient.detectFaces(facesRequest);

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

```

        float left = width * box.left();
        float top = height * box.top();
        System.out.println("Face:");

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

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

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

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

    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

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

```

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

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

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

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

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

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


Mendapatkan orientasi citra dan koordinat kotak pembatas

Aplikasi yang menggunakan Amazon Rekognition Image umumnya perlu menampilkan citra yang terdeteksi oleh operasi Amazon Rekognition Image dan kotak di sekitar wajah yang terdeteksi. Untuk menampilkan gambar dengan benar di aplikasi Anda, Anda perlu mengetahui orientasi gambar. Anda mungkin perlu memperbaiki orientasi ini. Untuk beberapa file .jpg, orientasi citra terdapat dalam metadata format file citra yang Tidak Dapat Diubah (Exif).

Untuk menampilkan kotak di sekitar wajah, Anda memerlukan koordinat untuk kotak pembatas wajah. Jika kotak tidak berorientasi dengan benar, Anda mungkin perlu menyesuaikan koordinat tersebut. Amazon Rekognition Operasi deteksi wajah gambar mengembalikan koordinat kotak pembatas untuk setiap wajah yang terdeteksi, tetapi tidak memperkirakan koordinat untuk file.jpg tanpa metadata Exif.

Contoh berikut menunjukkan cara mendapatkan koordinat kotak pembatas untuk wajah yang terdeteksi dalam gambar.

Gunakan informasi dalam contoh ini untuk memastikan bahwa citra Anda berorientasi dengan benar dan kotak pembatas ditampilkan di lokasi yang benar dalam aplikasi Anda.

Karena kode yang digunakan untuk memutar dan menampilkan citra dan kotak pembatas tergantung pada bahasa dan lingkungan yang Anda gunakan, kami tidak menjelaskan cara menampilkan citra dan kotak pembatas dalam kode Anda, atau cara mendapatkan informasi orientasi dari metadata Exif.

Menemukan orientasi citra

Untuk menampilkan citra dengan benar dalam aplikasi Anda, Anda mungkin perlu memutar itu. Citra berikut berorientasi pada 0 derajat dan ditampilkan dengan benar.



Namun, citra berikut diputar 90 derajat berlawanan arah jarum jam. Untuk menampilkannya dengan benar, Anda perlu menemukan orientasi citra dan menggunakan informasi tersebut dalam kode Anda untuk memutar citra menjadi 0 derajat.



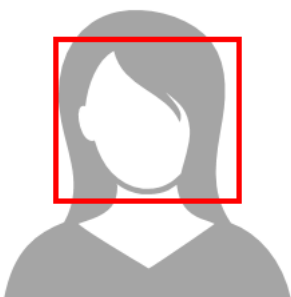
Beberapa citra dalam format .jpg berisi informasi orientasi dalam metadata Exif. Jika tersedia, metadata Exif untuk gambar berisi orientasi. Dalam metadata Exif, Anda dapat menemukan orientasi citra di bidang `orientation`. Meskipun Amazon Rekognition Image mengidentifikasi adanya informasi orientasi citra dalam metadata Exif, namun tidak menyediakan akses untuk itu. Untuk mengakses metadata Exif di citra, gunakan perpustakaan pihak ke tiga atau tulis kode Anda sendiri. Untuk informasi selengkapnya, lihat [Exif Versi 2.32](#).

Ketika Anda mengetahui orientasi citra, Anda dapat menulis kode untuk memutar dan menampilkannya dengan benar.

Menampilkan kotak pembatas

Operasi Amazon Rekognition Image yang menganalisis wajah dalam citra juga mengembalikan koordinat kotak pembatas yang mengelilingi wajah. Untuk informasi lebih lanjut, lihat [BoundingBox](#).

Untuk menampilkan kotak pembatas di sekitar wajah, mirip dengan kotak yang ditunjukkan pada gambar berikut, dalam aplikasi Anda, gunakan koordinat kotak pembatas dalam kode Anda. Koordinat kotak pembatas yang dikembalikan oleh operasi mencerminkan orientasi citra. Jika Anda harus memutar citra untuk menampilkannya dengan benar, Anda mungkin perlu menerjemahkan koordinat kotak pembatas.



Menampilkan kotak pembatas ketika informasi orientasi ada dalam metadata Exif

Jika orientasi citra disertakan dalam metadata Exif, operasi Amazon Rekognition Image melakukan hal berikut:

- Kirimkan nilai nol di bidang koreksi orientasi dalam respons operasi. Untuk memutar citra, gunakan orientasi yang disediakan dalam metadata Exif dalam kode Anda.
- Kembalikan koordinat kotak pembatas yang sudah berorientasi pada 0 derajat. Untuk menunjukkan kotak pembatas pada posisi yang benar, gunakan koordinat yang dikembalikan. Anda tidak perlu menerjemahkannya.

Contoh: Mendeteksi orientasi citra dan koordinat kotak pembatas untuk citra

Contoh berikut menunjukkan cara menggunakan AWS SDK untuk mendapatkan data orientasi gambar Exif dan koordinat kotak pembatas untuk selebriti yang terdeteksi oleh operasi.

RecognizeCelebrities

Note

Support untuk memperkirakan orientasi gambar menggunakan `OrientationCorrection` lapangan telah berhenti per Agustus 2021. Setiap nilai yang dikembalikan untuk bidang ini yang disertakan dalam respons API akan selalu NULL.

Java

Contoh ini memuat gambar dari sistem file lokal, memanggil `RecognizeCelebrities` operasi, menentukan tinggi dan lebar gambar, dan menghitung koordinat kotak pembatas wajah untuk gambar yang diputar. Contoh tidak menunjukkan cara memproses informasi orientasi yang disimpan dalam metadata Exif.

Dalam fungsi main, ganti nilai `photo` dengan nama dan jalur dari citra yang disimpan secara lokal baik dalam format `.png` atau `.jpg`.

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

package com.amazonaws.samples;
import java.awt.image.BufferedImage;
```

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

public class RotateImage {

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

        String photo = "photo.png";

        //Get Rekognition client
        AmazonRekognition amazonRekognition =
            AmazonRekognitionClientBuilder.defaultClient();

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

        try (InputStream inputStream = new FileInputStream(new File(photo))) {
            imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
        }
        catch(Exception e)
        {
            System.out.println("Failed to load file " + photo);
            System.exit(1);
        }
    }
}
```

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

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

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

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

//Call GetCelebrities

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

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

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

        System.out.println();
    }

} catch (AmazonRekognitionException e) {
```

```
        e.printStackTrace();
    }
}

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

    float left = 0;
    float top = 0;

    if(rotation==null){
        System.out.println("No estimated estimated orientation. Check Exif data.");
        return;
    }
    //Calculate face position based on image orientation.
    switch (rotation) {
        case "ROTATE_0":
            left = imageWidth * box.getLeft();
            top = imageHeight * box.getTop();
            break;
        case "ROTATE_90":
            left = imageHeight * (1 - (box.getTop() + box.getHeight()));
            top = imageWidth * box.getLeft();
            break;
        case "ROTATE_180":
            left = imageWidth - (imageWidth * (box.getLeft() + box.getWidth()));
            top = imageHeight * (1 - (box.getTop() + box.getHeight()));
            break;
        case "ROTATE_270":
            left = imageHeight * box.getTop();
            top = imageWidth * (1 - box.getLeft() - box.getWidth());
            break;
        default:
            System.out.println("No estimated orientation information. Check Exif
data.");
            return;
    }

    //Display face location information.
    System.out.println("Left: " + String.valueOf((int) left));
    System.out.println("Top: " + String.valueOf((int) top));
}
```

```
    System.out.println("Face Width: " + String.valueOf((int)(imageWidth *
box.getWidth())));
    System.out.println("Face Height: " + String.valueOf((int)(imageHeight *
box.getHeight())));

}
}
```

Python

Contoh ini menggunakan perpustakaan citra PIL/Pillow untuk mendapatkan lebar dan tinggi citra. Untuk informasi selengkapnya, lihat [Pillow](#). Contoh ini mempertahankan metadata exif yang mungkin Anda perlukan di tempat lain dalam aplikasi Anda.

Dalam fungsi main, ganti nilai photo dengan nama dan jalur dari citra yang disimpan secara lokal baik dalam format .png atau .jpg.

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

import boto3
import io
from PIL import Image

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

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

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

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

```
width, height = image.size

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

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

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

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

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

    # Value of "orientation correction" will always be null
    if 'OrientationCorrection' in response:
        show_bounding_box_positions(height, width, celebrity['Face']
['BoundingBox'])

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

def main():
    photo = 'photo'

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

if __name__ == "__main__":
```



```
main()
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RotateImage {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <sourceImage>

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

                """;
```

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

    String sourceImage = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

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

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

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

        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

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

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

```
        ShowBoundingBoxPositions(height,
                                width,
                                face.boundingBox(),
                                result.orientationCorrectionAsString());
    }

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

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

    // Calculate face position based on the image orientation.
    switch (rotation) {
        case "ROTATE_0" -> {
            left = imageWidth * box.left();
            top = imageHeight * box.top();
        }
        case "ROTATE_90" -> {
            left = imageHeight * (1 - (box.top() + box.height()));
            top = imageWidth * box.left();
        }
        case "ROTATE_180" -> {
            left = imageWidth - (imageWidth * (box.left() + box.width()));
            top = imageHeight * (1 - (box.top() + box.height()));
        }
        case "ROTATE_270" -> {
            left = imageHeight * box.top();
            top = imageWidth * (1 - box.left() - box.width());
        }
        default -> {
            System.out.println("No estimated orientation information. Check Exif
data.");
        }
    }
}
```

```
        return;
    }
}

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

Bekerja dengan analisis video tersimpan

Amazon Rekognition Video merupakan API yang dapat Anda gunakan untuk menganalisis video. Dengan Amazon Rekognition Video, Anda dapat mendeteksi label, wajah, orang, selebriti, dan konten dewasa (sugestif dan eksplisit) dalam video yang disimpan di dalam bucket Amazon Simple Storage Service (Amazon S3). Anda dapat menggunakan Amazon Rekognition Video pada kategori seperti media/hiburan dan keselamatan publik. Sebelumnya, memindai objek atau orang pada video akan memakan waktu berjam-jam untuk melihat perubahan rawan oleh manusia. Amazon Rekognition Video mengotomatisasi pendeteksian item dan ketika deteksi tersebut dijalankan di seluruh video.

Bagian ini mencakup tipe analisis yang dapat dilakukan oleh Amazon Rekognition Video, gambaran umum dari API, dan contoh untuk menggunakan Amazon Rekognition Video.

Topik

- [Tipe analisis](#)
- [Gambaran umum Amazon Rekognition Video API](#)
- [Memanggil operasi Amazon Rekognition Video](#)
- [Mengonfigurasi Amazon Rekognition Video](#)
- [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#)
- [Menganalisis video dengan AWS Command Line Interface](#)
- [Referensi: Notifikasi hasil analisis video](#)
- [Pemecahan masalah Amazon Rekognition Video](#)

Tipe analisis

Anda dapat menggunakan Amazon Rekognition Video untuk menganalisis video untuk informasi berikut:

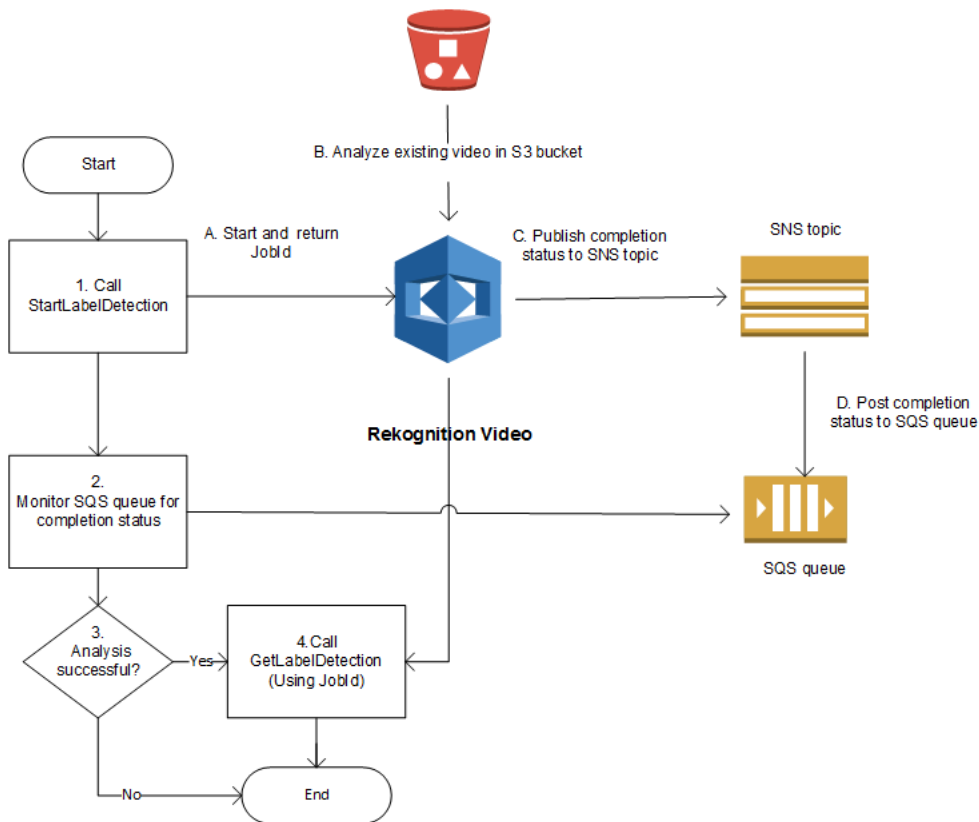
- [Segmen Video](#)
- [Label](#)
- [Konten dewasa yang sugestif dan eksplisit](#)
- [Teks](#)
- [Selebriti](#)
- [Wajah](#)
- [Orang](#)

Untuk informasi selengkapnya, lihat [Bagaimana Amazon Rekognition bekerja](#).

Gambaran umum Amazon Rekognition Video API

Amazon Rekognition Video memproses video yang disimpan di bucket Amazon S3. Pola desain merupakan seperangkat operasi tidak sinkron. Anda memulai analisis video dengan memanggil Start operasi seperti [StartLabelDetection](#). Status penyelesaian untuk permintaan diterbitkan ke topik Amazon Simple Notification Service (Amazon SNS). Untuk mendapatkan status penyelesaian dari topik Amazon SNS, Anda dapat menggunakan antrean Amazon Simple Queue Service (Amazon SQS) atau fungsi AWS Lambda. Setelah Anda memiliki status penyelesaian, Anda memanggil Get operasi, seperti [GetLabelDetection](#), untuk mendapatkan hasil permintaan.

Diagram berikut menunjukkan proses untuk mendeteksi label dalam video yang disimpan dalam bucket Amazon S3. Dalam diagram, antrean Amazon SQS mendapatkan status penyelesaian dari topik Amazon SNS. Atau, Anda dapat menggunakan AWS Lambda fungsi.



Proses ini sama untuk operasi Amazon Rekognition Video lainnya. Tabel berikut mencantumkan opsi Start dan operasi Get untuk masing-masing operasi non-penyimpanan Amazon Rekognition.

Deteksi	Mulai Operasi	Dapatkan Operasi
Segmen video	StartSegmentDetection	GetSegmentDetection
Label	StartLabelDetection	GetLabelDetection
Konten dewasa yang eksplisit atau sugestif	StartContentModeration	GetContentModeration
TEXT	StartTextDetection	GetTextDetection
Selebriti	StartCelebrityRecognition	GetCelebrityRecognition
Wajah	StartFaceDetection	GetFaceDetection
Orang	StartPersonTracking	GetPersonTracking

Untuk operasi Get selain `GetCelebrityRecognition`, Amazon Rekognition Video mengembalikan informasi pelacakan ketika entitas terdeteksi di seluruh input video.

Untuk informasi selengkapnya tentang penggunaan Amazon Rekognition Video, lihat [Memanggil operasi Amazon Rekognition Video](#). Untuk contoh yang melakukan analisis video dengan menggunakan Amazon SQS, lihat [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#). Sebagai AWS CLI contoh, lihat [Menganalisis video dengan AWS Command Line Interface](#).

Format dan penyimpanan video

Operasi Amazon Rekognition dapat menganalisis video yang disimpan di Bucket Amazon S3. Untuk daftar semua batasan operasi analisis video, lihat [Pedoman dan kuota](#).

Video harus dikodekan dengan menggunakan codec H.264. Format file yang didukung adalah MPEG-4 dan MOV.

Codec adalah perangkat lunak atau perangkat keras yang mengompres data agar pengiriman lebih cepat dan dekompresi data yang diterima menjadi bentuk aslinya. Codec H.264 biasanya digunakan untuk perekaman, pengompresan, dan pendistribusian konten video. Format file video dapat berisi satu codec atau lebih. Jika file video Anda dengan format MOV atau MPEG-4 tidak berfungsi dengan Amazon Rekognition Video, periksa apakah codec yang digunakan untuk menyandikan video adalah H.264.

API Video Rekognition Amazon apa pun yang menganalisis data audio hanya mendukung codec audio AAC.

Ukuran file maksimum untuk video yang disimpan adalah 10GB.

Mencari seseorang

Anda dapat menggunakan metadata wajah yang disimpan dalam koleksi untuk mencari orang dalam video. Misalnya, Anda dapat menelusuri video yang diarsipkan untuk orang tertentu atau untuk beberapa orang. Anda menyimpan metadata wajah dari gambar sumber dalam koleksi dengan menggunakan operasi [IndexFaces](#) Anda kemudian dapat menggunakan [StartFaceSearch](#) untuk mulai mencari wajah secara asinkron dalam koleksi. Anda gunakan [GetFaceSearch](#) untuk mendapatkan hasil pencarian. Untuk informasi selengkapnya, lihat [Mencari video yang disimpan untuk wajah](#). Mencari orang merupakan contoh dari operasi Amazon Rekognition berbasis penyimpanan. Untuk informasi selengkapnya, lihat [Operasi API berbasis penyimpanan](#).

Anda juga dapat mencari orang dalam video streaming. Untuk informasi selengkapnya, lihat [Bekerja dengan acara video streaming](#).

Memanggil operasi Amazon Rekognition Video

Amazon Rekognition Video adalah API secara tidak sinkron yang dapat Anda gunakan untuk menganalisis video yang disimpan di bucket Amazon Simple Storage Service (Amazon S3). Anda memulai analisis video dengan memanggil `Start` operasi Video Rekognition Amazon, seperti [StartPersonTracking](#). Amazon Rekognition Video menerbitkan hasil permintaan analisis ke topik Amazon Simple Notification Service (Amazon SNS). Anda dapat menggunakan antrean Amazon Simple Queue Service (Amazon SQS) atau AWS Lambda fungsi untuk mendapatkan status penyelesaian permintaan analisis video dari topik Amazon SNS. Terakhir, Anda mendapatkan hasil permintaan analisis video dengan memanggil operasi `Get` Rekognition Amazon, seperti [GetPersonTracking](#).

Informasi pada bagian ini menggunakan operasi pendeteksi label untuk menunjukkan bagaimana Amazon Rekognition Video mendeteksi label (objek, peristiwa, konsep, dan aktivitas) dalam video yang disimpan dalam bucket Amazon S3. Pendekatan yang sama berfungsi untuk operasi Video Rekognition Amazon lainnya—misalnya, dan [StartFaceDetectionStartPersonTracking](#). Contoh [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#) menunjukkan cara menganalisis video dengan menggunakan antrean Amazon SQS untuk mendapatkan status penyelesaian dari topik Amazon SNS. Amazon SQS juga digunakan sebagai dasar untuk contoh Amazon Rekognition Video lainnya, seperti [Lintasan orang](#). Sebagai AWS CLI contoh, lihat [Menganalisis video dengan AWS Command Line Interface](#).

Topik

- [Memulai analisis video](#)
- [Mendapatkan status penyelesaian permintaan analisis Amazon Rekognition Video](#)
- [Mendapatkan hasil analisis Amazon Rekognition Video](#)

Memulai analisis video

Anda memulai permintaan deteksi label Amazon Rekognition Video dengan menelepon [StartLabelDetection](#). Berikut ini adalah contoh permintaan JSON yang diberikan oleh `StartLabelDetection`.

```
{
  "Video": {
```



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

Parameter input Video menyediakan nama file video dan bucket Amazon S3 untuk mengambil dari kode video tersebut. `NotificationChannel` terdiri Amazon Resource Name (ARN) dari topik Amazon SNS yang memberitahukan Amazon Rekognition Video saat permintaan analisis video selesai. Topik Amazon SNS harus berada di wilayah AWS yang sama dengan titik akhir Amazon Rekognition Video yang Anda panggil. `NotificationChannel` juga berisi ARN untuk peran yang memungkinkan Amazon Rekognition Video untuk mempublikasikan topik Amazon SNS. Anda memberikan izin penerbitan kepada Amazon Rekognition untuk topik Amazon SNS Anda dengan menciptakan peran layanan IAM. Untuk informasi selengkapnya, lihat [Mengonfigurasi Amazon Rekognition Video](#).

Anda juga dapat menentukan parameter input opsional, `JobTag`, yang mengizinkan Anda untuk mengidentifikasi tugas yang memiliki status penyelesaian yang diterbitkan untuk topik Amazon SNS.

Untuk mencegah duplikasi pekerjaan analisis yang tidak disengaja, Anda dapat memberikan token idempotensi, `ClientRequestToken`. Jika Anda memberikan nilai untuk `ClientRequestToken`, operasi `Start` mengembalikan `JobId` yang sama untuk beberapa panggilan yang identik ke operasi awal, seperti `StartLabelDetection`. Token `ClientRequestToken` memiliki masa pakai 7 hari. Setelah 7 hari, Anda bisa menggunakannya kembali. Jika Anda menggunakannya kembali token selama masa token aktif, hal berikut akan terjadi:

- Jika Anda menggunakan kembali token dengan operasi `Start` dan parameter input yang sama, kode `JobId` yang sama akan dikembalikan. Tugas tidak lagi dijalankan dan Amazon Rekognition Video tidak mengirimkan status penyelesaian untuk topik Amazon SNS yang terdaftar.
- Jika Anda menggunakan kembali token dengan operasi `Start` dan perubahan pada input parameter kecil, Anda mendapatkan pengecualian yang ditimbulkan `IdempotentParameterMismatchException` (Kode status HTTP: 400).

- Anda seharusnya tidak menggunakan kembali token dengan operasi `Start` yang berbeda karena Anda akan mendapatkan hasil yang tak terduga dari Amazon Rekognition.

Respons terhadap operasi `StartLabelDetection` adalah pengidentifikasi tugas (`JobId`). Gunakan `JobId` untuk melacak permintaan dan mendapatkan hasil analisis setelah Amazon Rekognition Video telah menerbitkan status penyelesaian ke topik Amazon SNS. Sebagai contoh:

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

Jika Anda memulai terlalu banyak tugas secara bersamaan, hubungi `StartLabelDetection` untuk meningkatkan `LimitExceededException` (Kode status HTTP: 400) hingga jumlah tugas yang berjalan bersamaan di bawah batas layanan Amazon Rekognition.

Jika Anda menemukan bahwa pengecualian `LimitExceededException` dimunculkan dengan runtunan aktivitas, mohon pertimbangkan untuk menggunakan antrian Amazon SQS untuk mengelola permintaan yang masuk. Hubungi AWS dukungan jika Anda menemukan bahwa jumlah rata-rata permintaan bersamaan tidak dapat dikelola oleh antrian Amazon SQS dan Anda masih menerima pengecualian. `LimitExceededException`

Mendapatkan status penyelesaian permintaan analisis Amazon Rekognition Video

Amazon Rekognition Video mengirimkan pemberitahuan penyelesaian analisis untuk topik Amazon SNS yang terdaftar. Pemberitahuan tersebut mencakup pengidentifikasi tugas dan status penyelesaian operasi dalam string JSON. Permintaan analisis video yang berhasil memiliki status `SUCCEEDED`. Contohnya, hasil berikut menunjukkan berhasilnya pengolahan tugas pendeteksi label.

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

Untuk informasi selengkapnya, lihat [Referensi: Notifikasi hasil analisis video](#).

Untuk mendapatkan status informasi yang dipublikasikan ke topik Amazon SNS oleh Amazon Rekognition Video, gunakan salah satu opsi berikut:

- **AWS Lambda** — Anda dapat berlangganan fungsi AWS Lambda yang Anda tulis untuk topik Amazon SNS. Fungsi ini terpanggil ketika Amazon Rekognition memberitahu topik Amazon SNS bahwa permintaannya telah selesai. Gunakan fungsi Lambda jika Anda ingin kode sisi server untuk memproses hasil permintaan analisis video. Misalnya, Anda mungkin ingin menggunakan kode sisi server untuk menganotasi video atau membuat laporan tentang konten video sebelum mengembalikan informasi ke aplikasi klien. Kami juga merekomendasikan pemrosesan sisi server untuk video besar karena Amazon Rekognition API dapat menghasilkan data dalam jumlah besar.
- **Amazon Simple Queue Service** — Anda dapat berlangganan antrean Amazon SQS ke topik Amazon SNS. Anda kemudian dapat melakukan polling antrean Amazon SQS untuk mengambil status penyelesaian yang diterbitkan oleh Amazon Rekognition saat permintaan analisis video selesai. Untuk informasi selengkapnya, lihat [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#). Gunakan antrean Amazon SQS jika Anda ingin memanggil operasi Amazon Rekognition Video hanya dari aplikasi klien.

Important

Kami tidak merekomendasikan untuk mendapatkan status penyelesaian permintaan dengan berulang kali memanggil operasi Get Amazon Rekognition Video. Hal ini karena Amazon Rekognition Video membatasi operasi Get jika permintaan terlalu banyak dibuat. Jika Anda memproses beberapa video secara bersamaan, akan lebih mudah dan lebih efisien untuk memantau satu antrean SQS untuk notifikasi penyelesaian daripada membuat polling Amazon Rekognition Video untuk status setiap video secara individual.

Mendapatkan hasil analisis Amazon Rekognition Video

Untuk mendapatkan hasil permintaan analisis video, pertama pastikan bahwa status penyelesaian yang diambil dari topik Amazon SNS adalah SUCCEEDED. Kemudian panggil `GetLabelDetection`, yang memberikan nilai `JobId` yang dikembalikan dari `StartLabelDetection`. Permintaan JSON serupa dengan contoh berikut:

```
{
  "JobId": "270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1784ca90fc180e3",
  "MaxResults": 10,
```

```
"SortBy": "TIMESTAMP"
}
```

JobId adalah pengidentifikasi untuk operasi analisis video. Mengingat analisis video dapat menghasilkan data dalam jumlah besar, gunakan `MaxResults` untuk menentukan hasil jumlah maksimum agar dapat kembali dalam satu operasi `Get`. Nilai default untuk `MaxResults` adalah 1000. Jika Anda menentukan nilai yang lebih besar dari 1000, maksimum hasil dikembalikan adalah 1000. Jika operasi tidak mengembalikan seluruh set hasil, token pemberian nomor halaman untuk halaman berikutnya dikembalikan dalam respons operasi. Jika Anda memiliki token pemberian nomor halaman dari permintaan `Get` sebelumnya, gunakan token tersebut dengan `NextToken` untuk mendapatkan halaman hasil berikutnya.

Note

Amazon Rekognition mempertahankan hasil operasi analisis video selama 7 hari. Anda tidak akan dapat mengambil hasil analisis setelah periode ini.

Respons JSON dari operasi `GetLabelDetection` serupa dengan hal berikut:

```
{
  "Labels": [
    {
      "Timestamp": 0,
      "Label": {
        "Instances": [],
        "Confidence": 60.51791763305664,
        "Parents": [],
        "Name": "Electronics"
      }
    },
    {
      "Timestamp": 0,
      "Label": {
        "Instances": [],
        "Confidence": 99.53411102294922,
        "Parents": [],
        "Name": "Human"
      }
    },
  ]
}
```

```

    "Timestamp": 0,
    "Label": {
      "Instances": [
        {
          "BoundingBox": {
            "Width": 0.11109819263219833,
            "Top": 0.08098889887332916,
            "Left": 0.8881205320358276,
            "Height": 0.9073750972747803
          },
          "Confidence": 99.5831298828125
        },
        {
          "BoundingBox": {
            "Width": 0.1268676072359085,
            "Top": 0.14018426835536957,
            "Left": 0.0003282368124928324,
            "Height": 0.7993982434272766
          },
          "Confidence": 99.46029663085938
        }
      ],
      "Confidence": 99.53411102294922,
      "Parents": [],
      "Name": "Person"
    }
  },
  .
  .
  .
  {
    "Timestamp": 166,
    "Label": {
      "Instances": [],
      "Confidence": 73.6471176147461,
      "Parents": [
        {
          "Name": "Clothing"
        }
      ],
      "Name": "Sleeve"
    }
  }
}

```

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

GetContentModerationOperasi GetLabelDetection dan memungkinkan Anda untuk mengurutkan hasil analisis berdasarkan stempel waktu atau dengan nama label. Anda juga dapat mengumpulkan hasil berdasarkan segmen video atau stempel waktu.

Anda dapat mengurutkan hasil berdasarkan waktu deteksi (milidetik dari awal video) atau menurut abjad sesuai entitas yang terdeteksi (objek, wajah, selebriti, label moderasi, atau orang). Untuk mengurutkan berdasarkan waktu, tetapkan nilai dari `SortBy` input parameter ke `TIMESTAMP`. Jika `SortBy` tidak ditentukan, perilaku default menjadi pengurutan berdasarkan waktu. Contoh sebelumnya diurutkan berdasarkan waktu. Untuk mengurutkan berdasarkan entitas, gunakan parameter input `SortBy` dengan nilai yang sesuai untuk operasi yang Anda jalankan. Misalnya, untuk mengurutkan berdasarkan label yang terdeteksi dalam panggilan ke `GetLabelDetection`, gunakan nilai `NAME`.

Untuk menggabungkan hasil dengan stempel waktu, atur nilai parameter ke `AggregateBy` `TIMESTAMPS` Untuk agregat menurut segmen video, atur nilai `AggregateBy` ke `SEGMENTS`. `SEGMENTS` mode agregasi akan menggabungkan label dari waktu ke waktu, sementara `TIMESTAMPS` memberikan stempel waktu label terdeteksi pada, menggunakan 2 FPS sampling dan per frame output (Catatan: Tingkat pengambilan sampel saat ini dapat berubah, asumsi tidak boleh dibuat tentang laju pengambilan sampel saat ini). Jika tidak ada nilai yang ditentukan, metode agregasi default adalah `TIMESTAMPS`.

Mengonfigurasi Amazon Rekognition Video

Untuk menggunakan Amazon Rekognition Video API dengan video yang disimpan, Anda harus mengonfigurasi pengguna dan peran layanan IAM untuk mengakses topik Amazon SNS Anda. Anda juga harus berlangganan antrean Amazon SQS ke topik Amazon SNS.

Note

Jika Anda menggunakan petunjuk ini untuk mengatur contoh [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#), Anda tidak perlu melakukan langkah 3, 4, 5, dan 6. Contoh tersebut mencakup kode untuk membuat dan mengonfigurasi topik Amazon SNS dan antrean Amazon SQS.

Contoh dalam bagian ini membuat topik Amazon SNS baru dengan menggunakan petunjuk yang memberikan akses Amazon Rekognition Video ke beberapa topik. Jika Anda ingin menggunakan topik Amazon SNS yang sudah ada, gunakan [Memberikan akses ke topik Amazon SNS yang sudah ada](#) untuk langkah 3.

Untuk mengonfigurasi Amazon Rekognition Video

1. Siapkan AWS akun untuk mengakses Amazon Rekognition Video. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
2. Instal dan konfigurasi AWS SDK yang diperlukan. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
3. Untuk menjalankan contoh kode dalam panduan pengembang ini, pastikan bahwa pengguna yang Anda pilih memiliki akses terprogram. Untuk informasi selengkapnya, lihat [Memberikan akses programatis](#).

Pengguna Anda juga memerlukan setidaknya izin berikut:

- AmazonSQS FullAccess
- AmazonRekognitionFullAccess
- AmazonS3 FullAccess
- AmazonSNS FullAccess

Jika Anda menggunakan Pusat Identitas IAM untuk mengautentikasi, tambahkan izin ke set izin untuk peran Anda, jika tidak, tambahkan izin ke peran IAM Anda.

4. [Buat topik Amazon SNS](#) dengan menggunakan [konsol Amazon SNS](#). Tambahkan nama topik dengan. AmazonRekognition Perhatikan topik Amazon Resource Name (ARN). Pastikan topik berada di wilayah yang sama dengan titik akhir AWS yang Anda gunakan.

5. [Buat antrean standar Amazon SQS](#) dengan menggunakan [Konsol Amazon SQS](#). Perhatikan antrean ARN.
6. [Berlangganan antrean ke topik](#) yang Anda buat di langkah 3.
7. [Memberikan izin untuk topik Amazon SNS untuk mengirim pesan ke antrean Amazon SQS](#).
8. Buat peran layanan IAM untuk memberikan akses Amazon Rekognition Video ke topik Amazon SNS Anda. Catat Amazon Resource Name (ARN) dari peran layanan tersebut. Untuk informasi selengkapnya, lihat [Memberikan akses ke beberapa topik Amazon SNS](#).
9. Untuk memastikan akun Anda aman, Anda akan ingin membatasi ruang lingkup akses Rekognition hanya ke sumber daya yang Anda gunakan. Ini dapat dilakukan dengan melampirkan kebijakan Trust ke peran layanan IAM Anda. Untuk informasi tentang cara melakukannya, lihat [Pencegahan wakil bingung lintas layanan](#).
10. [Tambahkan kebijakan inline berikut](#) ke pengguna yang Anda buat di langkah 1:

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

Berikan kebijakan inline sebuah nama yang Anda pilih.

11. Jika Anda menggunakan AWS Key Management Service kunci yang dikelola pelanggan untuk mengenkripsi video di bucket Amazon S3, tambahkan izin ke kunci yang memungkinkan peran layanan yang Anda buat di langkah 7 untuk mendekripsi video. Minimal peran layanan membutuhkan izin `kms:GenerateDataKey` dan `kms:Decrypt` tindakan. Sebagai contoh:

```
{
  "Sid": "Decrypt only",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/user from step 1"
  },
}
```



```
"Action": [  
    "kms:Decrypt",  
    "kms:GenerateDataKey"  
],  
"Resource": "*" ]
```

Untuk informasi selengkapnya, lihat [Bucket Amazon S3 saya memiliki enkripsi default menggunakan kunci AWS KMS khusus. Bagaimana cara mengizinkan pengguna mengunduh dan mengunggah ke bucket?](#) dan [Melindungi Data Menggunakan Enkripsi Sisi Server dengan kunci KMS yang Disimpan di AWS Key Management Service \(SSE-KMS\)](#).

12. Anda sekarang dapat menjalankan contoh di [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#) dan [Menganalisis video dengan AWS Command Line Interface](#).

Memberikan akses ke beberapa topik Amazon SNS

Anda menggunakan peran layanan IAM untuk memberikan akses Amazon Rekognition Video ke topik Amazon SNS yang Anda buat. IAM menyediakan kasus penggunaan Rekognition untuk membuat peran layanan Amazon Rekognition Video.

Anda dapat memberikan akses Video Rekognition Amazon ke beberapa topik Amazon SNS dengan menggunakan kebijakan `AmazonRekognitionServiceRole` izin dan mendahului nama topik dengan —misalnya, `AmazonRekognitionAmazonRekognitionMyTopicName`

Untuk memberikan akses Amazon Rekognition Video ke beberapa topik Amazon SNS

1. [Buat peran layanan IAM](#). Gunakan informasi berikut ini untuk membuat peran layanan IAM:
 1. Pilih Rekognition untuk nama layanan.
 2. Pilih Rekognition untuk kasus penggunaan peran layanan. Anda akan melihat kebijakan `AmazonRekognitionServiceRoleizin` yang tercantum. `AmazonRekognitionServiceRole` memberikan Amazon Rekognition Video akses ke topik Amazon SNS yang diawali dengan `AmazonRekognition`
 3. Berikan peran layanan sebuah nama yang Anda pilih.
2. Catat peran layanan ARN. Anda memerlukannya untuk memulai operasi analisis video.

Memberikan akses ke topik Amazon SNS yang sudah ada

Anda dapat membuat kebijakan izin yang memungkinkan akses Amazon Rekognition Video ke topik Amazon SNS yang sudah ada.

Untuk memberikan akses Amazon Rekognition Video ke topik Amazon SNS yang sudah ada

1. [Buat kebijakan izin baru dengan editor kebijakan IAM JSON](#), dan gunakan kebijakan berikut. Ganti `topicarn` dengan Amazon Resource Name (ARN) topik Amazon SNS yang diinginkan.

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

2. [Buat peran layanan IAM](#), atau perbarui peran layanan IAM yang ada. Gunakan informasi berikut ini untuk membuat peran layanan IAM:
 1. Pilih Rekognition untuk nama layanan.
 2. Pilih Rekognition untuk kasus penggunaan peran layanan.
 3. Lampirkan kebijakan izin yang Anda buat pada langkah 1.
3. Catat peran layanan ARN. Anda memerlukannya untuk memulai operasi analisis video.

Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python (SDK)

Prosedur ini menunjukkan kepada Anda cara untuk mendeteksi label dalam video dengan menggunakan operasi pendeteksi label Amazon Rekognition Video, video yang disimpan dalam bucket Amazon S3, dan topik Amazon SNS. Prosedur ini juga menunjukkan cara menggunakan antrean Amazon SQS untuk mendapatkan status penyelesaian dari topik Amazon SNS. Untuk informasi selengkapnya, lihat [Memanggil operasi Amazon Rekognition Video](#). Anda tidak dibatasi

untuk menggunakan antrian Amazon SQS. Misalnya, Anda dapat menggunakan AWS Lambda fungsi untuk mendapatkan status penyelesaian. Untuk informasi selengkapnya, lihat [Memanggil fungsi Lambda menggunakan notifikasi Amazon SNS](#).

Kode sampel dalam prosedur ini menunjukkan kepada Anda cara melakukan hal berikut:

1. Buat topik Amazon SNS.
2. Buat antrian Amazon SQS.
3. Berikan izin Amazon Rekognition Video untuk memublikasikan status penyelesaian operasi analisis video untuk topik Amazon SNS.
4. Berlangganan antrian Amazon SQS ke topik Amazon SNS.
5. Mulai permintaan analisis video dengan menelepon [StartLabelDetection](#).
6. Dapatkan status penyelesaian dari antrian Amazon SQS. Contoh dari melacak pengidentifikasi tugas (JobId) yang dikembalikan di `StartLabelDetection` dan hanya mendapatkan hasil untuk mencocokkan pengidentifikasi tugas yang dibaca dari status penyelesaian. Hal ini merupakan pertimbangan penting jika aplikasi lain menggunakan antrian dan topik yang sama. Untuk kesederhanaan, contoh penghapusan tugas yang tidak cocok. Pertimbangkan untuk menambahkan penghapusan tugas ke antrian surat mati Amazon SQS untuk penyelidikan lebih lanjut.
7. Dapatkan dan tampilkan hasil analisis video dengan menelepon [GetLabelDetection](#).

Prasyarat

Contoh kode untuk prosedur ini disediakan di Javaa dan Python. Anda harus menginstal AWS SDK yang sesuai. Untuk informasi selengkapnya, lihat [Memulai dengan Amazon Rekognition](#). Akun AWS yang Anda gunakan harus memiliki izin akses ke Amazon Rekognition API. Untuk informasi selengkapnya, lihat [Tindakan yang Ditetapkan oleh Amazon Rekognition](#).

Mendeteksi label dalam video

1. Konfigurasi akses pengguna ke Amazon Rekognition Video dan konfigurasi akses Amazon Rekognition Video ke Amazon SNS. Untuk informasi selengkapnya, lihat [Mengonfigurasi Amazon Rekognition Video](#). Anda tidak perlu melakukan langkah 3, 4, 5, dan 6 karena contoh kode membuat dan mengonfigurasi topik Amazon SNS dan antrian Amazon SQS.

2. Unggah file video format MOV atau MPEG-4 ke Bucket Amazon S3. Untuk pengujian, unggah video yang panjangnya tidak lebih dari 30 detik.

Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

3. Gunakan contoh kode berikut untuk mendeteksi label dalam video.

Java

Pada fungsi main:

- Ganti `roleArn` dengan ARN dari peran layanan IAM yang Anda buat di langkah 7 dari [Untuk mengonfigurasi Amazon Rekognition Video](#).
- Ganti nilai-nilai bucket dan video dengan nama file bucket dan video yang Anda tentukan pada langkah 2.

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

package com.amazonaws.samples;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Condition;
import com.amazonaws.auth.policy.Principal;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.SQSActions;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.CelebrityDetail;
import com.amazonaws.services.rekognition.model.CelebrityRecognition;
import com.amazonaws.services.rekognition.model.CelebrityRecognitionSortBy;
import com.amazonaws.services.rekognition.model.ContentModerationDetection;
import com.amazonaws.services.rekognition.model.ContentModerationSortBy;
import com.amazonaws.services.rekognition.model.Face;
import com.amazonaws.services.rekognition.model.FaceDetection;
import com.amazonaws.services.rekognition.model.FaceMatch;
import com.amazonaws.services.rekognition.model.FaceSearchSortBy;
import com.amazonaws.services.rekognition.model.GetCelebrityRecognitionRequest;
```

```
import com.amazonaws.services.rekognition.model.GetCelebrityRecognitionResult;
import com.amazonaws.services.rekognition.model.GetContentModerationRequest;
import com.amazonaws.services.rekognition.model.GetContentModerationResult;
import com.amazonaws.services.rekognition.model.GetFaceDetectionRequest;
import com.amazonaws.services.rekognition.model.GetFaceDetectionResult;
import com.amazonaws.services.rekognition.model.GetFaceSearchRequest;
import com.amazonaws.services.rekognition.model.GetFaceSearchResult;
import com.amazonaws.services.rekognition.model.GetLabelDetectionRequest;
import com.amazonaws.services.rekognition.model.GetLabelDetectionResult;
import com.amazonaws.services.rekognition.model.GetPersonTrackingRequest;
import com.amazonaws.services.rekognition.model.GetPersonTrackingResult;
import com.amazonaws.services.rekognition.model.Instance;
import com.amazonaws.services.rekognition.model.Label;
import com.amazonaws.services.rekognition.model.LabelDetection;
import com.amazonaws.services.rekognition.model.LabelDetectionSortBy;
import com.amazonaws.services.rekognition.model.NotificationChannel;
import com.amazonaws.services.rekognition.model.Parent;
import com.amazonaws.services.rekognition.model.PersonDetection;
import com.amazonaws.services.rekognition.model.PersonMatch;
import com.amazonaws.services.rekognition.model.PersonTrackingSortBy;
import com.amazonaws.services.rekognition.model.S3Object;
import
    com.amazonaws.services.rekognition.model.StartCelebrityRecognitionRequest;
import com.amazonaws.services.rekognition.model.StartCelebrityRecognitionResult;
import com.amazonaws.services.rekognition.model.StartContentModerationRequest;
import com.amazonaws.services.rekognition.model.StartContentModerationResult;
import com.amazonaws.services.rekognition.model.StartFaceDetectionRequest;
import com.amazonaws.services.rekognition.model.StartFaceDetectionResult;
import com.amazonaws.services.rekognition.model.StartFaceSearchRequest;
import com.amazonaws.services.rekognition.model.StartFaceSearchResult;
import com.amazonaws.services.rekognition.model.StartLabelDetectionRequest;
import com.amazonaws.services.rekognition.model.StartLabelDetectionResult;
import com.amazonaws.services.rekognition.model.StartPersonTrackingRequest;
import com.amazonaws.services.rekognition.model.StartPersonTrackingResult;
import com.amazonaws.services.rekognition.model.Video;
import com.amazonaws.services.rekognition.model.VideoMetadata;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.CreateTopicResult;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.Message;
```

```
import com.amazonaws.services.sqs.model.QueueAttributeName;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import java.util.*;

public class VideoDetect {

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

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

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

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

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

        CreateTopicandQueue();

        //=====

        StartLabelDetection(bucket, video);

        if (GetSQSMessagesSuccess()==true)
```

```
        GetLabelDetectionResults();

        //=====

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

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

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

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

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

                    // Get status and job id from notification.
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
```

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

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

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

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

        System.out.println("Finished processing video");
        return success;
    }

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



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

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

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

}

private static void GetLabelDetectionResults() throws Exception{

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

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

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

        labelDetectionResult = rek.getLabelDetection(labelDetectionRequest);

        VideoMetadata videoMetaData=labelDetectionResult.getVideoMetadata();
```

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

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

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

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

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

    List<Parent> parents = label.getParents();
    if (parents.isEmpty()) {
        System.out.println("        None");
    } else {
        for (Parent parent : parents) {
            System.out.println("            " + parent.getName());
        }
    }
    System.out.println();
}
```

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

}

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

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

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

    // Authorize queue
    Policy policy = new Policy().withStatements(
        new Statement(Effect.Allow)
            .withPrincipals(Principal.AllUsers)
            .withActions(SQSActions.SendMessage)
            .withResources(new Resource(sqsQueueArn))
            .withConditions(new
Condition().withType("ArnEquals").withConditionKey("aws:SourceArn").withValues(snsTopic
));

    Map queueAttributes = new HashMap();
```

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

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

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

Python

Pada fungsi main:

- Ganti `roleArn` dengan ARN dari peran layanan IAM yang Anda buat di langkah 7 dari [Untuk mengonfigurasi Amazon Rekognition Video](#).
- Ganti nilai-nilai `bucket` dan `video` dengan nama file bucket dan video yang Anda tentukan pada langkah 2.
- Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.
- Anda juga dapat memasukkan kriteria filtrasi dalam parameter pengaturan. Misalnya, Anda dapat menggunakan `LabelsInclusionFilter` atau `LabelsExclusionFilter` bersama daftar nilai yang diinginkan. Dalam kode di bawah ini, Anda dapat menghapus komentar `Settings` bagian `Features and` dan memberikan nilai Anda sendiri untuk membatasi hasil yang dikembalikan hanya pada label yang Anda minati.

- Dalam panggilan ke `getLabelDetection`, Anda dapat memberikan nilai untuk `AggregateBy` argumen `SortBy` dan. Untuk mengurutkan berdasarkan waktu, tetapkan nilai dari `SortBy` input parameter ke `TIMESTAMP`. Untuk mengurutkan berdasarkan entitas, gunakan parameter input `SortBy` dengan nilai yang sesuai untuk operasi yang Anda jalankan. Untuk menggabungkan hasil dengan stempel waktu, atur nilai parameter ke `AggregateBy` `TIMESTAMPS` Untuk agregat berdasarkan segmen video, gunakan `SEGMENTS`.

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

import boto3
import json
import sys
import time

class VideoDetect:

    jobId = ''

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

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

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

    def GetSQSMessageSuccess(self):
```

```
jobFound = False
succeeded = False

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

    if sqsResponse:

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

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

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

```
ReceiptHandle=message['ReceiptHandle'])

    return succeeded

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

NotificationChannel={'RoleArn': self.roleArn,

'SNSTopicArn': self.snsTopicArn},

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

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

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

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

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

```

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

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

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

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

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

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

def CreateTopicandQueue(self):

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

```



```
# Create SNS topic

snsTopicName = "AmazonRekognitionExample" + millis

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

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

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

sqsQueueArn = attribs['QueueArn']

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

# Authorize SNS to write SQS queue
policy = """{{
"Version":"2012-10-17",
"Statement":[
  {{
    "Sid":"MyPolicy",
    "Effect":"Allow",
    "Principal" : {{"AWS" : "*"}},
    "Action":"SQS:SendMessage",
    "Resource": "{}",
    "Condition":{{
      "ArnEquals":{{
        "aws:SourceArn": "{}"
      }}
    }}
  }}
]]
}"""
}}""".format(sqsQueueArn, self.snsTopicArn)
```

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

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

def main():

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

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

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

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

    analyzer.DeleteTopicandQueue()

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

Node.Js

Dalam kode contoh berikut:

- Ganti nilai REGION dengan nama wilayah operasi akun Anda.
- Ganti nilainya bucket dengan nama bucket Amazon S3 yang berisi file video Anda.
- Ganti nilai videoName dengan nama file video di bucket Amazon S3 Anda.

- Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.
- Ganti `roleArn` dengan ARN dari peran layanan IAM yang Anda buat di langkah 7 dari [Untuk mengonfigurasi Amazon Rekognition Video](#).

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

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

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

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

// Set the parameters
```

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

const createTopicandQueue = async () => {
  try {
    // Create SNS topic
    const topicResponse = await snsClient.send(new
    CreateTopicCommand(snsTopicParams));
    const topicArn = topicResponse.TopicArn
    console.log("Success", topicResponse);
    // Create SQS Queue
    const sqsResponse = await sqsClient.send(new
    CreateQueueCommand(sqsParams));
    console.log("Success", sqsResponse);
    const sqsQueueCommand = await sqsClient.send(new
    GetQueueUrlCommand({QueueName: sqsQueueName}))
    const sqsQueueUrl = sqsQueueCommand.QueueUrl
    const attriBsResponse = await sqsClient.send(new
    GetQueueAttributesCommand({QueueUrl: sqsQueueUrl, AttributeNames:
    ['QueueArn']}))
    const attriBs = attriBsResponse.Attributes
    console.log(attriBs)
    const queueArn = attriBs.QueueArn
    // subscribe SQS queue to SNS topic
    const subscribed = await snsClient.send(new SubscribeCommand({TopicArn:
    topicArn, Protocol:'sqs', Endpoint: queueArn}))
    const policy = {
      Version: "2012-10-17",
      Statement: [
        {
          Sid: "MyPolicy",
          Effect: "Allow",
          Principal: {AWS: "*"},
          Action: "SQS:SendMessage",
          Resource: queueArn,
          Condition: {
            ArnEquals: {
              'aws:SourceArn': topicArn
            }
          }
        }
      ]
    }
  }
}
```

```
    }
  }
]
};

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

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

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

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

  // Begin retrieving label detection results
  while (finished == false){
    var response = await rekClient.send(new GetLabelDetectionCommand({JobId:
    startJobId, MaxResults: maxResults,
    NextToken: paginationToken, SortBy:'TIMESTAMP'}))
```

```
// Log metadata
console.log(`Codec: ${response.VideoMetadata.Codec}`)
console.log(`Duration: ${response.VideoMetadata.DurationMillis}`)
console.log(`Format: ${response.VideoMetadata.Format}`)
console.log(`Frame Rate: ${response.VideoMetadata.FrameRate}`)
console.log()
// For every detected label, log label, confidence, bounding box, and
timestamp
response.Labels.forEach(labelDetection => {
  var label = labelDetection.Label
  console.log(`Timestamp: ${labelDetection.Timestamp}`)
  console.log(`Label: ${label.Name}`)
  console.log(`Confidence: ${label.Confidence}`)
  console.log("Instances:")
  label.Instances.forEach(instance =>{
    console.log(`Confidence: ${instance.Confidence}`)
    console.log("Bounding Box:")
    console.log(`Top: ${instance.Confidence}`)
    console.log(`Left: ${instance.Confidence}`)
    console.log(`Width: ${instance.Confidence}`)
    console.log(`Height: ${instance.Confidence}`)
    console.log()
  })
  console.log()
  // Log parent if found
  console.log("  Parents:")
  label.Parents.forEach(parent =>{
    console.log(`    ${parent.Name}`)
  })
  console.log()
  // Search for pagination token, if found, set variable to next token
  if (String(response).includes("NextToken")){
    paginationToken = response.NextToken

  }else{
    finished = true
  }

})
}
}

// Checks for status of job completion
const getSQSMessageSuccess = async(sqsQueueUrl, startJobId) => {
```

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

// Once job found, log Job ID and return true if status is succeeded
for (var message of sqsReceivedResponse.Messages){
  console.log("Retrieved messages:")
  var notification = JSON.parse(message.Body)
  var rekMessage = JSON.parse(notification.Message)
  var messageJobId = rekMessage.JobId
  if (String(rekMessage.JobId).includes(String(startJobId))){
    console.log('Matching job found:')
    console.log(rekMessage.JobId)
    jobFound = true
    console.log(rekMessage.Status)
    if (String(rekMessage.Status).includes(String("SUCCEEDED"))){
      succeeded = true
      console.log("Job processing succeeded.")
    }
  }
}
```

```
        var sqsDeleteMessage = await sqsClient.send(new
DeleteMessageCommand({QueueUrl:sqsQueueUrl,
ReceiptHandle:message.ReceiptHandle}));
    }
    }else{
        console.log("Provided Job ID did not match returned ID.")
        var sqsDeleteMessage = await sqsClient.send(new
DeleteMessageCommand({QueueUrl:sqsQueueUrl,
ReceiptHandle:message.ReceiptHandle}));
    }
}
}
return succeeded
} catch(err) {
    console.log("Error", err);
}
};

// Start label detection job, sent status notification, check for success
status
// Retrieve results if status is "SUCEEDED", delete notification queue and
topic
const runLabelDetectionAndGetResults = async () => {
    try {
        const sqsAndTopic = await createTopicandQueue();
        const startLabelDetectionRes = await startLabelDetection(roleArn,
sqsAndTopic[1]);
        const getSqsMessageStatus = await getSqsMessageSuccess(sqsAndTopic[0],
startLabelDetectionRes)
        console.log(getSqsMessageSuccess)
        if (getSqsMessageSuccess){
            console.log("Retrieving results:")
            const results = await getLabelDetectionResults(startLabelDetectionRes)
        }
        const deleteQueue = await sqsClient.send(new DeleteQueueCommand({QueueUrl:
sqsAndTopic[0]}));
        const deleteTopic = await snsClient.send(new DeleteTopicCommand({TopicArn:
sqsAndTopic[1]}));
        console.log("Successfully deleted.")
    } catch (err) {
        console.log("Error", err);
    }
};
```



```
runLabelDetectionAndGetResults()
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;
//snippet-end:[rekognition.java2.recognize_video_detect.import]

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

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class VideoDetect {

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

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

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

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
            .build();

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

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

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

// snippet-start:[rekognition.java2.recognize_video_detect.main]
public static void startLabels(RekognitionClient rekClient,
                               NotificationChannel channel,
                               String bucket,
                               String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vid0b = Video.builder()
            .s3Object(s3obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
        StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vid0b)
            .minConfidence(50F)
            .build();

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

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

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

        GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
        status = result.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            ans = false;
        else
            System.out.println(yy + " status is: "+status);

        Thread.sleep(1000);
        yy++;
    }

    System.out.println(startJobId + " status is: "+status);

} catch (RekognitionException | InterruptedException e) {
    e.getMessage();
    System.exit(1);
}
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {

    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message: messages) {
                String notification = message.body();

                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
```

```
        JsonNode jsonMessageTree = mapper.readTree(notification);
        JsonNode messageBodyText = jsonMessageTree.get("Message");
        ObjectMapper operationResultMapper = new ObjectMapper();
        JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
        JsonNode operationJobId = jsonResultTree.get("JobId");
        JsonNode operationStatus = jsonResultTree.get("Status");
        System.out.println("Job found in JSON is " + operationJobId);

        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
            .queueUrl(queueUrl)
            .build();

        String jobId = operationJobId.textValue();
        if (startJobId.compareTo(jobId)==0) {
            System.out.println("Job id: " + operationJobId );
            System.out.println("Status : " +
operationStatus.toString());

            if (operationStatus.asText().equals("SUCCEEDED"))
                GetResultsLabels(rekClient);
            else
                System.out.println("Video analysis failed");

            sqs.deleteMessage(deleteMessageRequest);
        }

        else{
            System.out.println("Job received was not job " +
startJobId);

            sqs.deleteMessage(deleteMessageRequest);
        }
    }
}

} catch (RekognitionException e) {
    e.getMessage();
    System.exit(1);
} catch (JsonMappingException e) {
    e.printStackTrace();
} catch (JsonProcessingException e) {
    e.printStackTrace();
}
}
```

```
}

// Gets the job results by calling GetLabelDetection
private static void GetResultsLabels(RekognitionClient rekClient) {

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

    try {
        do {
            if (labelDetectionResult !=null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest=
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
                .maxResults(maxResults)
                .nextToken(paginationToken)
                .build();

            labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
            VideoMetadata videoMetaData=labelDetectionResult.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());

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

                System.out.println("    Label:" + label.name());
                System.out.println("    Confidence:" +
detectedLabel.label().confidence().toString());

                List<Instance> instances = label.instances();
                System.out.println("    Instances of " + label.name());
            }
        }
    }
}
```

```
        if (instances.isEmpty()) {
            System.out.println("        " + "None");
        } else {
            for (Instance instance : instances) {
                System.out.println("        Confidence: " +
instance.confidence().toString());
                System.out.println("        Bounding box: " +
instance.boundingBox().toString());
            }
        }
        System.out.println("    Parent labels for " + label.name() +
":");

        List<Parent> parents = label.parents();

        if (parents.isEmpty()) {
            System.out.println("        None");
        } else {
            for (Parent parent : parents) {
                System.out.println("    " + parent.name());
            }
        }
        System.out.println();
    }
} while (labelDetectionResult !=null &&
labelDetectionResult.nextToken() != null);

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

4. Bangun dan jalankan kode tersebut. Operasi mungkin membutuhkan waktu beberapa saat untuk menyelesaikan. Setelah selesai, daftar label yang terdeteksi di video akan ditampilkan. Untuk informasi selengkapnya, lihat [Mendeteksi label dalam video](#).

Menganalisis video dengan AWS Command Line Interface

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk memanggil operasi Amazon Rekognition Video. Pola desainnya sama dengan menggunakan Amazon Rekognition Video

API dengan AWS SDK atau AWS SDK for Java lainnya. Untuk informasi selengkapnya, lihat [Gambaran umum Amazon Rekognition Video API](#). Prosedur berikut menunjukkan cara menggunakan AWS CLI untuk mendeteksi label dalam video.

Anda mulai mendeteksi label dalam video dengan menelepon `start-label-detection`. Setelah Amazon Rekognition selesai menganalisis video, status penyelesaian dikirim ke topik Amazon SNS yang ditentukan dalam `--notification-channel` parameter `start-label-detection`. Anda dapat mendapatkan status penyelesaian dengan berlangganan antrean Amazon Simple Queue Service (Amazon SQS) ke topik Amazon SNS. Anda kemudian melakukan polling [Penerimaan-pesan](#) untuk mendapatkan status penyelesaian dari antrean Amazon SQS.

Saat menelepon `StartLabelDetection`, Anda dapat memfilter hasil dengan memberikan argumen filtrasi ke `LabelsInclusionFilter` dan/atau `LabelsExclusionFilter` argumen. Untuk informasi selengkapnya, lihat [Mendeteksi label dalam video](#).

Notifikasi status penyelesaian merupakan struktur JSON dalam tanggapan `receive-message`. Anda perlu untuk mengekstraksi JSON dari respons. Untuk informasi tentang status penyelesaian JSON, lihat [Referensi: Notifikasi hasil analisis video](#). Jika nilai dari bidang `Status` dari status penyelesaian JSON adalah `SUCCEEDED`, Anda bisa mendapatkan hasil permintaan analisis video dengan memanggil `get-label-detection`. Saat memanggil `GetLabelDetection`, Anda dapat mengurutkan dan menggabungkan hasil yang dikembalikan menggunakan `AggregateBy` argumen `SortBy` dan.

Prosedur berikut tidak melibatkan kode untuk polling antrean Amazon SQS. Juga, mereka tidak melibatkan kode untuk mengurai JSON yang kembali dari antrean Amazon SQS. Sebagai contoh di Java, lihat [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#).

Prasyarat

Untuk menjalankan prosedur ini, Anda harus AWS CLI menginstal. Untuk informasi selengkapnya, lihat [Memulai dengan Amazon Rekognition](#). Akun AWS yang Anda gunakan harus memiliki izin akses ke Amazon Rekognition API. Untuk informasi selengkapnya, [Tindakan yang Ditetapkan oleh Amazon Rekognition](#).

Untuk mengonfigurasi Amazon Rekognition Video dan mengunggah video

1. Konfigurasi akses pengguna ke Amazon Rekognition Video dan konfigurasi akses Amazon Rekognition Video ke Amazon SNS. Untuk informasi selengkapnya, lihat [Mengonfigurasi Amazon Rekognition Video](#).

2. Unggah file video format MOV atau MPEG-4 ke bucket S3 Anda. Saat pengembangan dan pengujian, kami menyarankan untuk menggunakan video pendek dengan durasi tidak lebih dari 30 detik.

Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Mendeteksi label dalam video

1. Jalankan AWS CLI perintah berikut untuk mulai mendeteksi label dalam video.

```
aws rekognition start-label-detection --video '{"S3Object":{"Bucket":"bucket-name","Name":"video-name"}}' \
  --notification-channel '{"SNSTopicArn":"TopicARN","RoleArn":"RoleARN"}' \
  --region region-name \
  --features GENERAL_LABELS \
  --profile profile-name \
  --settings '{"GeneralLabels":{"LabelInclusionFilters":["Car"]}]'
```

Perbarui nilai berikut:

- Ubah `bucketname` dan `videofile` ke nama bucket Amazon S3 dan nama file yang Anda tentukan pada langkah 2.
- Ubah `us-east-1` ke wilayah AWS yang Anda gunakan.
- Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.
- Ubah `TopicARN` ke ARN dari topik Amazon SNS yang Anda buat pada langkah 3 [Mengonfigurasi Amazon Rekognition Video](#).
- Perubah `RoleARN` ke ARN dari peran layanan IAM yang Anda buat di langkah 7 [Mengonfigurasi Amazon Rekognition Video](#).
- Jika diperlukan, Anda dapat menentukan `endpoint-url`. AWS CLI harus secara otomatis menentukan titik akhir URL yang tepat berdasarkan wilayah yang disediakan. Namun, jika Anda menggunakan titik akhir [dari VPC privat Anda](#), Anda mungkin perlu menentukan `endpoint-url`. Parameter daftar sumber daya sintaksis [Titik Akhir Layanan AWS](#) untuk menentukan titik akhir url, nama dan kode untuk masing-masing wilayah.

- Anda juga dapat memasukkan kriteria filtrasi dalam parameter pengaturan. Misalnya, Anda dapat menggunakan `LabelsInclusionFilter` atau `LabelsExclusionFilter` bersama daftar nilai yang diinginkan.

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda alih-alih tanda kutip tunggal dan hindari tanda kutip ganda bagian dalam dengan garis miring terbalik (yaitu `\`) untuk mengatasi kesalahan parser yang mungkin Anda temui. Sebagai contoh, lihat di bawah:

```
aws rekognition start-label-detection --video "{\"S3Object\":{\"Bucket\":\"bucket-name\",\"Name\":\"video-name\"}}" --notification-channel "{\"SNSTopicArn\":\"TopicARN\",\"RoleArn\":\"RoleARN\"}" \
--region us-east-1 --features GENERAL_LABELS --settings "{\"GeneralLabels\":{\"LabelInclusionFilters\":[\"Car\"]}}" --profile profile-name
```

2. Perhatikan nilai dari `JobId` dalam respons. Respons tersebut serupa dengan contoh JSON berikut ini.

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

3. Tulis kode untuk membuat polling antrean Amazon SQS untuk status penyelesaian JSON (dengan menggunakan [menerima-pesan](#)).
4. Tulis kode untuk mengekstraksi bidang `Status` dari status penyelesaian JSON.
5. Jika nilai `Status` adalah `SUCCEEDED`, jalankan AWS CLI perintah berikut untuk menunjukkan hasil deteksi label.

```
aws rekognition get-label-detection --job-id JobId \
--region us-east-1 --sort-by TIMESTAMP aggregate-by TIMESTAMPS
```

Perbarui nilai berikut:

- Ubah `JobId` untuk mencocokkan pengidentifikasi tugas yang Anda catat di langkah 2.
- Ubah `Endpoint` dan `us-east-1` ke titik akhir AWS dan wilayah yang Anda gunakan.

Hasil terlihat serupa dengan contoh JSON berikut:

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

Referensi: Notifikasi hasil analisis video

Amazon Rekognition menerbitkan hasil permintaan analisis Amazon Rekognition Video, termasuk status penyelesaian, ke topik Amazon Simple Notification Service (Amazon SNS). Untuk mendapatkan notifikasi dari topik Amazon SNS, gunakan antrian Amazon Simple Queue Service atau fungsi AWS Lambda. Untuk informasi selengkapnya, lihat [the section called “Memanggil operasi Amazon Rekognition Video”](#). Sebagai contoh, lihat [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#).

Muatan berada dalam format JSON berikut:

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

Nama	Penjelasan
JobId	Pengidentifikasi tugas. Cocokkan pengenalan pekerjaan yang dikembalikan dari Start operasi, seperti StartPersonTracking .
Status	Status tugas. Nilai yang valid adalah BERHASIL, GAGAL, atau ERROR.
API	Operasi Amazon Rekognition Video yang digunakan untuk menganalisis video input.
JobTag	Pengidentifikasi untuk tugas. Anda menentukan JobTag dalam panggilan untuk memulai operasi, seperti StartLabelDetection .
Stempel Waktu	Stempel waktu Unix untuk ketika tugas selesai.
Video	Detail tentang video yang telah diproses. Mencakup nama file dan bucket Amazon S3 tempat file disimpan.

Berikut ini adalah contoh notifikasi yang sukses dikirim ke topik Amazon SNS.

```
{
```

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

Pemecahan masalah Amazon Rekognition Video

Berikut ini mencakup informasi pemecahan masalah untuk bekerja dengan Amazon Rekognition Video dan video yang disimpan.

Saya tidak pernah menerima status penyelesaian yang dikirim ke topik Amazon SNS

Amazon Rekognition Video menerbitkan informasi status untuk topik Amazon SNS ketika analisis video selesai. Biasanya, Anda mendapatkan pesan status penyelesaian dengan berlangganan topik bersama antrean Amazon SQS atau fungsi Lambda. Untuk membantu penyelidikan Anda, silakan berlangganan topik Amazon SNS melalui email sehingga Anda menerima pesan yang dikirim ke topik Amazon SNS Anda di kotak masuk email Anda. Untuk informasi selengkapnya, lihat [Berlangganan topik Amazon SNS](#).

Jika Anda tidak menerima pesan dalam aplikasi Anda, pertimbangkan hal berikut:

- Verifikasi bahwa analisis telah selesai. Periksa nilai `JobStatus` dalam operasi respons `GetLabelDetection`, misalnya). Jika nilainya adalah `IN_PROGRESS`, analisis tersebut belum selesai, dan status penyelesaian belum dipublikasikan ke topik Amazon SNS.
- Verifikasi bahwa Anda memiliki peran layanan IAM yang memberikan izin Amazon Rekognition Video untuk memublikasikan ke topik Amazon SNS Anda. Untuk informasi selengkapnya, lihat [Mengonfigurasi Amazon Rekognition Video](#).
- Konfirmasikan bahwa peran layanan IAM yang Anda gunakan dapat dipublikasikan ke topik Amazon SNS dengan menggunakan kredensial peran dan bahwa izin peran layanan Anda dicakup dengan aman ke sumber daya yang Anda gunakan. Lakukan langkah-langkah berikut:
 - Dapatkan pengguna Amazon Resource Name (ARN):

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

- Tambahkan ARN pengguna ke hubungan kepercayaan peran. Untuk informasi selengkapnya, lihat [Memodifikasi peran](#). Contoh kebijakan trust berikut menentukan kredensi peran pengguna dan membatasi izin peran layanan hanya ke sumber daya yang Anda gunakan (untuk informasi selengkapnya tentang membatasi cakupan izin peran layanan secara aman, lihat): [Pencegahan wakil bingung lintas layanan](#)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "rekognition.amazonaws.com",
        "AWS": "arn:User ARN"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "Account ID"
        },
        "StringLike": {
          "aws:SourceArn":
            "arn:aws:rekognition:region:111122223333:streamprocessor/*"
        }
      }
    }
  ]
}
```

- Asumsikan peran: `aws sts assume-role --role-arn arn:Role ARN --role-session-name SessionName --profile RekognitionUser`
- Terbitkan topik Amazon SNS: `aws sns publish --topic-arn arn:Topic ARN --message "Hello World!" --region us-east-1 --profile RekognitionUser`

Jika perintah AWS CLI berfungsi, Anda menerima pesan (di kotak masuk email Anda, jika Anda telah berlangganan topik melalui email). Jika Anda tidak menerima pesan:

- Periksa apakah Anda telah mengonfigurasi Amazon Rekognition Video. Untuk informasi selengkapnya, lihat [Mengonfigurasi Amazon Rekognition Video](#).
- Periksa tips lain untuk pertanyaan terkait pemecahan masalah ini.
- Periksa apakah Anda menggunakan topik Amazon SNS:

- Jika Anda menggunakan peran layanan IAM untuk memberikan akses Amazon Rekognition Video ke satu topik Amazon SNS, periksa apakah Anda telah memberikan izin untuk topik Amazon SNS yang benar. Untuk informasi selengkapnya, lihat [Memberikan akses ke topik Amazon SNS yang sudah ada](#).
- Jika Anda menggunakan peran layanan IAM untuk memberikan akses Video Rekognition Amazon ke beberapa topik SNS, verifikasi bahwa Anda menggunakan topik yang benar dan nama topik sudah ditambahkan. Untuk informasi selengkapnya, lihat [Memberikan akses ke beberapa topik Amazon SNS](#).
- Jika Anda menggunakan suatu AWS Lambda fungsi, konfirmasi bahwa fungsi Lambda Anda berlangganan topik Amazon SNS yang benar. Untuk informasi selengkapnya, lihat fungsi [Fanout ke Lambda](#).
- Jika Anda berlangganan antrean Amazon SQS untuk topik Amazon SNS Anda, Konfirmasikan bahwa topik Amazon SNS Anda memiliki izin untuk mengirim pesan ke antrean Amazon SQS. Untuk informasi selengkapnya, lihat [Berikan izin pada topik Amazon SNS untuk mengirim pesan ke antrean Amazon SQS](#).

Saya memerlukan bantuan tambahan untuk memecahkan masalah topik Amazon SNS

Anda dapat menggunakan AWS X-Ray Amazon SNS untuk melacak dan menganalisis pesan yang melakukan perjalanan melalui aplikasi Anda. Untuk informasi selengkapnya, lihat [Amazon SNS dan AWS X-Ray](#)

[Untuk bantuan tambahan, Anda dapat memposting pertanyaan Anda ke forum Rekognition Amazon atau mempertimbangkan untuk mendaftar AWS untuk dukungan teknis.](#)

Bekerja dengan acara video streaming

Anda dapat menggunakan Amazon Rekognition Video untuk mendeteksi dan mengenali wajah atau mendeteksi objek dalam streaming video. Amazon Rekognition Video menggunakan Amazon Kinesis Video Streams untuk menerima dan memproses video streaming. Anda membuat prosesor aliran dengan parameter yang menunjukkan apa yang Anda inginkan untuk dideteksi oleh prosesor stream dari aliran video. Rekognition mengirimkan hasil deteksi label dari streaming acara video sebagai pemberitahuan Amazon SNS dan Amazon S3. Keluaran rekognition menghadapi hasil pencarian ke aliran data Kinesis.

Pemroses aliran pencarian wajah digunakan `FaceSearchSettings` untuk mencari wajah dari koleksi. Untuk informasi selengkapnya tentang cara menerapkan prosesor aliran pencarian wajah

untuk menganalisis wajah dalam streaming video, lihat [the section called “Mencari wajah dalam koleksi dalam streaming video”](#).

Penggunaan prosesor aliran deteksi label `ConnectedHomeSettings` untuk mencari orang, paket, dan hewan peliharaan dalam streaming acara video. Untuk informasi selengkapnya tentang cara menerapkan pemroses aliran deteksi label, lihat [the section called “Mendeteksi label dalam streaming acara video”](#).

Ikhtisar operasi prosesor aliran Video Rekognition Amazon

Anda mulai menganalisis video streaming dengan memulai pemroses aliran dan video streaming Amazon Rekognition Video ke Amazon Rekognition Video. Pemroses aliran Amazon Rekognition Video memungkinkan Anda memulai, menghentikan, dan mengelola pemroses aliran. Anda membuat pemroses aliran dengan memanggil [CreateStreamProcessor](#). Parameter permintaan untuk membuat prosesor aliran pencarian wajah mencakup Amazon Resource Names (ARN) untuk aliran video Kinesis, aliran data Kinesis, dan pengenal untuk koleksi yang digunakan untuk mengenali wajah dalam video streaming. Parameter permintaan untuk membuat prosesor stream pemantauan keamanan mencakup Amazon Resource Names (ARN) untuk aliran video Kinesis dan topik Amazon SNS, jenis objek yang ingin Anda deteksi dalam aliran video, dan informasi untuk bucket Amazon S3 untuk hasil keluaran. Anda juga menyertakan nama yang Anda tentukan untuk prosesor aliran.

Anda mulai memproses video dengan memanggil operasi [StartStreamProcessor](#). Untuk mendapatkan informasi status untuk pemroses aliran, panggil [DescribeStreamProcessor](#). Operasi lain yang dapat Anda hubungi adalah [TagResource](#) untuk menandai prosesor stream dan [DeleteStreamProcessor](#) untuk menghapus prosesor aliran. Jika Anda menggunakan prosesor aliran pencarian wajah, Anda juga dapat menggunakan [StopStreamProcessor](#) untuk menghentikan prosesor aliran. Untuk mendapatkan daftar pemroses aliran di akun Anda, panggil [ListStreamProcessors](#).

Setelah pemroses aliran mulai berjalan, Anda melakukan streaming video ke Amazon Rekognition Video melalui aliran video Kinesis yang Anda tentukan di `CreateStreamProcessor`. Anda dapat menggunakan Kinesis Video Streams SDK [PutMedia](#) operasi untuk mengirimkan video ke aliran video Kinesis. Sebagai contoh, lihat [PutMediaAPI Contoh](#).

Untuk informasi tentang cara aplikasi Anda dapat menggunakan hasil analisis Amazon Rekognition Video dari prosesor aliran pencarian wajah, lihat [Membaca hasil analisis video streaming](#).

Menandai pemroses aliran Amazon Rekognition Video

Anda dapat mengidentifikasi, mengatur, mencari, dan memfilter pemroses aliran Amazon Rekognition dengan menggunakan tanda. Setiap tanda adalah label yang terdiri dari kunci dan nilai yang ditentukan pengguna.

Topik

- [Menambahkan tanda ke pemroses aliran baru](#)
- [Menambahkan tanda ke pemroses aliran yang sudah ada](#)
- [Mencantumkan tanda dalam pemroses aliran](#)
- [Menghapus tanda dari pemroses aliran](#)

Menambahkan tanda ke pemroses aliran baru

Anda dapat menambahkan tanda ke pemroses aliran saat Anda membuatnya menggunakan operasi `CreateStreamProcessor`. Tentukan satu tanda atau lebih dalam parameter input array `Tags`. Berikut ini adalah contoh JSON untuk permintaan `CreateStreamProcessor` dengan tanda.

```
{
  "Name": "streamProcessorForCam",
  "Input": {
    "KinesisVideoStream": {
      "Arn": "arn:aws:kinesisvideo:us-east-1:nnnnnnnnnnnn:stream/
inputVideo"
    }
  },
  "Output": {
    "KinesisDataStream": {
      "Arn": "arn:aws:kinesis:us-east-1:nnnnnnnnnnnn:stream/outputData"
    }
  },
  "RoleArn": "arn:aws:iam::nnnnnnnnnnnn:role/roleWithKinesisPermission",
  "Settings": {
    "FaceSearch": {
      "CollectionId": "collection-with-100-faces",
      "FaceMatchThreshold": 85.5
    },
    "Tags": {
      "Dept": "Engineering",
      "Name": "Ana Silva Carolina",

```

```
    "Role": "Developer"
  }
}
```

Menambahkan tanda ke pemroses aliran yang sudah ada

Untuk menambahkan satu tanda atau lebih ke pemroses aliran yang ada, gunakan operasi `TagResource`. Tentukan Amazon Resource Name (ARN) pemroses aliran (`ResourceArn`) dan tanda (Tags) yang ingin Anda tambahkan. Contoh berikut menunjukkan cara menambahkan dua tanda.

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

Note

Jika Anda tidak tahu nama Sumber Daya Amazon aliran pemroses, Anda dapat menggunakan operasi `DescribeStreamProcessor`.

Mencantumkan tanda dalam pemroses aliran

Untuk daftar tanda yang terlampir pada pemroses aliran, gunakan operasi `ListTagsForResource` dan tentukan ARN dari pemroses aliran (`ResourceArn`). Respons adalah peta kunci tanda dan nilai-nilai yang terlampir pada pemroses aliran yang ditentukan.

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

Output menampilkan daftar tanda yang terlampir pada pemroses aliran:

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

```
}
```

Menghapus tanda dari pemroses aliran

Untuk membuang satu tanda atau lebih dari sebuah pemroses aliran, gunakan operasi `UntagResource`. Tentukan ARN dari model (`ResourceArn`) dan tombol tanda (`Tag-Keys`) yang ingin Anda hapus.

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

Atau, Anda dapat menentukan tanda-kunci dalam format ini:

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

Penanganan kesalahan

Bagian ini menjelaskan kesalahan waktu aktif dan cara menanganinya. Ini juga menjelaskan pesan kesalahan dan kode yang khusus untuk Amazon Rekognition.

Topik

- [Komponen kesalahan](#)
- [Pesan dan kode kesalahan](#)
- [Penanganan kesalahan dalam aplikasi Anda](#)

Komponen kesalahan

Ketika program Anda mengirimkan permintaan, Amazon Rekognition mencoba untuk memprosesnya. Jika permintaan berhasil, Amazon Rekognition mengembalikan kode status sukses HTTP (200 OK), bersama dengan hasil dari operasi yang diminta.

Jika permintaan tidak berhasil, Amazon Rekognition mengembalikan kesalahan. Setiap kesalahan memiliki tiga komponen:

- Kode status HTTP (seperti 400).
- Nama pengecualian (seperti `InvalidS3ObjectException`).

- Pesan kesalahan (seperti Unable to get object metadata from S3. Check object key, region and/or access permissions.).

SDK AWS menangani kesalahan penyebaran untuk aplikasi Anda, sehingga Anda dapat mengambil tindakan yang tepat. Misalnya, dalam program Java, Anda bisa menulis logika try-catch untuk menangani ResourceNotFoundException.

Jika Anda tidak menggunakan SDK AWS, Anda perlu mengurai isi dari respons tingkat rendah dari Amazon Rekognition. Berikut ini adalah contoh responsnya:

```
HTTP/1.1 400 Bad Request
Content-Type: application/x-amz-json-1.1
Date: Sat, 25 May 2019 00:28:25 GMT
x-amzn-RequestId: 03507c9b-7e84-11e9-9ad1-854a4567eb71
Content-Length: 222
Connection: keep-alive

{"__type":"InvalidS3ObjectException","Code":"InvalidS3ObjectException","Logref":"5022229e-7e48-
to get object metadata from S3. Check object key, region and/or access permissions."}
```

Pesan dan kode kesalahan

Berikut ini adalah daftar pengecualian yang dikembalikan oleh Amazon Rekognition, dikelompokkan berdasarkan kode status HTTP. Jika Boleh diulang? adalah Ya, Anda dapat mengirimkan permintaan yang sama lagi. Jika Boleh diulang? adalah Tidak, Anda harus memperbaiki masalah di sisi klien sebelum mengirimkan permintaan baru.

Kode status HTTP 400

Kode status 400 HTTP menunjukkan adanya masalah dengan permintaan Anda. Beberapa contoh masalah adalah kegagalan autentikasi, parameter yang diperlukan hilang, atau melebihi operasi yang ditetapkan throughput. Anda harus memperbaiki masalah dalam aplikasi Anda sebelum mengirimkan permintaan lagi.

AccessDeniedException

Pesan:Terjadi galat (AccessDeniedException) Saat memanggil <Operation>operasi: Pengguna: <User ARN>tidak berwenang untuk melakukan: <Operation>pada sumber daya:<Resource ARN>.

Anda tidak berwenang untuk melakukan tindakan. Gunakan Amazon Resource Name (ARN) dari pengguna resmi atau IAM role untuk melakukan operasi.

Boleh diulang? Tidak

GroupFacesInProgressException

Pesan:Gagal menjadwalkanGroupFacespekerjaan. Sudah ada grup yang menghadapi tugas untuk koleksi ini.

Coba lagi operasi setelah tugas yang ada selesai.

Boleh diulang? Tidak

IdempotentParameterMismatchException

Pesan:YangClientRequestToken: <Token>Anda telah disediakan sudah digunakan.

Parameter input ClientRequestToken digunakan kembali dengan suatu operasi, tapi setidaknya salah satu parameter input lainnya berbeda dari panggilan ke operasi sebelumnya.

Boleh diulang? Tidak

ImageTooLargeException

Pesan: Ukuran citra terlalu besar.

Ukuran citra input melebihi batas yang diizinkan. Jika Anda memanggil [DetectProtectiveEquipment](#), ukuran citra atau resolusi melebihi batas yang diizinkan. Untuk informasi selengkapnya, lihat [Pedoman dan kuota dalam Amazon Rekognition](#).

Boleh diulang? Tidak

InvalidImageFormatException

Pesan: Permintaan memiliki format citra yang tidak sah.

Tidak mendukung format citra yang disediakan. Gunakan format citra yang didukung (.JPEG dan .PNG). Untuk informasi selengkapnya, lihat [Pedoman dan kuota dalam Amazon Rekognition](#).

Boleh diulang? Tidak

InvalidPaginationTokenException

Pesan

- Token Tidak Valid
- Token Pagination Tidak Valid

Token pemberian nomor halaman dalam permintaan tidak valid. Token mungkin telah kedaluwarsa.

Boleh diulang? Tidak

InvalidParameterException

Pesan: Permintaan memiliki parameter yang tidak valid.

Parameter input melanggar batasan. Validasi parameter Anda sebelum memanggil operasi API lagi.

Boleh diulang? Tidak

InvalidObjectException

Pesan:

- Permintaan memiliki objek S3 tidak valid.
- Tidak bisa mendapatkan metadata objek dari S3. Periksa kunci objek, wilayah dan/atau izin akses.

Amazon Rekognition tidak dapat mengakses objek S3 yang ditentukan dalam permintaan. Untuk informasi selengkapnya, lihat [Konfigurasi Akses ke S3: Akses Pengelolaan AWS S3](#). Untuk informasi pemecahan masalah, lihat [Memecahkan Masalah Amazon S3](#).

Boleh diulang? Tidak

LimitExceededException

Pesan:

- <Current Limit>Batas prosesor aliran terlampaui untuk akun, batas -.
- <Number of Open Jobs>terbuka Pekerjaan untuk pengguna <User ARN>batas maksimum: <Maximum Limit>

Melebihi batas layanan Amazon Rekognition. Misalnya, jika Anda memulai terlalu banyak tugas untuk Amazon Rekognition Video secara bersamaan, panggilan untuk memulai operasi, seperti `StartLabelDetection`, menaikkan pengecualian `LimitExceededException` (kode status HTTP: 400) hingga jumlah tugas yang berjalan bersama berada di bawah batas layanan Amazon Rekognition.

Boleh diulang? Tidak

`ProvisionedThroughputExceededException`

Pesan:

- Tingkat Disediakan terlampaui.
- Batas unduhan S3 terlampaui.

Jumlah permintaan melebihi batas throughput Anda. Untuk informasi selengkapnya, lihat [Service Limits Amazon Rekognition](#).

Untuk meminta penambahan batas, ikuti petunjuk di [the section called “Buat kasus untuk mengubah kuota TPS”](#).

Boleh diulang? Ya

`ResourceAlreadyExistsException`

Pesan: Koleksi id: <Collection Id> sudah ada.

Koleksi dengan ID tertentu sudah ada.

Boleh diulang? Tidak

`ResourceInUseException`

Pesan:

- Nama prosesor streaming sudah digunakan.
- Sumber daya yang ditentukan sedang digunakan.
- Prosesor tidak tersedia untuk menghentikan aliran.
- Tidak dapat menghapus prosesor stream.

Coba lagi bila sumber daya tersedia.

Boleh diulang? Tidak

ResourceNotFoundException

Pesan: Berbagai pesan tergantung pada panggilan API.

Sumber daya yang ditentukan tidak ada.

Boleh diulang? Tidak

ThrottlingException

Pesan: Pelan-pelan; terjadi peningkatan angka permintaan secara mendadak.

Angka peningkatan permintaan Anda terlalu cepat. Perlambat angka permintaan Anda dan tingkatkan secara bertahap. Kami merekomendasikan agar Anda mundur secara eksponensial dan mencoba lagi. Secara default, SDK AWS menggunakan logika pengulangan otomatis dan backoff eksponensial. Untuk informasi selengkapnya, lihat [Pengulangan Kesalahan dan Backoff Eksponensial di AWS](#) dan [Backoff Eksponensial dan Jitter](#).

Boleh diulang? Ya

VideoTooLargeException

Pesan: Ukuran video dalam bit: <Video Size> lebih dari batas maksimum: <Max Size> byte.

Ukuran file atau durasi media yang disediakan terlalu besar. Untuk informasi selengkapnya, lihat [Pedoman dan kuota dalam Amazon Rekognition](#).

Boleh diulang? Tidak

Kode Status HTTP 5xx

Kode status 5xx HTTP menunjukkan masalah yang harus diselesaikan oleh AWS. Ini mungkin kesalahan sementara. Jika ya, Anda dapat mencoba kembali permintaan Anda hingga berhasil. Jika tidak, masuk ke [AWS Service Health Dashboard](#) untuk melihat apakah ada masalah operasional dengan layanan ini.

InternalServerError(HTTP 500)

Pesan: Kesalahan server internal

Amazon Rekognition mengalami masalah layanan. Coba lagi panggilan Anda. Anda harus mundur secara eksponensial dan coba lagi. Secara default, SDK AWS menggunakan logika pengulangan otomatis dan backoff eksponensial. Untuk informasi selengkapnya, lihat [Pengulangan Kesalahan dan Backoff Eksponensial di AWS](#) dan [Backoff Eksponensial dan Jitter](#).

Boleh diulang? Ya

ThrottlingException(HTTP 500)

Pesan: Layanan Tidak Tersedia

Amazon Rekognition untuk sementara tidak dapat memproses permintaan. Coba lagi panggilan Anda. Kami merekomendasikan agar Anda mundur secara eksponensial dan mencoba lagi. Secara default, SDK AWS menggunakan logika pengulangan otomatis dan backoff eksponensial. Untuk informasi selengkapnya, lihat [Pengulangan Kesalahan dan Backoff Eksponensial di AWS](#) dan [Backoff Eksponensial dan Jitter](#).

Boleh diulang? Ya

Penanganan kesalahan dalam aplikasi Anda

Agar aplikasi Anda berjalan lancar, Anda perlu menambahkan logika untuk menangkap kesalahan dan menanggapi. Pendekatan umum termasuk menggunakan blok `try-catch` atau pernyataan `if-then`.

SDK AWS melakukan pemeriksaan pengulangan dan kesalahan mereka sendiri. Jika Anda mengalami kesalahan saat menggunakan salah satu SDK AWS, kode dan deskripsi kesalahan dapat membantu Anda memecahkan masalah itu.

Anda juga harus melihat `Request ID` dalam respons. `Request ID` dapat membantu jika Anda perlu bekerja dengan AWS Support untuk mendiagnosis suatu masalah.

Cuplikan kode Java berikut mencoba untuk mendeteksi objek dalam citra dan melakukan penanganan kesalahan dasar. (Dalam hal ini, itu hanya menginformasikan pengguna bahwa permintaan gagal.)

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

    System.out.println("Detected labels for " + photo);
    for (Label label: labels) {
        System.out.println(label.getName() + ": " + label.getConfidence().toString());
    }
}
catch(AmazonRekognitionException e) {
    System.err.println("Could not complete operation");
    System.err.println("Error Message: " + e.getMessage());
    System.err.println("HTTP Status: " + e.getStatusCode());
    System.err.println("AWS Error Code: " + e.getErrorCode());
    System.err.println("Error Type: " + e.getErrorType());
    System.err.println("Request ID: " + e.getRequestId());
}
catch (AmazonClientException ace) {
    System.err.println("Internal error occurred communicating with Rekognition");
    System.out.println("Error Message: " + ace.getMessage());
}
```

Dalam contoh kode ini, try-catch mencoba menangani dua tipe pengecualian:

- `AmazonRekognitionException` — Pengecualian ini terjadi jika permintaan klien dikirim dengan benar ke Amazon Rekognition, tetapi Amazon Rekognition tidak dapat memproses permintaan dan sebagai gantinya mengembalikan respons kesalahan.
- `AmazonClientException` – Pengecualian ini terjadi jika klien tidak bisa mendapatkan respons dari layanan, atau jika klien tidak bisa mengurai respons dari layanan.

Menggunakan Amazon Rekognition sebagai layanan resmi FedRAMP

Program kepatuhan FedRAMP AWS mencakup Amazon Rekognition sebagai layanan resmi FedRAMP. Jika Anda adalah pelanggan federal atau komersial, Anda dapat menggunakan layanan ini untuk memproses dan menyimpan beban kerja yang sensitif di Wilayah AWS US East dan US West, dengan data hingga tingkat dampak sedang. Anda dapat menggunakan layanan untuk beban kerja sensitif di AWS GovCloud (AS) Batas otorisasi wilayah, dengan data hingga tingkat dampak tinggi. Untuk informasi selengkapnya tentang kepatuhan FedRAMP, lihat [Kepatuhan FedRAMP AWS](#).

Agar FedRAMP patuh, Anda dapat menggunakan titik akhir Federal Information Processing Standard (FIPS). Ini memberi Anda akses ke modul kriptografi tervalidasi FIPS 140-2 ketika Anda bekerja dengan informasi sensitif. Untuk informasi selengkapnya tentang titik akhir FIPS, lihat [Gambaran Umum FIPS 140-2](#).

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) atau salah satu AWS SDK untuk menentukan titik akhir yang digunakan oleh Amazon Rekognition.

Untuk titik akhir yang dapat digunakan dengan Amazon Rekognition, lihat [Wilayah dan Titik Akhir Amazon Rekognition](#).

Berikut ini adalah contoh dari topik [Daftar Koleksi](#) di Panduan Developer Amazon Rekognition. Mereka dimodifikasi untuk menentukan Wilayah dan titik akhir FIPS melalui Amazon Rekognition mana yang diakses.

Java

Java, menggunakan metode `withEndpointConfiguration` ketika Anda membangun klien Amazon Rekognition. Contoh ini menunjukkan koleksi yang Anda miliki yang menggunakan titik akhir FIPS di Wilayah US East (N.Virginia):

```
//Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-
rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;

import java.util.List;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.ListCollectionsRequest;
import com.amazonaws.services.rekognition.model.ListCollectionsResult;

public class ListCollections {

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

        AmazonRekognition amazonRekognition =
            AmazonRekognitionClientBuilder.standard()
```

```
        .withEndpointConfiguration(new
    AwsClientBuilder.EndpointConfiguration("https://rekognition-fips.us-
    east-1.amazonaws.com", "us-east-1"))
        .build();

    System.out.println("Listing collections");
    int limit = 10;
    ListCollectionsResult listCollectionsResult = null;
    String paginationToken = null;
    do {
        if (listCollectionsResult != null) {
            paginationToken = listCollectionsResult.getNextToken();
        }
        ListCollectionsRequest listCollectionsRequest = new
    ListCollectionsRequest()
            .withMaxResults(limit)
            .withNextToken(paginationToken);

    listCollectionsResult=amazonRekognition.listCollections(listCollectionsRequest);

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

AWS CLI

Untuk AWS CLI, gunakan pendapat `--endpoint-url` untuk menentukan titik akhir melalui Amazon Rekognition mana yang diakses. Contoh ini menunjukkan koleksi yang Anda miliki yang menggunakan titik akhir FIPS di Wilayah US East (Ohio):

```
aws rekognition list-collections --endpoint-url https://rekognition-fips.us-
east-2.amazonaws.com --region us-east-2
```

Python

Untuk Python, gunakan pendapat `endpoint_url` di dalam fungsi `boto3.client`. Atur itu ke titik akhir yang ingin Anda tentukan. Contoh ini menunjukkan koleksi yang Anda miliki yang menggunakan titik akhir FIPS di Wilayah US West (Oregon):

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

import boto3

def list_collections():

    max_results=2

    client=boto3.client('rekognition', endpoint_url='https://rekognition-fips.us-
west-2.amazonaws.com', region_name='us-west-2')

    #Display all the collections
    print('Displaying collections...')
    response=client.list_collections(MaxResults=max_results)
    collection_count=0
    done=False

    while done==False:
        collections=response['CollectionIds']

        for collection in collections:
            print (collection)
            collection_count+=1
        if 'NextToken' in response:
            nextToken=response['NextToken']

    response=client.list_collections(NextToken=nextToken,MaxResults=max_results)

    else:
        done=True

    return collection_count

def main():
```

```
    collection_count=list_collections()
    print("collections: " + str(collection_count))
if __name__ == "__main__":
    main()
```

Praktik terbaik untuk sensor, citra input, dan video

Bagian ini berisi informasi praktik terbaik untuk menggunakan Amazon Rekognition.

Topik

- [Latensi operasi Amazon Rekognition Image](#)
- [Rekomendasi untuk citra input perbandingan wajah](#)
- [Rekomendasi untuk pengaturan kamera \(citra dan video\)](#)
- [Rekomendasi untuk pengaturan kamera \(video tersimpan dan streaming\)](#)
- [Rekomendasi untuk pengaturan kamera \(video streaming\)](#)
- [Rekomendasi untuk Penggunaan Face Liveness](#)

Latensi operasi Amazon Rekognition Image

Untuk memastikan latensi serendah mungkin untuk operasi Amazon Rekognition Image, pertimbangkan hal berikut:

- Wilayah untuk bucket Amazon S3 yang berisi citra Anda harus sesuai dengan Wilayah yang digunakan untuk operasi API Amazon Rekognition Image.
- Memanggil operasi Amazon Rekognition Image dengan bit citra lebih cepat daripada mengunggah citra ke bucket Amazon S3 dan kemudian mereferensi citra yang diunggah dalam operasi Amazon Rekognition Image. Pertimbangkan pendekatan ini jika Anda mengunggah citra ke Amazon Rekognition Image untuk pemrosesan mendekati waktu nyata. Misalnya, citra yang diunggah dari kamera atau citra IP yang diunggah melalui portal web.
- Jika citra sudah ada di dalam bucket Amazon S3, mereferensikannya dalam operasi Amazon Rekognition Image mungkin lebih cepat daripada meneruskan bit citra ke operasi.

Rekomendasi untuk citra input perbandingan wajah

Model yang digunakan untuk operasi perbandingan wajah dirancang untuk bekerja dengan berbagai macam pose, ekspresi wajah, rentang usia, rotasi, kondisi pencahayaan, dan ukuran. Kami menyarankan Anda menggunakan pedoman berikut saat memilih foto referensi untuk [CompareFaces](#) atau untuk menambahkan wajah ke koleksi menggunakan [IndexFaces](#).

Rekomendasi umum untuk gambar masukan untuk operasi wajah

- Gunakan citra yang terang dan tajam. Hindari penggunaan gambar yang mungkin buram karena subjek dan gerakan kamera sebanyak mungkin. [DetectFaces](#) dapat digunakan untuk menentukan kecerahan dan ketajaman wajah.
- Untuk tujuan deteksi tatapan, Anda disarankan untuk mengunggah gambar asli dengan ukuran dan kualitas asli.
- Gunakan citra dengan wajah yang berada dalam kisaran sudut yang disarankan. Pitch harus kurang dari 30 derajat menghadap ke bawah dan kurang dari 45 derajat menghadap ke atas. Sudut kemiringan harus kurang dari 45 derajat di kedua arah. Tidak ada batasan pada roll.
- Gunakan citra wajah dengan kedua mata terbuka dan terlihat.
- Gunakan citra wajah yang tidak dikaburkan atau dipotong ketat. Citra harus memperlihatkan kepala dan bahu penuh orang tersebut. Citra tidak dipotong ke kotak batas wajah.
- Hindari item yang menghalangi wajah, seperti ikat kepala dan masker.
- Gunakan citra wajah yang mengisi sebagian besar citra. Citra dengan wajah yang mengisi sebagian besar citra dicocokkan dengan akurasi yang lebih besar.
- Pastikan resolusi citra cukup besar. Amazon Rekognition dapat mengenali wajah berukuran 50 x 50 piksel dalam resolusi citra hingga 1920 x 1080. Citra yang beresolusi lebih tinggi memerlukan ukuran wajah minimum yang lebih besar. Wajah yang lebih besar dari ukuran minimum memberikan serangkaian hasil perbandingan wajah yang lebih akurat.
- Gunakan citra berwarna.
- Gunakan citra dengan pencahayaan datar pada wajah, berlawanan dengan pencahayaan yang bervariasi seperti bayangan.
- Gunakan citra yang memiliki kontras yang cukup dengan latar belakang. Latar belakang monokrom kontras tinggi bekerja dengan baik.
- Gunakan citra wajah dengan ekspresi wajah netral dengan mulut tertutup dan sedikit atau tanpa senyum untuk aplikasi yang membutuhkan presisi tinggi.

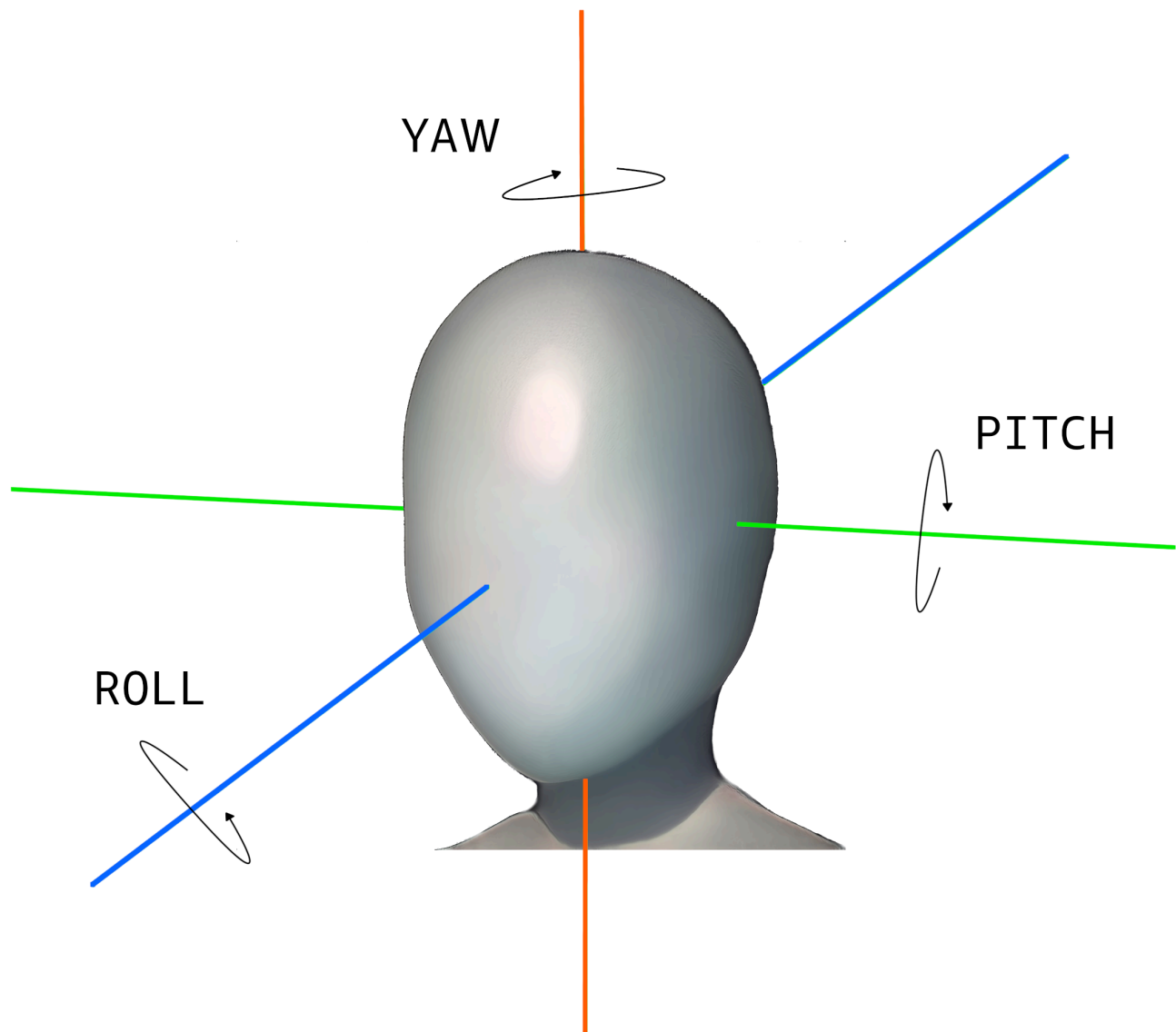
Rekomendasi untuk mencari wajah dalam koleksi

- Saat mencari wajah dalam koleksi, pastikan gambar wajah terbaru diindeks.
- Saat membuat koleksi menggunakan IndexFaces, gunakan beberapa citra wajah individu dengan sudut kemiringan vertikal dan horizontal yang berbeda (dalam kisaran sudut yang disarankan). Kami merekomendasikan setidaknya lima citra orang tersebut terindeks—lurus, wajah menghadap

ke kiri dengan sudut kemiringan horizontal 45 derajat atau kurang, wajah menghadap ke kanan dengan sudut kemiringan horizontal 45 derajat atau kurang, wajah menghadap ke bawah dengan sudut kemiringan vertikal 30 derajat atau kurang, dan wajah menghadap ke atas dengan sudut kemiringan vertikal 45 derajat atau kurang. Jika Anda ingin melacak bahwa instans wajah ini milik individu yang sama, pertimbangkan untuk menggunakan atribut ID citra eksternal jika hanya ada satu wajah dalam citra yang terindeks. Misalnya, lima gambar John Doe dapat dilacak dalam koleksi dengan ID gambar eksternal sebagai `John_Doe_1.jpg`, ... `John_Doe_5.jpg`

Rekomendasi untuk pengaturan kamera (citra dan video)

Rekomendasi berikut adalah tambahan untuk [Rekomendasi untuk citra input perbandingan wajah](#).



- Resolusi citra – Tidak ada persyaratan minimum untuk resolusi citra, selama resolusi wajah adalah 50 x 50 piksel untuk citra dengan resolusi total hingga 1920 x 1080. Citra yang beresolusi lebih tinggi memerlukan ukuran wajah minimum yang lebih besar.

Note

Rekomendasi sebelumnya didasarkan pada resolusi asli kamera. Membuat citra beresolusi tinggi dari citra beresolusi rendah tidak memberikan hasil yang diperlukan untuk pencarian wajah (karena artefak yang dihasilkan oleh up-sampling citra).

- Sudut Kamera – Ada tiga pengukuran untuk sudut kamera—sudut kemiringan vertikal dan horizontal dan roll.
- Sudut Kemiringan Vertikal – Kami merekomendasikan sudut kemiringan vertikal kurang dari 30 derajat saat kamera menghadap ke bawah dan kurang dari 45 derajat saat kamera menghadap ke atas.
- Roll – Tidak ada persyaratan minimum untuk parameter ini. Amazon Rekognition dapat menangani sejumlah roll.
- Sudut kemiringan horizontal – Kami merekomendasikan sudut kemiringan horizontal kurang dari 45 derajat di kedua arah.

Sudut wajah di sepanjang sumbu apa pun yang ditangkap oleh kamera adalah kombinasi dari kedua sudut kamera yang menghadap layar dan sudut kepala subjek berada dalam layar. Contohnya, jika kamera 30 derajat menghadap ke bawah dan orang tersebut menundukkan kepala 30 derajat, pitch wajah yang sebenarnya seperti yang terlihat oleh kamera adalah 60 derajat. Dalam hal ini, Amazon Rekognition tidak akan bisa mengenali wajah. Sebaiknya atur kamera sedemikian rupa sehingga sudut kamera didasarkan pada asumsi bahwa orang umumnya melihat ke kamera dengan sudut kemiringan vertikal keseluruhan (kombinasi wajah dan kamera) pada 30 derajat atau kurang.

- Perbesaran Kamera – Resolusi wajah minimum yang disarankan 50 x 50 piksel harus mendorong pengaturan kamera ini. Sebaiknya gunakan pengaturan perbesaran kamera agar wajah yang diinginkan berada pada resolusi tidak kurang dari 50 x 50 piksel.
- Tinggi Kamera – Sudut kemiringan vertikal kamera yang disarankan harus mendukung parameter ini.

Rekomendasi untuk pengaturan kamera (video tersimpan dan streaming)

Rekomendasi berikut adalah tambahan untuk [Rekomendasi untuk pengaturan kamera \(citra dan video\)](#).

- Codec harus dikodekan h.264.
- Kecepatan frame yang direkomendasikan adalah 30 fps. (Tidak boleh kurang dari 5 fps.)
- Bitrate encoder yang direkomendasikan adalah 3 Mbps. (Tidak boleh kurang dari 1,5 Mbps.)
- Kecepatan frame vs Resolusi Frame – Jika bitrate encoder merupakan kendala, sebaiknya pilih resolusi bingkai yang lebih tinggi pada frame rate yang lebih tinggi untuk hasil pencarian wajah yang lebih baik. Hal ini memastikan agar Amazon Rekognition mendapat frame kualitas terbaik dalam bitrate yang dialokasikan. Namun, terdapat kekurangan. Karena kecepatan frame yang rendah, kamera melewatkan gerakan cepat di layar. Sangat penting untuk memahami pertukaran antara dua parameter ini untuk pengaturan tertentu. Misalnya, jika bitrate maksimum yang mungkin adalah 1,5 Mbps, kamera dapat menangkap 1080p pada 5 fps atau 720p pada 15 fps. Pilihan antara keduanya bergantung pada aplikasi, selama resolusi wajah 50 x 50 piksel yang direkomendasikan terpenuhi.

Rekomendasi untuk pengaturan kamera (video streaming)

Rekomendasi berikut adalah tambahan untuk [Rekomendasi untuk pengaturan kamera \(video tersimpan dan streaming\)](#).

Kendala tambahan dengan aplikasi streaming adalah bandwidth internet. Untuk video yang ditayangkan secara langsung, Amazon Rekognition hanya menerima Amazon Kinesis Video Streams sebagai input. Anda harus memahami ketergantungan antara bitrate encoder dan bandwidth jaringan yang tersedia. Bandwidth yang tersedia minimal harus mendukung bitrate yang digunakan kamera untuk mengodekan streaming langsung. Hal ini memastikan agar apa pun yang ditangkap kamera disampaikan melalui Amazon Kinesis Video Streams. Jika bandwidth yang tersedia kurang dari bitrate encoder, Amazon Kinesis Video Streams menurunkan bit berdasarkan bandwidth jaringan. Hal ini menghasilkan kualitas video yang rendah.

Pengaturan streaming yang khas melibatkan menghubungkan beberapa kamera ke hub jaringan yang menyambungkan aliran. Dalam hal ini, bandwidth harus mengakomodasi jumlah kumulatif dari

aliran yang berasal dari semua kamera yang terhubung ke hub. Misalnya, jika hub terhubung ke lima kamera yang dikodekan pada 1,5 Mbps, bandwidth jaringan yang tersedia harus minimal 7,5 Mbps. Untuk memastikan bahwa tidak ada paket yang dihapus, Anda harus mempertimbangkan menjaga bandwidth jaringan lebih tinggi dari 7,5 Mbps untuk menanggulangi citra yang cacat karena koneksi antara kamera dan hub yang turun. Nilai sebenarnya tergantung pada keandalan jaringan internal.

Rekomendasi untuk Penggunaan Face Liveness

Kami merekomendasikan praktik terbaik berikut saat menggunakan Rekognition Face Liveness:

- Pengguna harus menyelesaikan pemeriksaan Face Liveness di lingkungan yang tidak terlalu gelap atau terlalu terang dan memiliki pencahayaan yang cukup seragam.
- Pengguna harus meningkatkan kecerahan layar tampilan mereka ke tingkat maksimum saat melakukan pemeriksaan pada browser web. Mobile Native SDK menyesuaikan kecerahan tampilan secara otomatis.
- Pilih ambang batas skor kepercayaan yang mencerminkan sifat kasus penggunaan Anda. Untuk kasus penggunaan dengan masalah keamanan yang lebih besar, gunakan ambang batas tinggi.
- Jalankan pemeriksaan tinjauan manusia secara teratur pada gambar audit untuk memastikan bahwa serangan spoof dikurangi pada ambang kepercayaan yang Anda tetapkan.
- Tawarkan jalur verifikasi keaktifan wajah alternatif kepada pengguna Anda jika mereka peka terhadap foto atau tidak ingin memverifikasi keaktifan wajah mereka menggunakan Rekognition.
- Jangan mengirim atau menampilkan skor pemeriksaan keaktifan pada aplikasi pengguna. Hanya kirim sinyal pass atau gagal.
- Izinkan hanya lima pemeriksaan keaktifan yang gagal dalam tiga menit dari satu perangkat. Setelah lima gagal, batas waktu pengguna selama 30-60 menit. Jika pola terlihat 3-5 kali berulang kali, blokir perangkat pengguna agar tidak melakukan panggilan tambahan.
- Terapkan layar bersiap-siap di alur kerja Anda sehingga pengguna dapat lebih mudah melewati pemeriksaan Face Liveness.
- Anda bertanggung jawab untuk memberikan pemberitahuan privasi yang memadai secara hukum kepada, dan mendapatkan persetujuan yang diperlukan dari, Pengguna Akhir Anda untuk pemrosesan, penyimpanan, penggunaan, dan transfer konten oleh Face Liveness.

Mendeteksi objek dan konsep

Bagian ini memberikan informasi untuk mendeteksi label dalam citra dan video dengan Amazon Rekognition Image dan Amazon Rekognition Video.

Label atau tag adalah objek atau konsep (termasuk adegan dan tindakan) yang ditemukan dalam gambar atau video berdasarkan isinya. Misalnya, gambar orang di pantai tropis mungkin berisi label seperti Pohon Palem (objek), Pantai (adegan), Lari (aksi), dan Outdoor (konsep).

Label didukung oleh operasi deteksi label Rekognition

- [Untuk mengunduh daftar label dan kotak pembatas objek terbaru yang didukung oleh Amazon Rekognition, klik di sini.](#)
- Untuk mengunduh daftar label dan kotak pembatas objek sebelumnya, klik [di sini](#).

Note

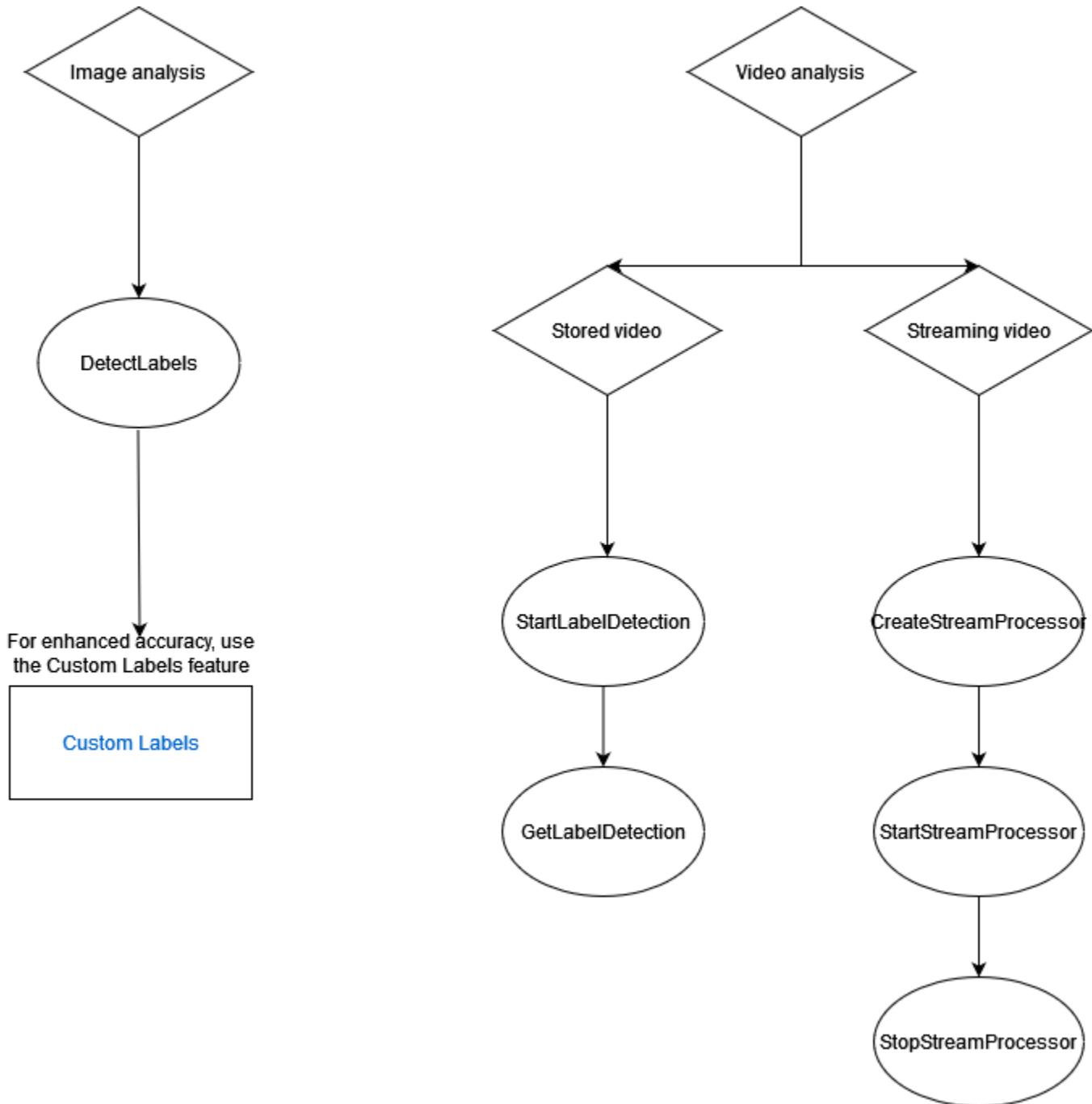
Amazon Rekognition membuat prediksi biner gender (pria, wanita, gadis, dll.) berdasarkan penampilan fisik seseorang dalam citra tertentu. Prediksi semacam ini tidak dirancang untuk mengategorikan identitas gender seseorang, dan Anda tidak seharusnya menggunakan Amazon Rekognition untuk membuat penentuan seperti itu. Misalnya, seorang aktor pria yang mengenakan wig berambut panjang serta anting-anting untuk suatu peran mungkin diprediksi sebagai perempuan.

Menggunakan Amazon Rekognition untuk membuat prediksi biner gender paling cocok untuk kasus penggunaan jika statistik distribusi gender agregat perlu dianalisis tanpa mengidentifikasi pengguna tertentu. Misalnya, persentase pengguna perempuan dibandingkan dengan laki-laki di platform media sosial.

Kami tidak menyarankan penggunaan prediksi biner gender untuk membuat keputusan yang memengaruhi hak, privasi, atau akses seseorang ke layanan.

Amazon Rekognition mengembalikan label dalam bahasa Inggris. Anda dapat menggunakan [Amazon Translate](#) untuk menerjemahkan label bahasa Inggris ke [bahasa lain](#).

Diagram berikut menunjukkan urutan operasi panggilan, tergantung pada tujuan Anda untuk menggunakan Gambar Rekognition Amazon atau operasi Video Rekognition Amazon:



Objek Respons Label

Kotak Pembatas

Amazon Rekognition Image dan Amazon Rekognition Video dapat mengembalikan kotak pembatas untuk label objek umum seperti mobil, furnitur, pakaian atau hewan peliharaan. Informasi kotak pembatas tidak dikembalikan untuk label objek yang kurang lazim. Anda dapat menggunakan kotak

pembatas untuk menemukan lokasi objek yang tepat dalam citra, menghitung instans objek yang terdeteksi, atau mengukur ukuran objek menggunakan dimensi kotak pembatas.

Misalnya, pada citra berikut, Amazon Rekognition Image mampu mendeteksi keberadaan seseorang, papan luncur, mobil yang terparkir dan informasi lainnya. Amazon Rekognition Image juga mengembalikan kotak pembatas untuk seseorang yang terdeteksi, dan benda-benda lain yang terdeteksi seperti mobil dan roda.



Skor Keyakinan

Video Rekognition Amazon dan Gambar Rekognition Amazon memberikan skor persentase untuk seberapa besar kepercayaan Amazon Rekognition terhadap keakuratan setiap label yang terdeteksi.

Induk

Amazon Rekognition Image dan Amazon Rekognition Video menggunakan taksonomi hierarkis label leluhur untuk mengategorikan label. Misalnya, seseorang yang berjalan di seberang jalan mungkin terdeteksi sebagai Pejalan Kaki. Label induk untuk Pejalan Kaki adalah Orang. Kedua label ini

dikembalikan dalam respons. Semua label leluhur dikembalikan dan label yang diberikan berisi daftar induknya dan label leluhur lainnya. Misalnya, label kakek-nenek dan kakek-nenek buyut, jika ada. Anda dapat menggunakan label induk untuk membangun grup label terkait dan mengizinkan kueri label serupa dalam satu atau beberapa citra. Sebagai contoh, kueri untuk semua Kendaraan mungkin mengembalikan mobil dari satu citra dan sepeda motor dari citra lainnya.

Kategori

Gambar Rekognition Amazon dan Video Rekognition Amazon mengembalikan informasi tentang kategori label. Label adalah bagian dari kategori yang mengelompokkan masing-masing label berdasarkan fungsi dan konteks umum, seperti 'Kendaraan dan Otomotif' dan 'Makanan dan Minuman'. Kategori label dapat berupa subkategori dari kategori induk.

Alias

Selain mengembalikan label, Amazon Rekognition Image dan Amazon Rekognition Video mengembalikan alias apa pun yang terkait dengan label. Alias adalah label dengan arti atau label yang sama yang dapat dipertukarkan secara visual dengan label utama yang dikembalikan. Misalnya, 'Telepon Seluler' adalah alias 'Ponsel'.

Di versi sebelumnya, Amazon Rekognition Image mengembalikan alias seperti 'Ponsel' dalam daftar nama label utama yang sama yang berisi 'Ponsel'. Amazon Rekognition Image sekarang mengembalikan 'Ponsel' di bidang yang disebut "alias" dan 'Ponsel' dalam daftar nama label utama. Jika aplikasi Anda bergantung pada struktur yang dikembalikan oleh Rekognition versi sebelumnya, Anda mungkin perlu mengubah respons saat ini yang dikembalikan oleh operasi deteksi label gambar atau video ke dalam struktur respons sebelumnya, di mana semua label dan alias dikembalikan sebagai label utama.

Jika Anda perlu mengubah respons saat ini dari DetectLabels API (untuk deteksi label dalam gambar) menjadi struktur respons sebelumnya, lihat contoh kode di [Mengubah respon DetectLabels](#).

Jika Anda perlu mengubah respons saat ini dari GetLabelDetection API (untuk deteksi label dalam video yang disimpan) ke dalam struktur respons sebelumnya, lihat contoh kode di [Mengubah Respon GetLabelDetection](#).


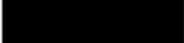





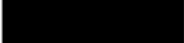




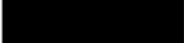

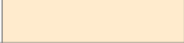


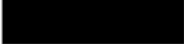


Properti Gambar

Amazon Rekognition Image mengembalikan informasi tentang kualitas gambar (ketajaman, kecerahan, dan kontras) untuk seluruh gambar. Ketajaman dan kecerahan juga dikembalikan untuk

latar depan dan latar belakang gambar. Image Properties juga dapat digunakan untuk mendeteksi warna dominan dari seluruh gambar, latar depan, latar belakang, dan objek dengan kotak pembatas.



Berikut ini adalah contoh ImageProperties data yang terkandung dalam respons DetectLabels operasi untuk gambar yang sedang berlangsung:

Image Properties	Dominant Colors Examples and Pixel Percentage		Image Quality
Entire Image		Hex code #808080, RGB (128, 128, 128), 15.72	Brightness: 76.08 Sharpness: 89.72 Contrast: 88.42
		Hex code #000000, RGB (0, 0, 0), 15.10	
		Hex code #696969, RGB (105, 105, 105), 14.02	
		Hex code #8fbc8f, RGB (143, 188, 143), 12.70	
		Hex code #5f9ea0, RGB (95, 158, 160), 11.92	
Foreground		Hex code #8fbc8f, RGB (143, 188, 143), 30.18	Brightness: 79.48 Sharpness: 93.47
		Hex code #5f9ea0, RGB (95, 158, 160), 24.29	
		Hex code #000000, RGB (0, 0, 0), 12.02	
		Hex code #2f4f4f, RGB (47, 79, 79), 9.20	
		Hex code #696969, RGB (105, 105, 105), 8.95	
Background		Hex code #808080, RGB (128, 128, 128), 21.16	Brightness: 74.42 Sharpness: 87.84
		Hex code #2f4f4f, RGB (47, 79, 79), 14.61	
		Hex code #000000, RGB (0, 0, 0), 14.23	
		Hex code #696969, RGB (105, 105, 105), 13.19	
		Hex code #ffebcd, RGB (255, 235, 205), 12.80	
Car (example of objects with bounding boxes)		Hex code #5f9ea0, RGB (95, 158, 160), 29.18	Not applicable
		Hex code #8fbc8f, RGB (143, 188, 143), 14.39	
		Hex code #000000, RGB (0, 0, 0), 11.76	
		Hex code #808080, RGB (128, 128, 128), 11.38	
		Hex code #2f4f4f, RGB (47, 79, 79), 9.44	

Properti Gambar tidak tersedia untuk Amazon Rekognition Video.

Versi Model

Amazon Rekognition Image dan Amazon Rekognition Video mengembalikan versi model deteksi label yang digunakan untuk mendeteksi label dalam citra atau video yang disimpan.

Filter Inklusi dan Pengecualian

Anda dapat memfilter hasil yang ditampilkan oleh Amazon Rekognition Image dan operasi deteksi label Amazon Rekognition Video. Filter hasil dengan memberikan kriteria filtrasi untuk label dan kategori. Filter label bisa inklusif atau eksklusif.

Lihat [Mendeteksi label dalam citra](#) untuk informasi lebih lanjut mengenai penyaringan hasil yang diperoleh dengan `DetectLabels`.

Lihat [Mendeteksi label dalam video](#) untuk informasi lebih lanjut mengenai penyaringan hasil yang diperoleh oleh `GetLabelDetection`.

Menyortir dan Mengagregasikan Hasil

Hasil yang diperoleh dari operasi Video Rekognition Amazon tertentu dapat diurutkan dan dikumpulkan menurut stempel waktu dan segmen video. Saat mengambil hasil pekerjaan Deteksi Label atau Moderasi Konten, dengan `GetLabelDetection` atau `GetContentModeration` masing-masing, Anda dapat menggunakan `AggregateBy` argumen `SortBy` dan untuk menentukan bagaimana Anda ingin hasil Anda dikembalikan. Anda dapat menggunakan `SortBy` dengan `TIMESTAMP` atau `NAME` (nama Label), dan menggunakan `TIMESTAMPS` atau `SEGMENTS` dengan `AggregateBy` argumen.

Mendeteksi label dalam citra

Anda dapat menggunakan [DetectLabels](#) operasi untuk mendeteksi label (objek dan konsep) dalam gambar dan mengambil informasi tentang properti gambar. Properti gambar mencakup atribut seperti warna latar depan dan latar belakang dan ketajaman, kecerahan, dan kontras gambar. Anda dapat mengambil hanya label dalam gambar, hanya properti gambar, atau keduanya. Sebagai contoh, lihat [Menganalisis citra yang disimpan di bucket Amazon S3](#).

Contoh berikut menggunakan berbagai AWS SDK dan AWS CLI to `callDetectLabels`. Untuk informasi tentang respons operasi `DetectLabels`, lihat [DetectLabels respon](#).

Untuk mendeteksi label dalam citra

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan `AmazonRekognitionFullAccess` dan `AmazonS3ReadOnlyAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Unggah citra yang berisi satu atau beberapa objek—seperti pohon, rumah, dan perahu—ke bucket S3 Anda. Citra harus dalam format `.jpg` atau `.png`.

Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

3. Gunakan contoh berikut untuk memanggil operasi DetectLabels.

Java

Contoh ini menampilkan daftar label yang terdeteksi pada citra input. Ganti nilai-nilai bucket dan photo dengan nama bucket Amazon S3 dan citra yang Anda gunakan di langkah 2.

```
package com.amazonaws.samples;
import java.util.List;

import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.DetectLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.Instance;
import com.amazonaws.services.rekognition.model.Label;
import com.amazonaws.services.rekognition.model.Parent;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;

public class DetectLabels {

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

        String photo = "photo";
        String bucket = "bucket";

        AmazonRekognition rekognitionClient =
AmazonRekognitionClientBuilder.defaultClient();

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

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

            System.out.println("Detected labels for " + photo + "\n");
        }
    }
}
```

```

        for (Label label : labels) {
            System.out.println("Label: " + label.getName());
            System.out.println("Confidence: " +
label.getConfidence().toString() + "\n");

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

            List<Parent> parents = label.getParents();
            if (parents.isEmpty()) {
                System.out.println("  None");
            } else {
                for (Parent parent : parents) {
                    System.out.println("  " + parent.getName());
                }
            }
            System.out.println("-----");
            System.out.println();

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

```

AWS CLI

Contoh ini menampilkan output JSON dari operasi CLI `detect-labels`. Ganti nilai-nilai bucket dan photo dengan nama bucket Amazon S3 dan citra yang Anda gunakan di Langkah 2. Ganti nilai `profile-name` dengan nama profil pengembang Anda.

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

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda alih-alih tanda kutip tunggal dan hindari tanda kutip ganda bagian dalam dengan garis miring terbalik (yaitu \) untuk mengatasi kesalahan parser yang mungkin Anda temui. Sebagai contoh, lihat berikut ini:

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

Python

Contoh ini menampilkan label yang terdeteksi dalam citra input. Dalam fungsi main, ganti nilai-nilai bucket dan photo dengan nama bucket Amazon S3 dan citra yang Anda gunakan pada Langkah 2. Ganti nilai profile_name di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import boto3

def detect_labels(photo, bucket):

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

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

```
MaxLabels=10,
# Uncomment to use image properties and filtration settings
#Features=["GENERAL_LABELS", "IMAGE_PROPERTIES"],
#Settings={"GeneralLabels": {"LabelInclusionFilters":["Cat"]},
# "ImageProperties": {"MaxDominantColors":10}}
)

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

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

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

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

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

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



```
        print("Quality:")
        print(response["ImageProperties"]["Quality"])
        print()

    return len(response['Labels'])

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

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

.NET

Contoh ini menampilkan daftar label yang terdeteksi pada citra input. Ganti nilai-nilai bucket dan photo dengan nama bucket Amazon S3 dan citra yang Anda gunakan di Langkah 2.

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

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

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

        AmazonRekognitionClient rekognitionClient = new
AmazonRekognitionClient();

        DetectLabelsRequest detectlabelsRequest = new DetectLabelsRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
```

```
        {
            Name = photo,
            Bucket = bucket
        },
    },
    MaxLabels = 10,
    MinConfidence = 75F
};

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

Ruby

Contoh ini menampilkan daftar label yang terdeteksi pada citra input. Ganti nilai-nilai bucket dan photo dengan nama bucket Amazon S3 dan citra yang Anda gunakan di Langkah 2.

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

```
s3_object: {
  bucket: bucket,
  name: photo
},
},
max_labels: 10
}
response = client.detect_labels attrs
puts "Detected labels for: #{photo}"
response.labels.each do |label|
  puts "Label:      #{label.name}"
  puts "Confidence: #{label.confidence}"
  puts "Instances:"
  label['instances'].each do |instance|
    box = instance['bounding_box']
    puts "  Bounding box:"
    puts "    Top:      #{box.top}"
    puts "    Left:     #{box.left}"
    puts "    Width:    #{box.width}"
    puts "    Height:   #{box.height}"
    puts "    Confidence: #{instance.confidence}"
  end
  puts "Parents:"
  label.parents.each do |parent|
    puts "  #{parent.name}"
  end
  puts "-----"
  puts ""
end
```

Node.js

Contoh ini menampilkan daftar label yang terdeteksi pada citra input. Ganti nilai-nilai bucket dan photo dengan nama bucket Amazon S3 dan citra yang Anda gunakan di Langkah 2. Ganti nilai profile_name di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

Jika Anda menggunakan TypeScript definisi, Anda mungkin perlu menggunakan `import AWS from 'aws-sdk'` alih-alih `const AWS = require('aws-sdk')`, untuk menjalankan program dengan Node.js. Anda dapat melihat [AWS SDK for Javascript](#) untuk detailnya selengkapnya. Tergantung bagaimana konfigurasi diatur, Anda juga mungkin perlu menentukan wilayah Anda dengan `AWS.config.update({region: region});`.

```
// Load the SDK
var AWS = require('aws-sdk');
const bucket = 'bucket-name' // the bucketname without s3://
const photo = 'image-name' // the name of file

var credentials = new AWS.SharedIniFileCredentials({profile: 'profile-name'});
AWS.config.credentials = credentials;
AWS.config.update({region: 'region-name'});

const client = new AWS.Rekognition();
const params = {
  Image: {
    S3Object: {
      Bucket: bucket,
      Name: photo
    },
  },
  MaxLabels: 10
}
client.detectLabels(params, function(err, response) {
  if (err) {
    console.log(err, err.stack); // if an error occurred
  } else {
    console.log(`Detected labels for: ${photo}`)
    response.Labels.forEach(label => {
      console.log(`Label:      ${label.Name}`)
      console.log(`Confidence: ${label.Confidence}`)
      console.log("Instances:")
      label.Instances.forEach(instance => {
        let box = instance.BoundingBox
        console.log(" Bounding box:")
        console.log(`   Top:      ${box.Top}`)
        console.log(`   Left:     ${box.Left}`)
        console.log(`   Width:    ${box.Width}`)
        console.log(`   Height:   ${box.Height}`)
        console.log(` Confidence: ${instance.Confidence}`)
      })
      console.log("Parents:")
      label.Parents.forEach(parent => {
        console.log(`  ${parent.Name}`)
      })
    })
  }
})
```

```
        console.log("-----")
        console.log("")
    }) // for response.labels
} // if
});
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

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

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

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "  <bucket> <image>\n\n" +
            "Where:\n" +
```

```
        " bucket - The name of the Amazon S3 bucket that contains the
image (for example, ,ImageBucket)." +
        " image - The name of the image located in the Amazon S3 bucket
(for example, Lake.png). \n\n";

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

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

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

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

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

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

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

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

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

DetectLabels permintaan operasi

Input ke `DetectLabel` adalah citra. Dalam contoh input JSON ini, citra sumber dimuat dari Bucket Amazon S3. `MaxLabels` adalah jumlah maksimum label yang dikembalikan dalam respons. `MinConfidence` adalah kepercayaan minimum yang harus dimiliki Amazon Rekognition Image dalam keakuratan label yang terdeteksi agar dapat dikembalikan dalam respons.

Fitur memungkinkan Anda menentukan satu atau lebih fitur gambar yang ingin Anda kembalikan, memungkinkan Anda untuk memilih `GENERAL_LABELS` dan `IMAGE_PROPERTIES`. Termasuk `GENERAL_LABELS` akan mengembalikan label yang terdeteksi dalam gambar input, sementara termasuk `IMAGE_PROPERTIES` akan memungkinkan Anda untuk mengakses warna dan kualitas gambar.

Pengaturan memungkinkan Anda memfilter item yang dikembalikan untuk `IMAGE_PROPERTIES` fitur `GENERAL_LABELS` dan fitur. Untuk label, Anda dapat menggunakan filter inklusif dan eksklusif. Anda juga dapat memfilter berdasarkan label khusus, label individual atau berdasarkan kategori label:

- `LabelInclusionFilters` - Memungkinkan Anda menentukan label mana yang ingin Anda sertakan dalam respons.
- `LabelExclusionFilters` - Memungkinkan Anda menentukan label mana yang ingin Anda kecualikan dari respons.

- `LabelCategoryInclusionFilters` - Memungkinkan Anda menentukan kategori label mana yang ingin Anda sertakan dalam respons.
- `LabelCategoryExclusionFilters` - Memungkinkan Anda menentukan kategori label mana yang ingin Anda kecualikan dari respons.

Anda juga dapat menggabungkan filter inklusif dan eksklusif sesuai dengan kebutuhan Anda, tidak termasuk beberapa label atau kategori dan termasuk yang lain.

`IMAGE_PROPERTIES` mengacu pada warna dominan gambar dan atribut kualitas seperti ketajaman, kecerahan, dan kontras. Saat mendeteksi `IMAGE_PROPERTIES` Anda dapat menentukan jumlah maksimum warna dominan untuk kembali (default adalah 10) dengan menggunakan `MaxDominantColors` parameter.

```
{
  "Image": {
    "S3Object": {
      "Bucket": "bucket",
      "Name": "input.jpg"
    }
  },
  "MaxLabels": 10,
  "MinConfidence": 75,
  "Features": [ "GENERAL_LABELS", "IMAGE_PROPERTIES" ],
  "Settings": {
    "GeneralLabels": {
      "LabelInclusionFilters": [<Label(s)>],
      "LabelExclusionFilters": [<Label(s)>],
      "LabelCategoryInclusionFilters": [<Category Name(s)>],
      "LabelCategoryExclusionFilters": [<Category Name(s)>]
    },
    "ImageProperties": {
      "MaxDominantColors": 10
    }
  }
}
```

DetectLabels respon

Respons dari `DetectLabels` adalah array label yang terdeteksi dalam citra dan tingkat kepercayaan yang digunakan untuk mendeteksinya.

Berikut ini adalah contoh respons dari DetectLabels. Contoh respons di bawah ini berisi berbagai atribut yang dikembalikan untuk GENERAL_LABELS, termasuk:

- Nama - Nama label yang terdeteksi. Dalam contoh ini, operasi mendeteksi objek dengan label Ponsel.
- Keyakinan - Setiap label memiliki tingkat kepercayaan yang terkait. Dalam contoh ini, kepercayaan untuk label adalah 99,36%.
- Orang tua - Label leluhur untuk label yang terdeteksi. Dalam contoh ini, label Ponsel memiliki satu label induk bernama Telepon.
- Alias - Informasi tentang kemungkinan Alias untuk label. Dalam contoh ini, label Ponsel memiliki kemungkinan alias Ponsel.
- Kategori - Kategori label yang dimiliki label yang terdeteksi. Dalam contoh ini, itu adalah Teknologi dan Komputasi.

Respons untuk label objek umum mencakup informasi kotak pembatas untuk lokasi label pada citra input. Misalnya, label Orang memiliki array instans yang berisi dua kotak batas. Ini adalah lokasi dari dua orang yang terdeteksi dalam citra.

Respons juga mencakup atribut mengenai IMAGE_PROPERTIES. Atribut yang disajikan oleh fitur IMAGE_PROPERTIES adalah:

- Kualitas - Informasi tentang Ketajaman, Kecerahan, dan Kontras gambar input, diberi skor antara 0 hingga 100. Kualitas dilaporkan untuk seluruh gambar dan untuk latar belakang dan latar depan gambar, jika tersedia. Namun, Kontras hanya dilaporkan untuk seluruh gambar sementara Ketajaman dan Kecerahan juga dilaporkan untuk Latar Belakang dan Latar Depan.
- Warna Dominan - Array warna dominan dalam gambar. Setiap warna dominan dijelaskan dengan nama warna yang disederhanakan, palet warna CSS, nilai RGB, dan kode hex.
- Latar Depan - Informasi tentang Warna dominan, Ketajaman dan Kecerahan latar depan gambar input.
- Latar Belakang - Informasi tentang Warna dominan, Ketajaman dan Kecerahan latar belakang gambar input.

Ketika GENERAL_LABELS dan IMAGE_PROPERTIES digunakan bersama sebagai parameter input, Amazon Rekognition Image juga akan mengembalikan warna dominan objek dengan kotak pembatas.

Bidang `LabelModelVersion` berisi nomor versi model deteksi yang digunakan oleh `DetectLabels`.

```
{
  "Labels": [
    {
      "Name": "Mobile Phone",
      "Parents": [
        {
          "Name": "Phone"
        }
      ],
      "Aliases": [
        {
          "Name": "Cell Phone"
        }
      ],
      "Categories": [
        {
          "Name": "Technology and Computing"
        }
      ],
      "Confidence": 99.9364013671875,
      "Instances": [
        {
          "BoundingBox": {
            "Width": 0.26779675483703613,
            "Height": 0.8562285900115967,
            "Left": 0.3604024350643158,
            "Top": 0.09245597571134567,
          }
          "Confidence": 99.9364013671875,
          "DominantColors": [
            {
              "Red": 120,
              "Green": 137,
              "Blue": 132,
              "HexCode": "3A7432",
              "SimplifiedColor": "red",
              "CssColor": "fuchsia",
              "PixelPercentage": 40.10
            }
          ]
        }
      ]
    }
  ]
}
```

```
    ],
  },
]
},
],
"ImageProperties": {
  "Quality": {
    "Brightness": 40,
    "Sharpness": 40,
    "Contrast": 24,
  },
  "DominantColors": [
    {
      "Red": 120,
      "Green": 137,
      "Blue": 132,
      "HexCode": "3A7432",
      "SimplifiedColor": "red",
      "CssColor": "fuchsia",
      "PixelPercentage": 40.10
    }
  ],
  "Foreground": {
    "Quality": {
      "Brightness": 40,
      "Sharpness": 40,
    },
    "DominantColors": [
      {
        "Red": 200,
        "Green": 137,
        "Blue": 132,
        "HexCode": "3A7432",
        "CSSColor": "",
        "SimplifiedColor": "red",
        "PixelPercentage": 30.70
      }
    ],
  },
  "Background": {
    "Quality": {
      "Brightness": 40,
      "Sharpness": 40,
    },
  },
}
```

```
    "DominantColors": [
      {
        "Red": 200,
        "Green": 137,
        "Blue": 132,
        "HexCode": "3A7432",
        "CSSColor": "",
        "SimplifiedColor": "Red",
        "PixelPercentage": 10.20
      }
    ],
  },
  "LabelModelVersion": "3.0"
}
```

Mengubah respon DetectLabels

Saat menggunakan DetectLabels API, Anda mungkin memerlukan struktur respons untuk meniru struktur respons API yang lebih lama, di mana label utama dan alias terkandung dalam daftar yang sama.

Berikut ini adalah contoh respons API saat ini dari [DetectLabels](#):

```
"Labels": [
  {
    "Name": "Mobile Phone",
    "Confidence": 99.99717712402344,
    "Instances": [],
    "Parents": [
      {
        "Name": "Phone"
      }
    ],
    "Aliases": [
      {
        "Name": "Cell Phone"
      }
    ]
  }
]
```

Contoh berikut menunjukkan respons sebelumnya dari [DetectLabels](#) API:

```
"Labels": [
  {
    "Name": "Mobile Phone",
    "Confidence": 99.99717712402344,
    "Instances": [],
    "Parents": [
      {
        "Name": "Phone"
      }
    ]
  },
  {
    "Name": "Cell Phone",
    "Confidence": 99.99717712402344,
    "Instances": [],
    "Parents": [
      {
        "Name": "Phone"
      }
    ]
  },
]
```

Jika diperlukan, Anda dapat mengubah respons saat ini untuk mengikuti format respons yang lebih lama. Anda dapat menggunakan kode contoh berikut untuk mengubah respons API terbaru ke struktur respons API sebelumnya:

Python

Contoh kode berikut menunjukkan cara mengubah respons saat ini dari DetectLabels API. Dalam contoh kode di bawah ini, Anda dapat mengganti nilai *EXAMPLE_INFERENCE_OUTPUT* dengan *output* dari operasi yang telah Anda jalankan. DetectLabels

```
from copy import deepcopy

LABEL_KEY = "Labels"
ALIASES_KEY = "Aliases"
INSTANCE_KEY = "Instances"
NAME_KEY = "Name"
```

```
#Latest API response sample
EXAMPLE_INFERENCE_OUTPUT = {
    "Labels": [
        {
            "Name": "Mobile Phone",
            "Confidence": 97.530106,
            "Categories": [
                {
                    "Name": "Technology and Computing"
                }
            ],
            "Aliases": [
                {
                    "Name": "Cell Phone"
                }
            ],
            "Instances": [
                {
                    "BoundingBox": {
                        "Height": 0.1549897,
                        "Width": 0.07747964,
                        "Top": 0.50858885,
                        "Left": 0.00018205095
                    },
                    "Confidence": 98.401276
                }
            ]
        },
        {
            "Name": "Urban",
            "Confidence": 99.99982,
            "Categories": [
                "Colors and Visual Composition"
            ]
        }
    ]
}

def expand_aliases(inferenceOutputsWithAliases):
    if LABEL_KEY in inferenceOutputsWithAliases:
        expandInferenceOutputs = []
        for primaryLabelDict in inferenceOutputsWithAliases[LABEL_KEY]:
            if ALIASES_KEY in primaryLabelDict:
```

```

        for alias in primaryLabelDict[ALIASES_KEY]:
            aliasLabelDict = deepcopy(primaryLabelDict)
            aliasLabelDict[NAME_KEY] = alias[NAME_KEY]
            del aliasLabelDict[ALIASES_KEY]
            if INSTANCE_KEY in aliasLabelDict:
                del aliasLabelDict[INSTANCE_KEY]
            expandInferenceOutputs.append(aliasLabelDict)

    inferenceOutputsWithAliases[LABEL_KEY].extend(expandInferenceOutputs)

return inferenceOutputsWithAliases

if __name__ == "__main__":

    outputWithExpandAliases = expand_aliases(EXAMPLE_INFERENCE_OUTPUT)
    print(outputWithExpandAliases)

```

Di bawah ini adalah contoh respons yang diubah:

```

#Output example after the transformation
{
  "Labels": [
    {
      "Name": "Mobile Phone",
      "Confidence": 97.530106,
      "Categories": [
        {
          "Name": "Technology and Computing"
        }
      ],
      "Aliases": [
        {
          "Name": "Cell Phone"
        }
      ],
      "Instances": [
        {
          "BoundingBox": {
            "Height": 0.1549897,
            "Width": 0.07747964,
            "Top": 0.50858885,

```

```
        "Left":0.00018205095
      },
      "Confidence":98.401276
    }
  ]
},
{
  "Name": "Cell Phone",
  "Confidence": 97.530106,
  "Categories": [
    {
      "Name": "Technology and Computing"
    }
  ],
  "Instances":[]
},
{
  "Name": "Urban",
  "Confidence": 99.99982,
  "Categories": [
    "Colors and Visual Composition"
  ]
}
]
```

Mendeteksi label dalam video

Amazon Rekognition Video dapat mendeteksi label (objek dan konsep), dan waktu label terdeteksi, dalam video. Untuk contoh kode SDK, lihat [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#). AWS CLI Sebagai contoh, lihat [Menganalisis video dengan AWS Command Line Interface](#).

Deteksi label dengan Amazon Rekognition Video adalah operasi tidak sinkron. Untuk memulai deteksi label dalam video, hubungi [StartLabelDeteksi](#).

Amazon Rekognition Video menerbitkan status penyelesaian analisis video ke topik Amazon Simple Notification Service. Jika analisis video berhasil, hubungi [GetLabelDeteksi](#) untuk mendapatkan label yang terdeteksi. Untuk informasi tentang memanggil operasi API analisis video, lihat [Memanggil operasi Amazon Rekognition Video](#).

StartLabelPermintaan Deteksi

Contoh berikut adalah permintaan untuk `StartLabelDetection` operasi. Anda menyediakan `StartLabelDetection` operasi dengan video yang disimpan dalam ember Amazon S3. Dalam permintaan contoh JSON, bucket Amazon S3 dan nama video ditentukan, bersama `MinConfidence` dengan, `FeaturesSettings`, dan `NotificationChannel`

`MinConfidence` adalah keyakinan minimum yang harus dimiliki Video Rekognition Amazon dalam keakuratan label yang terdeteksi, atau kotak pembatas instance (jika terdeteksi), agar dapat dikembalikan dalam respons.

Dengan `Features`, Anda dapat menentukan bahwa Anda ingin `GENERAL_LABELS` dikembalikan sebagai bagian dari respons.

Dengan `Settings`, Anda dapat memfilter item yang dikembalikan untuk `GENERAL_LABELS`. Untuk label, Anda dapat menggunakan filter inklusif dan eksklusif. Anda juga dapat memfilter berdasarkan label khusus, label individual atau berdasarkan kategori label:

- `LabelInclusionFilters`- Digunakan untuk menentukan label mana yang ingin Anda sertakan dalam respons
- `LabelExclusionFilters`- Digunakan untuk menentukan label mana yang ingin Anda kecualikan dari respons.
- `LabelCategoryInclusionFilters`- Digunakan untuk menentukan kategori label mana yang ingin Anda sertakan dalam respons.
- `LabelCategoryExclusionFilters`- Digunakan untuk menentukan kategori label mana yang ingin Anda kecualikan dari respons.

Anda juga dapat menggabungkan filter inklusif dan eksklusif sesuai dengan kebutuhan Anda, tidak termasuk beberapa label atau kategori dan termasuk yang lain.

`NotificationChannel` adalah ARN dari topik Amazon SNS yang Anda inginkan agar Video Rekognition Amazon mempublikasikan status penyelesaian operasi deteksi label. Jika Anda menggunakan kebijakan `AmazonRekognitionServiceRole` izin, maka topik Amazon SNS harus memiliki nama topik yang dimulai dengan `Rekognition`.

Berikut ini adalah contoh `StartLabelDetection` permintaan dalam bentuk JSON, termasuk filter:

```
{
  "ClientRequestToken": "5a6e690e-c750-460a-9d59-c992e0ec8638",
```

```
"JobTag": "5a6e690e-c750-460a-9d59-c992e0ec8638",
"Video": {
  "S3Object": {
    "Bucket": "bucket",
    "Name": "video.mp4"
  }
},
"Features": ["GENERAL_LABELS"],
"MinConfidence": 75,
"Settings": {
  "GeneralLabels": {
    "LabelInclusionFilters": ["Cat", "Dog"],
    "LabelExclusionFilters": ["Tiger"],
    "LabelCategoryInclusionFilters": ["Animals and Pets"],
    "LabelCategoryExclusionFilters": ["Popular Landmark"]
  }
},
"NotificationChannel": {
  "RoleArn": "arn:aws:iam::012345678910:role/SNSAccessRole",
  "SNSTopicArn": "arn:aws:sns:us-east-1:012345678910:notification-topic",
}
}
```

GetLabelDetection Respon Operasi

GetLabelDetection mengembalikan array (Labels) yang berisi informasi tentang label yang terdeteksi dalam video. Array dapat diurutkan berdasarkan waktu atau oleh label yang terdeteksi saat menentukan SortBy parameter. Anda juga dapat memilih bagaimana item respons digabungkan dengan menggunakan parameter. AggregateBy

Contoh berikut adalah respons JSON dari GetLabelDetection. Dalam respons, perhatikan hal berikut:

- Urutkan sesuai urutan - array label yang dikembalikan diurutkan berdasarkan waktu. Untuk mengurutkan berdasarkan label, tentukan NAME dalam parameter input SortBy untuk GetLabelDetection. Jika label muncul beberapa kali dalam video, akan ada beberapa contoh elemen () [LabelDetection](#). Urutan oder default adalahTIMESTAMP, sedangkan urutan urutan sekunder adalahNAME.
- Informasi label - Elemen LabelDetection array berisi objek ([Label](#)), yang pada gilirannya berisi nama label dan kepercayaan Amazon Rekognition dalam keakuratan label yang terdeteksi.

Sebuah `Label` objek juga mencakup taksonomi hierarkis label dan informasi kotak pembatas untuk label umum. `Timestamp` adalah waktu label terdeteksi, didefinisikan sebagai jumlah milidetik yang berlalu sejak awal video.

Informasi tentang Kategori atau Alias apa pun yang terkait dengan label juga dikembalikan. Untuk hasil yang dikumpulkan berdasarkan video `SEGMENTSStartTimeMillis`, `DurationMillis` struktur `EndTimeMillis`, dan dikembalikan, yang menentukan waktu mulai, waktu akhir, dan durasi segmen masing-masing.

- Agregasi - Menentukan bagaimana hasil dikumpulkan ketika dikembalikan. Defaultnya adalah agregat dengan `TIMESTAMPS`. Anda juga dapat memilih untuk agregat menurut `SEGMENTS`, yang mengumpulkan hasil melalui jendela waktu. Jika digabungkan dengan `SEGMENTS`, informasi tentang instance yang terdeteksi dengan kotak pembatas tidak dikembalikan. Hanya label yang terdeteksi selama segmen yang dikembalikan.
- Informasi halaman - Contoh menunjukkan satu halaman informasi deteksi label. Anda dapat menentukan berapa banyak objek `LabelDetection` yang akan dikembalikan dalam parameter input `MaxResults` untuk `GetLabelDetection`. Jika ada lebih banyak hasil daripada `MaxResults`, `GetLabelDetection` mengembalikan token (`NextToken`) yang digunakan untuk mendapatkan halaman hasil berikutnya. Untuk informasi selengkapnya, lihat [Mendapatkan hasil analisis Amazon Rekognition Video](#).
- Informasi video - Respons mencakup informasi tentang format video (`VideoMetadata`) di setiap halaman informasi yang dikembalikan oleh `GetLabelDetection`.

Berikut ini adalah contoh `GetLabelDetection` respons dalam bentuk JSON dengan agregasi oleh `TIMESTAMPS`:

```
{
  "JobStatus": "SUCCEEDED",
  "LabelModelVersion": "3.0",
  "Labels": [
    {
      "Timestamp": 1000,
      "Label": {
        "Name": "Car",
        "Categories": [
          {
            "Name": "Vehicles and Automotive"
          }
        ]
      }
    }
  ],
}
```

```
    "Aliases": [
      {
        "Name": "Automobile"
      }
    ],
    "Parents": [
      {
        "Name": "Vehicle"
      }
    ],
    "Confidence": 99.9364013671875, // Classification confidence
    "Instances": [
      {
        "BoundingBox": {
          "Width": 0.26779675483703613,
          "Height": 0.8562285900115967,
          "Left": 0.3604024350643158,
          "Top": 0.09245597571134567
        },
        "Confidence": 99.9364013671875 // Detection confidence
      }
    ]
  },
  {
    "Timestamp": 1000,
    "Label": {
      "Name": "Cup",
      "Categories": [
        {
          "Name": "Kitchen and Dining"
        }
      ],
      "Aliases": [
        {
          "Name": "Mug"
        }
      ],
      "Parents": [],
      "Confidence": 99.9364013671875, // Classification confidence
      "Instances": [
        {
          "BoundingBox": {
            "Width": 0.26779675483703613,
```

```
        "Height": 0.8562285900115967,
        "Left": 0.3604024350643158,
        "Top": 0.09245597571134567
    },
    "Confidence": 99.9364013671875 // Detection confidence
}
]
}
},
{
    "Timestamp": 2000,
    "Label": {
        "Name": "Kangaroo",
        "Categories": [
            {
                "Name": "Animals and Pets"
            }
        ],
        "Aliases": [
            {
                "Name": "Wallaby"
            }
        ],
        "Parents": [
            {
                "Name": "Mammal"
            }
        ],
        "Confidence": 99.9364013671875,
        "Instances": [
            {
                "BoundingBox": {
                    "Width": 0.26779675483703613,
                    "Height": 0.8562285900115967,
                    "Left": 0.3604024350643158,
                    "Top": 0.09245597571134567,
                },
                "Confidence": 99.9364013671875
            }
        ]
    }
},
{
    "Timestamp": 4000,
```

```
    "Label": {
      "Name": "Bicycle",
      "Categories": [
        {
          "Name": "Hobbies and Interests"
        }
      ],
      "Aliases": [
        {
          "Name": "Bike"
        }
      ],
      "Parents": [
        {
          "Name": "Vehicle"
        }
      ],
      "Confidence": 99.9364013671875,
      "Instances": [
        {
          "BoundingBox": {
            "Width": 0.26779675483703613,
            "Height": 0.8562285900115967,
            "Left": 0.3604024350643158,
            "Top": 0.09245597571134567
          },
          "Confidence": 99.9364013671875
        }
      ]
    }
  ],
  "VideoMetadata": {
    "ColorRange": "FULL",
    "DurationMillis": 5000,
    "Format": "MP4",
    "FrameWidth": 1280,
    "FrameHeight": 720,
    "FrameRate": 24
  }
}
```

Berikut ini adalah contoh GetLabelDetection respon dalam bentuk JSON dengan agregasi oleh SEGMENTS:

```
{
  "JobStatus": "SUCCEEDED",
  "LabelModelVersion": "3.0",
  "Labels": [
    {
      "StartTimestampMillis": 225,
      "EndTimestampMillis": 3578,
      "DurationMillis": 3353,
      "Label": {
        "Name": "Car",
        "Categories": [
          {
            "Name": "Vehicles and Automotive"
          }
        ],
        "Aliases": [
          {
            "Name": "Automobile"
          }
        ],
        "Parents": [
          {
            "Name": "Vehicle"
          }
        ],
        "Confidence": 99.9364013671875 // Maximum confidence score for Segment
mode
      }
    },
    {
      "StartTimestampMillis": 7578,
      "EndTimestampMillis": 12371,
      "DurationMillis": 4793,
      "Label": {
        "Name": "Kangaroo",
        "Categories": [
          {
            "Name": "Animals and Pets"
          }
        ],
      },
    }
  ]
}
```

```
        "Aliases": [
          {
            "Name": "Wallaby"
          }
        ],
        "Parents": [
          {
            "Name": "Mammal"
          }
        ],
        "Confidence": 99.9364013671875
      }
    },
    {
      "StartTimestampMillis": 22225,
      "EndTimestampMillis": 22578,
      "DurationMillis": 2353,
      "Label": {
        "Name": "Bicycle",
        "Categories": [
          {
            "Name": "Hobbies and Interests"
          }
        ],
        "Aliases": [
          {
            "Name": "Bike"
          }
        ],
        "Parents": [
          {
            "Name": "Vehicle"
          }
        ],
        "Confidence": 99.9364013671875
      }
    }
  ],
  "VideoMetadata": {
    "ColorRange": "FULL",
    "DurationMillis": 5000,
    "Format": "MP4",
    "FrameWidth": 1280,
    "FrameHeight": 720,
```



```
    "FrameRate": 24
  }
}
```

Mengubah Respon GetLabelDetection

Saat mengambil hasil dengan operasi GetLabelDetection API, Anda mungkin memerlukan struktur respons untuk meniru struktur respons API yang lebih lama, di mana label utama dan alias terkandung dalam daftar yang sama.

Contoh respons JSON yang ditemukan di bagian sebelumnya menampilkan bentuk respons API saat ini dari GetLabelDetection

Contoh berikut menunjukkan respons sebelumnya dari GetLabelDetection API:

```
{
  "Labels": [
    {
      "Timestamp": 0,
      "Label": {
        "Instances": [],
        "Confidence": 60.51791763305664,
        "Parents": [],
        "Name": "Leaf"
      }
    },
    {
      "Timestamp": 0,
      "Label": {
        "Instances": [],
        "Confidence": 99.53411102294922,
        "Parents": [],
        "Name": "Human"
      }
    },
    {
      "Timestamp": 0,
      "Label": {
        "Instances": [
          {
            "BoundingBox": {
              "Width": 0.11109819263219833,
              "Top": 0.08098889887332916,
```

```
        "Left": 0.8881205320358276,
        "Height": 0.9073750972747803
    },
    "Confidence": 99.5831298828125
},
{
    "BoundingBox": {
        "Width": 0.1268676072359085,
        "Top": 0.14018426835536957,
        "Left": 0.0003282368124928324,
        "Height": 0.7993982434272766
    },
    "Confidence": 99.46029663085938
}
],
"Confidence": 99.63411102294922,
"Parents": [],
"Name": "Person"
}
},
.
.
.
{
    "Timestamp": 166,
    "Label": {
        "Instances": [],
        "Confidence": 73.6471176147461,
        "Parents": [
            {
                "Name": "Clothing"
            }
        ],
        "Name": "Sleeve"
    }
}
},
"LabelModelVersion": "2.0",
"JobStatus": "SUCCEEDED",
"VideoMetadata": {
    "Format": "QuickTime / MOV",
    "FrameRate": 23.976024627685547,
```

```
    "Codec": "h264",
    "DurationMillis": 5005,
    "FrameHeight": 674,
    "FrameWidth": 1280
  }
}
```

Jika diperlukan, Anda dapat mengubah respons saat ini untuk mengikuti format respons yang lebih lama. Anda dapat menggunakan kode contoh berikut untuk mengubah respons API terbaru ke struktur respons API sebelumnya:

```
from copy import deepcopy

VIDEO_LABEL_KEY = "Labels"
LABEL_KEY = "Label"
ALIASES_KEY = "Aliases"
INSTANCE_KEY = "Instances"
NAME_KEY = "Name"

#Latest API response sample for AggregatedBy SEGMENTS
EXAMPLE_SEGMENT_OUTPUT = {
    "Labels": [
        {
            "Timestamp": 0,
            "Label":{
                "Name": "Person",
                "Confidence": 97.530106,
                "Parents": [],
                "Aliases": [
                    {
                        "Name": "Human"
                    },
                ],
                "Categories": [
                    {
                        "Name": "Person Description"
                    }
                ],
            },
            "StartTimestampMillis": 0,
            "EndTimestampMillis": 500666,
            "DurationMillis": 500666
        },
    ],
}
```

```
{
  "Timestamp": 6400,
  "Label": {
    "Name": "Leaf",
    "Confidence": 89.77790069580078,
    "Parents": [
      {
        "Name": "Plant"
      }
    ],
    "Aliases": [],
    "Categories": [
      {
        "Name": "Plants and Flowers"
      }
    ]
  },
  "StartTimestampMillis": 6400,
  "EndTimestampMillis": 8200,
  "DurationMillis": 1800
},
]
```

#Output example after the transformation for AggregatedBy SEGMENTS

```
EXPECTED_EXPANDED_SEGMENT_OUTPUT = {
  "Labels": [
    {
      "Timestamp": 0,
      "Label": {
        "Name": "Person",
        "Confidence": 97.530106,
        "Parents": [],
        "Aliases": [
          {
            "Name": "Human"
          }
        ],
        "Categories": [
          {
            "Name": "Person Description"
          }
        ]
      }
    ]
  },
}
```

```
    },
    "StartTimestampMillis": 0,
    "EndTimestampMillis": 500666,
    "DurationMillis": 500666
  },
  {
    "Timestamp": 6400,
    "Label": {
      "Name": "Leaf",
      "Confidence": 89.77790069580078,
      "Parents": [
        {
          "Name": "Plant"
        }
      ],
      "Aliases": [],
      "Categories": [
        {
          "Name": "Plants and Flowers"
        }
      ]
    },
    "StartTimestampMillis": 6400,
    "EndTimestampMillis": 8200,
    "DurationMillis": 1800
  },
  {
    "Timestamp": 0,
    "Label": {
      "Name": "Human",
      "Confidence": 97.530106,
      "Parents": [],
      "Categories": [
        {
          "Name": "Person Description"
        }
      ]
    },
    "StartTimestampMillis": 0,
    "EndTimestampMillis": 500666,
    "DurationMillis": 500666
  },
]
```

```
}

#Latest API response sample for AggregatedBy TIMESTAMPS
EXAMPLE_TIMESTAMP_OUTPUT = {
  "Labels": [
    {
      "Timestamp": 0,
      "Label": {
        "Name": "Person",
        "Confidence": 97.530106,
        "Instances": [
          {
            "BoundingBox": {
              "Height": 0.1549897,
              "Width": 0.07747964,
              "Top": 0.50858885,
              "Left": 0.00018205095
            },
            "Confidence": 97.530106
          },
        ],
        "Parents": [],
        "Aliases": [
          {
            "Name": "Human"
          },
        ],
        "Categories": [
          {
            "Name": "Person Description"
          }
        ],
      },
    },
    {
      "Timestamp": 6400,
      "Label": {
        "Name": "Leaf",
        "Confidence": 89.77790069580078,
        "Instances": [],
        "Parents": [
          {
            "Name": "Plant"
          }
        ],
      },
    },
  ],
}
```

```

    ],
    "Aliases": [],
    "Categories": [
      {
        "Name": "Plants and Flowers"
      }
    ],
  },
],
}

```

#Output example after the transformation for AggregatedBy TIMESTAMPS

```

EXPECTED_EXPANDED_TIMESTAMP_OUTPUT = {
  "Labels": [
    {
      "Timestamp": 0,
      "Label": {
        "Name": "Person",
        "Confidence": 97.530106,
        "Instances": [
          {
            "BoundingBox": {
              "Height": 0.1549897,
              "Width": 0.07747964,
              "Top": 0.50858885,
              "Left": 0.00018205095
            },
            "Confidence": 97.530106
          }
        ],
        "Parents": [],
        "Aliases": [
          {
            "Name": "Human"
          }
        ],
        "Categories": [
          {
            "Name": "Person Description"
          }
        ],
      },
    ],
  },
}

```

```

    {
      "Timestamp": 6400,
      "Label": {
        "Name": "Leaf",
        "Confidence": 89.77790069580078,
        "Instances": [],
        "Parents": [
          {
            "Name": "Plant"
          }
        ],
        "Aliases": [],
        "Categories": [
          {
            "Name": "Plants and Flowers"
          }
        ],
      },
    },
    {
      "Timestamp": 0,
      "Label": {
        "Name": "Human",
        "Confidence": 97.530106,
        "Parents": [],
        "Categories": [
          {
            "Name": "Person Description"
          }
        ],
      },
    },
  ]
}

```

```
def expand_aliases(inferenceOutputsWithAliases):
```

```

    if VIDEO_LABEL_KEY in inferenceOutputsWithAliases:
        expandInferenceOutputs = []
        for segmentLabelDict in inferenceOutputsWithAliases[VIDEO_LABEL_KEY]:
            primaryLabelDict = segmentLabelDict[LABEL_KEY]
            if ALIASES_KEY in primaryLabelDict:
                for alias in primaryLabelDict[ALIASES_KEY]:
                    aliasLabelDict = deepcopy(segmentLabelDict)

```



```
aliasLabelDict[LABEL_KEY][NAME_KEY] = alias[NAME_KEY]
del aliasLabelDict[LABEL_KEY][ALIASES_KEY]
if INSTANCE_KEY in aliasLabelDict[LABEL_KEY]:
    del aliasLabelDict[LABEL_KEY][INSTANCE_KEY]
expandInferenceOutputs.append(aliasLabelDict)

inferenceOutputsWithAliases[VIDEO_LABEL_KEY].extend(expandInferenceOutputs)

return inferenceOutputsWithAliases

if __name__ == "__main__":

    segmentOutputWithExpandAliases = expand_aliases(EXAMPLE_SEGMENT_OUTPUT)
    assert segmentOutputWithExpandAliases == EXPECTED_EXPANDED_SEGMENT_OUTPUT

    timestampOutputWithExpandAliases = expand_aliases(EXAMPLE_TIMESTAMP_OUTPUT)
    assert timestampOutputWithExpandAliases == EXPECTED_EXPANDED_TIMESTAMP_OUTPUT
```

Mendeteksi label dalam streaming acara video

Anda dapat menggunakan Amazon Rekognition Video untuk mendeteksi label dalam streaming video. Untuk melakukan ini, Anda membuat prosesor aliran ([CreateStreamProcessor](#)) untuk memulai dan mengelola analisis streaming video.

Amazon Rekognition Video menggunakan Amazon Kinesis Video Streams untuk menerima dan memproses video streaming. Saat Anda membuat prosesor aliran, Anda memilih apa yang Anda inginkan untuk dideteksi oleh prosesor aliran. Anda dapat memilih orang, paket dan hewan peliharaan, atau orang dan paket. Hasil analisis dikeluarkan ke bucket Amazon S3 Anda dan notifikasi Amazon SNS. Perhatikan bahwa Amazon Rekognition Video mendeteksi keberadaan seseorang dalam video, tetapi tidak mendeteksi apakah orang tersebut adalah individu tertentu. Untuk mencari wajah dari koleksi dalam video streaming, lihat [the section called “Mencari wajah dalam koleksi dalam streaming video”](#).

Untuk menggunakan Amazon Rekognition Video dengan streaming video, aplikasi Anda memerlukan hal berikut:

- Aliran video Kinesis untuk mengirimkan video streaming ke Amazon Rekognition Video. Untuk informasi selengkapnya, lihat [Panduan Developer Amazon Kinesis Video Streams](#).

- Sebuah pemroses aliran Amazon Rekognition Video untuk mengelola analisis video streaming. Untuk informasi selengkapnya, lihat [Ikhtisar operasi prosesor aliran Video Rekognition Amazon](#).
- Bucket Amazon S3. Amazon Rekognition Video menerbitkan output sesi ke bucket S3. Outputnya mencakup bingkai gambar tempat seseorang atau objek yang diminati terdeteksi untuk pertama kalinya. Anda harus menjadi pemilik ember S3.
- Topik Amazon SNS bahwa Amazon Rekognition Video menerbitkan peringatan cerdas dan end-of-session ringkasan untuk.

Topik

- [Mempersiapkan Amazon Rekognition Video Amazon dan sumber daya Amazon Kinesis](#)
- [Operasi deteksi label untuk streaming acara video](#)

Mempersiapkan Amazon Rekognition Video Amazon dan sumber daya Amazon Kinesis

Prosedur berikut menjelaskan langkah-langkah yang Anda ambil untuk menyediakan aliran video Kinesis dan sumber daya lain yang digunakan untuk mendeteksi label dalam video streaming.

Prasyarat

Untuk menjalankan prosedur ini, AWS SDK for Java harus dipasang. Untuk informasi selengkapnya, lihat [Memulai dengan Amazon Rekognition](#). Yang AWS SDK yang Anda gunakan memerlukan izin akses ke Amazon Rekognition API. Untuk informasi selengkapnya, lihat [Tindakan yang Ditetapkan oleh Amazon Rekognition](#) di Panduan Pengguna IAM.

Untuk mendeteksi label dalam aliran video (AWS SDK)

1. Buat bucket Amazon S3. Perhatikan nama bucket dan awalan kunci apa pun yang ingin Anda gunakan. Anda menggunakan informasi ini nanti.
2. Buat topik Amazon SNS. Anda dapat menggunakannya untuk menerima pemberitahuan ketika objek yang menarik pertama kali terdeteksi dalam aliran video. Perhatikan Amazon Resource Name (ARN) untuk topik tersebut. Untuk informasi lebih lanjut, lihat [Membuat topik Amazon SNS](#) dalam panduan pengembang Amazon SNS.
3. Berlangganan titik akhir ke topik Amazon SNS. Untuk informasi lebih lanjut, lihat [Berlangganan topik Amazon SNS](#) dalam panduan pengembang Amazon SNS.
4. [Membuat aliran video Kinesis](#) dan perhatikan Amazon Resource Name (ARN) stream.

5. Jika belum melakukannya, buat peran layanan IAM untuk memberikan akses Amazon Rekognition Video ke aliran video Kinesis, bucket S3, dan topik Amazon SNS Anda. Untuk informasi selengkapnya, lihat [Memberikan akses untuk prosesor aliran deteksi label](#).

Anda kemudian dapat [membuat prosesor aliran deteksi label](#) dan [mulai prosesor aliran](#) menggunakan nama prosesor stream yang Anda pilih.

Note

Mulai prosesor streaming hanya setelah Anda memverifikasi bahwa Anda dapat menelan media ke dalam aliran video Kinesis.

Orientasi dan pengaturan kamera

Amazon Rekognition Video Streaming Video Events dapat mendukung semua kamera yang didukung oleh Kinesis Video Streams. Untuk hasil terbaik, sebaiknya letakkan kamera antara 0 hingga 45 derajat dari tanah. Kamera harus berada dalam posisi tegak kanonik. Misalnya, jika ada seseorang dalam bingkai, orang tersebut harus berorientasi secara vertikal, dan kepala orang tersebut harus lebih tinggi dalam bingkai daripada kaki.

Memberikan akses untuk prosesor aliran deteksi label

Anda menggunakan AWS Identity and Access Management Peran layanan (IAM) untuk memberikan akses baca Amazon Rekognition Video ke aliran video Kinesis. Untuk melakukannya, gunakan peran IAM untuk memberikan akses Amazon Rekognition Video ke bucket Amazon S3 Anda dan ke topik Amazon SNS.

Anda dapat membuat kebijakan izin yang memungkinkan akses Amazon Rekognition Video ke topik Amazon SNS, bucket Amazon S3, dan aliran video Kinesis yang ada. Untuk step-by-step menggunakan prosedur AWS CLI, lihat [the section called “AWS CLI perintah untuk mengatur peran IAM deteksi label”](#).

Untuk memberikan Amazon Rekognition Video akses ke sumber daya untuk deteksi label

1. [Buat kebijakan izin baru dengan editor kebijakan IAM JSON](#), dan gunakan kebijakan berikut. Gantikan `vs-stream-named` dengan nama aliran video Kinesis, `topic-arn` dengan Amazon Resource Name (ARN) dari topik Amazon SNS yang ingin Anda gunakan, dan `bucket-named` dengan nama bucket Amazon S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "KinesisVideoPermissions",
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:GetDataEndpoint",
        "kinesisvideo:GetMedia"
      ],
      "Resource": [
        "arn:aws:kinesisvideo::stream/kvs-stream-name/*"
      ]
    },
    {
      "Sid": "SNSPermissions",
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "arn:aws:sns::sns-topic-name"
      ]
    },
    {
      "Sid": "S3Permissions",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name/*"
      ]
    }
  ]
}
```

2. [Buat peran layanan IAM](#), atau perbarui peran layanan IAM yang ada. Gunakan informasi berikut ini untuk membuat peran layanan IAM:

1. Pilih Rekognition untuk nama layanan.
2. Pilih Rekognition untuk kasus penggunaan peran layanan.
3. Lampirkan kebijakan izin yang Anda buat pada langkah 1.
3. Catat ARN peran layanan. Anda membutuhkannya untuk membuat prosesor aliran sebelum Anda melakukan operasi analisis video.
4. (Opsional) Jika Anda menggunakan milik Anda sendiri AWS KMS kunci untuk mengenkripsi data yang dikirim ke bucket S3 Anda, Anda harus menambahkan pernyataan berikut dengan peran IAM. (Ini adalah peran IAM yang Anda buat untuk kebijakan kunci, yang sesuai dengan kunci yang dikelola pelanggan yang ingin Anda gunakan.)

```

    {
        "Sid": "Allow use of the key by label detection Role",
        "Effect": "Allow",
        "Principal": {
            "AWS":
"arn:aws:iam:::role/REPLACE_WITH_LABEL_DETECTION_ROLE_CREATED"
        },
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDataKey*"
        ],
        "Resource": "*"
    }

```

AWS CLI perintah untuk mengatur peran IAM deteksi label

Jika Anda belum melakukannya, atur dan konfigurasi AWS CLI dengan kredensialmu.

Masukkan perintah berikut ke dalam AWS CLI untuk mengatur peran IAM dengan izin yang diperlukan untuk deteksi label.

1. `export IAM_ROLE_NAME=labels-test-role`
2. `export AWS_REGION=us-east-1`
3. Buat file kebijakan hubungan kepercayaan (misalnya, `assume-role-rekognition.json`) dengan konten berikut.

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

4. `aws iam create-role --role-name $IAM_ROLE_NAME --assume-role-policy-document file:///path-to-assume-role-rekognition.json --region $AWS_REGION`

5. `aws iam attach-role-policy --role-name $IAM_ROLE_NAME --policy-arn "arn:aws:iam::aws:policy/service-role/AmazonRekognitionServiceRole" --region $AWS_REGION`

6. Jika nama topik SNS yang ingin Anda terima pemberitahuan tidak dimulai dengan "AmazonRekognition" awalan, tambahkan kebijakan berikut:

```
aws iam attach-role-policy --role-name $IAM_ROLE_NAME --policy-arn "arn:aws:iam::aws:policy/AmazonSNSFullAccess" --region $AWS_REGION
```

7. Jika Anda menggunakan kunci AWS KMS Anda sendiri untuk mengenkripsi data yang dikirim ke bucket Amazon S3 Anda, perbarui kebijakan kunci kunci kunci yang dikelola pelanggan yang ingin Anda gunakan.

a. Buat file `kms_key_policy.json` yang berisi konten berikut:

```
{
  "Sid": "Allow use of the key by label detection Role",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam:::role/REPLACE_WITH_IAM_ROLE_NAME_CREATED"
  },
  "Action": [
```

```
"kms:Encrypt",
"kms:GenerateDataKey*"
],
"Resource": "*"
}
```

- b. `export KMS_KEY_ID=labels-kms-key-id`. Ganti `KMS_KEY_ID` dengan ID kunci KMS yang Anda buat.
- c. `aws kms put-key-policy --policy-name default --key-id $KMS_KEY_ID --policy file://path-to-kms-key-policy.json`

Operasi deteksi label untuk streaming acara video

Amazon Rekognition Video dapat mendeteksi orang atau objek yang relevan dalam video streaming dan memberi tahu Anda saat terdeteksi. Saat Anda membuat prosesor aliran deteksi label, pilih label apa yang ingin dideteksi Amazon Rekognition Video. Ini bisa berupa orang, paket dan hewan peliharaan, atau orang, paket, dan hewan peliharaan. Pilih hanya label tertentu yang ingin Anda deteksi. Dengan begitu, satu-satunya label yang relevan membuat notifikasi. Anda dapat mengonfigurasi opsi untuk menentukan kapan harus menyimpan informasi video, dan kemudian melakukan pemrosesan tambahan berdasarkan label yang terdeteksi dalam bingkai.

Setelah menyiapkan sumber daya, proses untuk mendeteksi label dalam video streaming adalah sebagai berikut:

1. Buat prosesor aliran
2. Mulai prosesor aliran
3. Jika objek yang diminati terdeteksi, Anda menerima notifikasi Amazon SNS untuk kejadian pertama dari setiap objek yang diminati.
4. Prosesor aliran berhenti saat waktu yang ditentukan `MaxDurationInSeconds` selesai.
5. Anda menerima pemberitahuan Amazon SNS akhir dengan ringkasan peristiwa.
6. Amazon Rekognition Video menerbitkan ringkasan sesi terperinci ke bucket S3 Anda.

Topik

- [Membuat prosesor aliran deteksi label Amazon Rekognition Video](#)
- [Memulai prosesor aliran deteksi label Amazon Rekognition Video](#)

- [Menganalisis hasil deteksi label](#)

Membuat prosesor aliran deteksi label Amazon Rekognition Video

Sebelum Anda dapat menganalisis video streaming, Anda membuat pemroses aliran Amazon Rekognition Video ([CreateStreamProcessor](#)).

Jika Anda ingin membuat prosesor aliran untuk mendeteksi label yang menarik dan orang, berikan sebagai masukan aliran video Kinesis (Input), informasi bucket Amazon S3 (Output), dan topik Amazon SNS ARN ([StreamProcessorNotificationChannel](#)). Anda juga dapat memberikan ID kunci KMS untuk mengenkripsi data yang dikirim ke bucket S3 Anda. Anda menentukan apa yang ingin Anda deteksiSettings, seperti orang, paket dan orang, atau hewan peliharaan, orang, dan paket. Anda juga dapat menentukan di mana dalam bingkai yang ingin dipantau Amazon RekognitionRegionsOfInterest. Berikut ini adalah contoh JSON untuk permintaan [CreateStreamProcessor](#).

```
{
  "DataSharingPreference": { "OptIn":TRUE
  },
  "Input": {
    "KinesisVideoStream": {
      "Arn": "arn:aws:kinesisvideo:us-east-1:nnnnnnnnnnnn:stream/muh_video_stream/
nnnnnnnnnnnn"
    }
  },
  "KmsKeyId": "muhkey",
  "Name": "muh-default_stream_processor",
  "Output": {
    "S3Destination": {
      "Bucket": "s3bucket",
      "KeyPrefix": "s3prefix"
    }
  },
  "NotificationChannel": {
    "SNSTopicArn": "arn:aws:sns:us-east-2:nnnnnnnnnnnn:MyTopic"
  },
  "RoleArn": "arn:aws:iam::nnnnnnnnnn:role/Admin",
  "Settings": {
```



```
"ConnectedHome": {
  "Labels": [
    "PET"
  ]
  "MinConfidence": 80
},
"RegionsOfInterest": [
  {
    "BoundingBox": {
      "Top": 0.11,
      "Left": 0.22,
      "Width": 0.33,
      "Height": 0.44
    }
  },
  {
    "Polygon": [
      {
        "X": 0.11,
        "Y": 0.11
      },
      {
        "X": 0.22,
        "Y": 0.22
      },
      {
        "X": 0.33,
        "Y": 0.33
      }
    ]
  }
]
}
```

Perhatikan bahwa Anda dapat mengubah `MinConfidence` nilai saat Anda menentukan `ConnectedHomeSettings` untuk prosesor stream. `MinConfidence` adalah nilai numerik mulai dari 0 sampai 100 yang menunjukkan seberapa pasti algoritma adalah tentang prediksi. Misalnya, pemberitahuan untuk `person` dengan nilai kepercayaan 90 berarti bahwa algoritma benar-benar yakin bahwa orang tersebut hadir dalam video. Nilai kepercayaan 10 menunjukkan

bahwa mungkin ada seseorang. Anda dapat mengatur `MinConfidence` ke nilai yang diinginkan dari pilihan Anda antara 0 dan 100 tergantung pada seberapa sering Anda ingin diberitahu. Misalnya, jika Anda ingin diberitahu hanya ketika Rekognition benar-benar yakin ada paket dalam bingkai video maka Anda dapat mengatur `MinConfidence` untuk 90.

Secara default, `MinConfidence` diatur ke 50. Jika Anda ingin mengoptimalkan algoritma untuk presisi yang lebih tinggi, maka Anda dapat mengatur `MinConfidence` lebih tinggi dari 50. Anda kemudian menerima lebih sedikit pemberitahuan, tetapi setiap notifikasi lebih dapat diandalkan. Jika Anda ingin mengoptimalkan algoritma untuk penarikan yang lebih tinggi, maka Anda dapat mengatur `MinConfidence` menjadi lebih rendah dari 50 untuk menerima lebih banyak pemberitahuan.

Memulai prosesor aliran deteksi label Amazon Rekognition Video

Anda mulai menganalisis streaming video dengan memanggil [StartStreamProcessor](#) dengan nama pemroses aliran yang Anda tentukan di `CreateStreamProcessor`. Ketika Anda menjalankan `StartStreamProcessor` operasi pada prosesor aliran deteksi label, Anda memasukkan informasi mulai dan berhenti untuk menentukan waktu pemrosesan.

Saat Anda memulai prosesor aliran, status prosesor aliran deteksi label berubah dengan cara berikut:

1. Saat Anda menelepon `StartStreamProcessor`, status prosesor aliran deteksi label beralih dari `STOPPED` atau `FAILED` kepada `STARTING`.
2. Sementara prosesor aliran deteksi label berjalan, itu tetap di `STARTING`.
3. Ketika prosesor aliran deteksi label selesai berjalan, negara menjadi baik `STOPPED` atau `FAILED`.

Yang `StartSelector` menentukan titik awal dalam aliran Kinesis untuk memulai pemrosesan. Anda dapat menggunakan stempel waktu KVS Produser atau nomor Fragmen KVS. Untuk informasi lebih lanjut, lihat [Fragmen](#).

Note

Jika Anda menggunakan stempel waktu KVS Produser, Anda harus memasukkan waktu dalam milidetik.

Yang `StopSelector` menentukan kapan harus berhenti memproses sungai. Anda dapat menentukan jumlah waktu maksimum untuk memproses video. Defaultnya adalah durasi maksimum 10 detik.

Perhatikan bahwa waktu pemrosesan aktual mungkin sedikit lebih lama dari durasi maksimum, tergantung pada ukuran fragmen KVS individu. Jika durasi maksimum telah tercapai atau terlampaui pada akhir fragmen, waktu pemrosesan berhenti.

Berikut ini adalah contoh JSON untuk permintaan `StartStreamProcessor`.

```
{
  "Name": "string",
  "StartSelector": {
    "KVSSStreamStartSelector": {
      "KVSProducerTimestamp": 1655930623123
    },
    "StopSelector": {
      "MaxDurationInSeconds": 11
    }
  }
}
```

Jika prosesor aliran berhasil dimulai, respons HTTP 200 dikembalikan. Tubuh JSON kosong disertakan.

Menganalisis hasil deteksi label

Ada tiga cara Amazon Rekognition Video menerbitkan notifikasi dari prosesor aliran deteksi label: pemberitahuan Amazon SNS untuk peristiwa deteksi objek, pemberitahuan Amazon SNS untuk end-of-session ringkasan, dan laporan bucket Amazon S3 terperinci.

- Pemberitahuan Amazon SNS untuk peristiwa deteksi objek.

Jika label terdeteksi dalam aliran video, Anda menerima notifikasi Amazon SNS untuk peristiwa deteksi objek. Amazon Rekognition menerbitkan notifikasi saat pertama kali seseorang atau objek yang diminati terdeteksi dalam aliran video. Pemberitahuan mencakup informasi seperti jenis label yang terdeteksi, kepercayaan diri, dan tautan ke gambar pahlawan. Mereka juga menyertakan gambar yang dipotong dari orang atau objek yang terdeteksi dan stempel waktu deteksi. Pemberitahuan memiliki format berikut:

```
{"Subject": "Rekognition Stream Processing Event",
  "Message": {
```

```
"inputInformation": {
  "kinesisVideo": {
    "streamArn": string
  }
},
"eventNamespace": {
  "type": "LABEL_DETECTED"
},
"labels": [{
  "id": string,
  "name": "PERSON" | "PET" | "PACKAGE",
  "frameImageUri": string,
  "croppedImageUri": string,
  "videoMapping": {
    "kinesisVideoMapping": {
      "fragmentNumber": string,
      "serverTimestamp": number,
      "producerTimestamp": number,
      "frameOffsetMillis": number
    }
  },
  "boundingBox": {
    "left": number,
    "top": number,
    "height": number,
    "width": number
  }
}],
"eventId": string,
"tags": {
  [string]: string
},
"sessionId": string,
"startStreamProcessorRequest": object
}
}
```

- Amazon SNS end-of-session ringkasan.

Anda juga menerima notifikasi Amazon SNS saat sesi pemrosesan aliran selesai. Notifikasi ini mencantumkan metadata untuk sesi tersebut. Ini termasuk detail seperti durasi aliran yang diproses. Pemberitahuan memiliki format berikut:

```
{"Subject": "Rekognition Stream Processing Event",
  "Message": {
    "inputInformation": {
      "kinesisVideo": {
        "streamArn": string,
        "processedVideoDurationMillis": number
      }
    },
    "eventNamespace": {
      "type": "STREAM_PROCESSING_COMPLETE"
    },
    "streamProcessingResults": {
      "message": string
    },
    "eventId": string,
    "tags": {
      [string]: string
    },
    "sessionId": string,
    "startStreamProcessorRequest": object
  }
}
```

- Laporan bucket Amazon S3.

Amazon Rekognition Video menerbitkan hasil inferensi terperinci dari operasi analisis video ke bucket Amazon S3 yang disediakan di `CreateStreamProcessor` operasi. Hasil ini termasuk bingkai gambar di mana objek yang menarik atau orang terdeteksi untuk pertama kalinya.

Frame tersedia di S3 di jalur berikut: `ObjectKeyPrefix/StreamProcessorName/SessionId/service_determined_unique_path`. Di jalan ini, `LabelKeyPrefix` adalah pelangan yang disediakan argumen opsional, `StreamProcessorName` adalah nama sumber daya prosesor stream, dan `SessionId` adalah ID unik untuk sesi pemrosesan aliran. Ganti ini sesuai dengan situasi Anda.

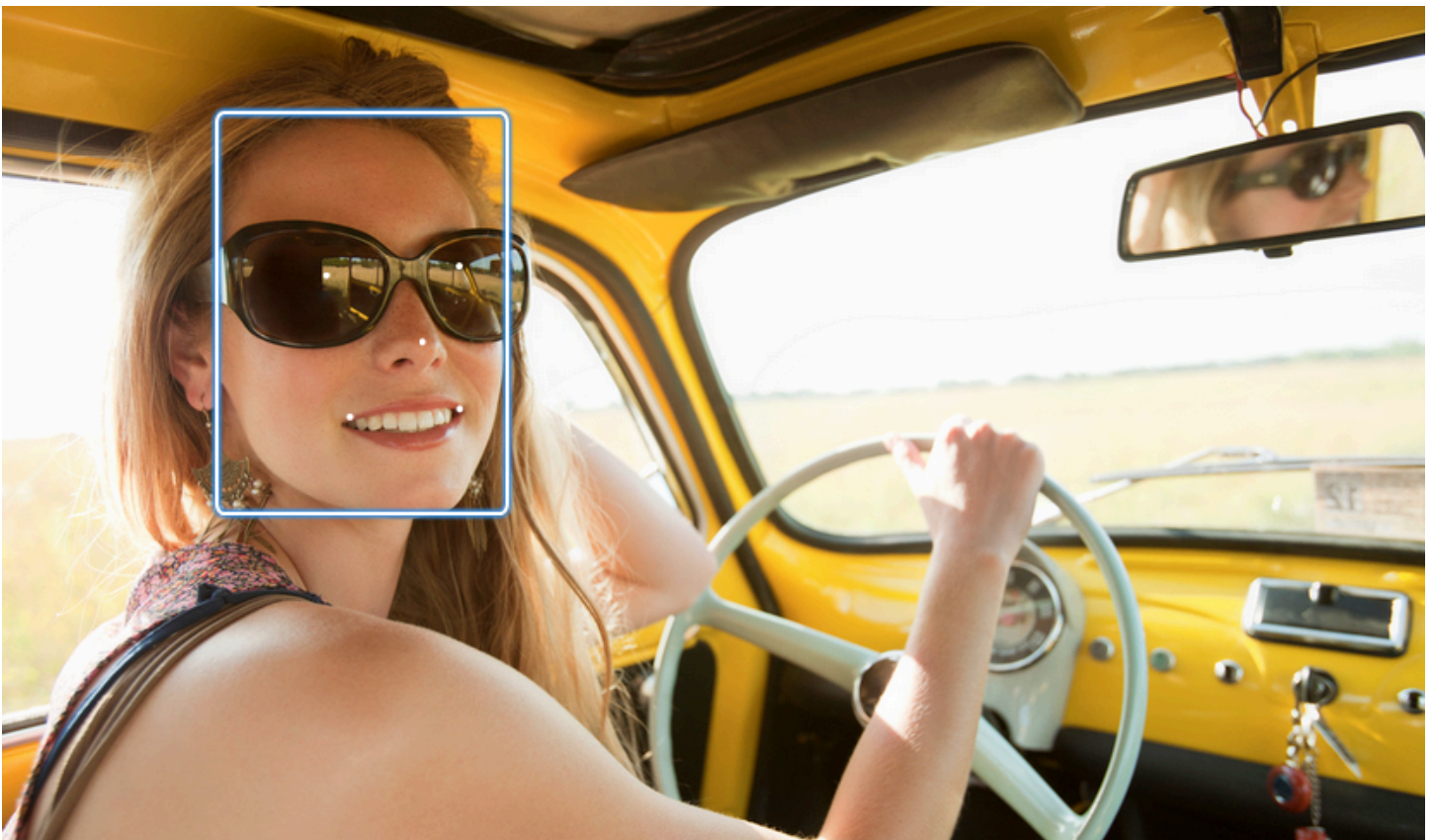
Mendeteksi label kustom

Label Kustom Amazon Rekognition dapat mengidentifikasi objek dan pemandangan dalam citra yang khusus untuk kebutuhan bisnis Anda, seperti logo atau bagian-bagian mesin rekayasa. Untuk informasi selengkapnya, lihat [Apa itu Label Kustom Amazon Rekognition?](#) dalam Panduan Developer Label Kustom Amazon Rekognition.

Mendeteksi dan menganalisis wajah

Amazon Rekognition memberi Anda API yang dapat Anda gunakan untuk mendeteksi dan menganalisis wajah dalam gambar dan video. Bagian ini memberikan gambaran umum tentang operasi non-penyimpanan untuk analisis wajah. Operasi ini mencakup fungsionalitas seperti mendeteksi tengara wajah, menganalisis emosi, dan membandingkan wajah.

Amazon Rekognition dapat mengidentifikasi tengara wajah (misalnya, posisi mata), mendeteksi emosi (misalnya, kebahagiaan atau kesedihan), dan atribut lainnya (misalnya, kehadiran kacamata, oklusi wajah). Ketika wajah terdeteksi, sistem menganalisis atribut wajah dan mengembalikan skor kepercayaan untuk setiap atribut.



Bagian ini berisi contoh untuk operasi gambar dan video.

Untuk informasi selengkapnya tentang penggunaan operasi gambar Rekognition, lihat. [Bekerja dengan citra](#)

Untuk informasi selengkapnya tentang penggunaan operasi video Rekognition, lihat. [Bekerja dengan analisis video tersimpan](#)

Perhatikan bahwa operasi ini adalah operasi non-penyimpanan. Anda dapat menggunakan operasi penyimpanan dan koleksi Wajah untuk menyimpan metadata wajah untuk wajah yang terdeteksi dalam gambar. Nantinya Anda dapat mencari wajah yang tersimpan baik dalam citra dan video. Misalnya, layanan ini memungkinkan pencarian orang tertentu dalam video. Untuk informasi selengkapnya, lihat [Mencari wajah dalam koleksi](#).

Untuk informasi selengkapnya, lihat bagian Wajah di FAQ [Rekognition Amazon](#).

Note

Model deteksi wajah yang digunakan oleh Amazon Rekognition Image dan Amazon Rekognition Video tidak mendukung deteksi wajah dalam karakter kartun/animasi atau entitas bukan manusia. Jika Anda ingin mendeteksi karakter kartun dalam citra atau video, sebaiknya gunakan Label Kustom Amazon Rekognition. Untuk informasi selengkapnya, lihat [Panduan Developer Label Kustom Amazon Rekognition](#).

Topik

- [Gambaran umum deteksi wajah dan perbandingan wajah](#)
- [Pedoman atribut wajah](#)
- [Mendeteksi wajah dalam citra](#)
- [Membandingkan wajah dalam citra](#)
- [Mendeteksi wajah dalam video yang tersimpan](#)

Gambaran umum deteksi wajah dan perbandingan wajah

Amazon Rekognition memberi pengguna akses ke dua aplikasi pembelajaran mesin utama untuk gambar yang berisi wajah: deteksi wajah dan perbandingan wajah. Mereka memberdayakan fitur-fitur penting seperti analisis wajah dan verifikasi identitas, menjadikannya penting untuk berbagai aplikasi mulai dari keamanan hingga organisasi foto pribadi.

Deteksi Wajah

Sistem deteksi wajah menjawab pertanyaan: “Apakah ada wajah dalam gambar ini?” Aspek kunci dari deteksi wajah meliputi:

- Lokasi dan orientasi: Menentukan keberadaan, lokasi, skala, dan orientasi wajah dalam gambar atau bingkai video.
- Atribut wajah: Mendeteksi wajah terlepas dari atribut seperti jenis kelamin, usia, atau rambut wajah.
- Informasi Tambahan: Memberikan detail tentang oklusi wajah dan arah pandangan mata.

Perbandingan Wajah

Sistem perbandingan wajah berfokus pada pertanyaan: “Apakah wajah dalam satu gambar cocok dengan wajah di gambar lain?” Fungsionalitas sistem perbandingan wajah meliputi:

- Prediksi pencocokan wajah: Membandingkan wajah dalam gambar dengan wajah dalam database yang disediakan untuk memprediksi kecocokan.
- Penanganan atribut wajah: Menangani atribut untuk membandingkan wajah terlepas dari ekspresi, rambut wajah, dan usia.

Skor kepercayaan diri dan deteksi yang terlewat

Baik deteksi wajah dan sistem perbandingan wajah menggunakan skor kepercayaan. Skor kepercayaan menunjukkan kemungkinan prediksi, seperti kehadiran wajah atau kecocokan antar wajah. Skor yang lebih tinggi menunjukkan kemungkinan yang lebih besar. Misalnya, kepercayaan 90% menunjukkan probabilitas deteksi atau kecocokan yang lebih tinggi dari 60%.

Jika sistem deteksi wajah tidak mendeteksi wajah dengan benar, atau memberikan prediksi kepercayaan rendah untuk wajah yang sebenarnya, ini adalah deteksi yang terlewat/negatif palsu. Jika sistem salah memprediksi keberadaan wajah pada tingkat kepercayaan tinggi, ini adalah alarm palsu/positif palsu.

Demikian pula, sistem perbandingan wajah mungkin tidak cocok dengan dua wajah milik orang yang sama (deteksi terlewat/negatif palsu) atau mungkin salah memprediksi bahwa dua wajah dari orang yang berbeda adalah orang yang sama (alarm palsu/positif palsu).

Desain aplikasi dan pengaturan ambang batas

- Anda dapat menetapkan ambang batas yang menentukan tingkat kepercayaan minimum yang diperlukan untuk mengembalikan hasil. Memilih ambang kepercayaan yang tepat sangat penting untuk desain aplikasi dan pengambilan keputusan berdasarkan output sistem.
- Tingkat kepercayaan yang Anda pilih harus mencerminkan kasus penggunaan Anda. Beberapa contoh kasus penggunaan dan ambang kepercayaan:

- Aplikasi Foto: Ambang batas yang lebih rendah (misalnya, 80%) mungkin cukup untuk mengidentifikasi anggota keluarga dalam foto.
- Skenario Taruhan Tinggi: Dalam kasus penggunaan di mana risiko deteksi yang terlewat atau alarm palsu lebih tinggi, seperti aplikasi keamanan, sistem harus menggunakan tingkat kepercayaan yang lebih tinggi. Dalam kasus seperti itu, ambang batas yang lebih tinggi (misalnya, 99%) direkomendasikan untuk kecocokan wajah yang akurat.

Untuk informasi lebih lanjut tentang pengaturan dan pemahaman ambang kepercayaan, lihat.

[Mencari wajah dalam koleksi](#)

Pedoman atribut wajah

Berikut adalah spesifikasi mengenai bagaimana Amazon Rekognition memproses dan mengembalikan atribut wajah.

- FaceDetail Objek: Untuk setiap wajah yang terdeteksi, FaceDetail objek dikembalikan. Ini FaceDetail berisi data tentang tengara wajah, kualitas, pose, dan banyak lagi.
- Prediksi Atribut: Atribut seperti emosi, jenis kelamin, usia, dan lainnya diprediksi. Tingkat kepercayaan diberikan untuk setiap prediksi, dan prediksi dikembalikan dengan skor kepercayaan masing-masing. Ambang kepercayaan 99% direkomendasikan untuk kasus penggunaan sensitif. Untuk estimasi usia, titik tengah rentang usia yang diprediksi menawarkan perkiraan terbaik.

Perhatikan bahwa prediksi gender dan emosi didasarkan pada penampilan fisik dan tidak boleh digunakan untuk menentukan identitas gender atau keadaan emosional yang sebenarnya. Prediksi biner tipe kelamin (laki-laki/perempuan) didasarkan pada penampilan fisik wajah dalam citra tertentu. Itu tidak menunjukkan identitas gender seseorang, dan Anda tidak boleh menggunakan Rekognition untuk membuat tekad seperti itu. Kami tidak menyarankan penggunaan prediksi biner gender untuk membuat keputusan yang memengaruhi hak, privasi, atau akses seseorang ke layanan. Demikian pula, prediksi emosional tidak menunjukkan keadaan emosi internal seseorang yang sebenarnya, dan Anda tidak boleh menggunakan Rekognition untuk membuat tekad seperti itu. Seseorang yang berpura-pura memiliki wajah bahagia dalam sebuah gambar mungkin terlihat bahagia, tetapi mungkin tidak mengalami kebahagiaan.

Kasus Aplikasi dan Penggunaan

Berikut adalah beberapa aplikasi praktis dan kasus penggunaan untuk atribut ini:

- Aplikasi: Atribut seperti Smile, Pose, dan Sharpness dapat digunakan untuk memilih gambar profil atau memperkirakan demografi secara anonim.
- Kasus Penggunaan Umum: Aplikasi media sosial dan estimasi demografis di acara atau toko ritel adalah contoh khas.

Untuk informasi lebih rinci tentang setiap atribut, lihat [FaceDetail](#).

Mendeteksi wajah dalam citra

Amazon Rekognition Image menyediakan operasi [DetectFaces](#) yang mencari fitur wajah utama seperti mata, hidung, dan mulut untuk mendeteksi wajah dalam gambar input. Amazon Rekognition Image mendeteksi 100 wajah terbesar dalam sebuah citra.

Anda dapat menyediakan citra input sebagai array bit citra (bit citra yang dikodekan base64), atau menentukan objek Amazon S3. Dalam prosedur ini, Anda mengunggah citra (JPEG atau PNG) ke bucket S3 dan menentukan nama kunci objek.

Untuk mendeteksi wajah dalam citra

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan `AmazonRekognitionFullAccess` dan `AmazonS3ReadOnlyAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Unggah citra (yang berisi satu atau beberapa wajah) ke bucket S3 Anda.

Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

3. Gunakan contoh berikut untuk memanggil `DetectFaces`.

Java

Contoh ini menampilkan perkiraan rentang usia untuk wajah yang terdeteksi, dan daftar JSON untuk semua atribut wajah yang terdeteksi. Ubah nilai `photo` ke nama file citra. Ubah nilai `bucket` ke bucket Amazon S3 tempat citra disimpan.

```
//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(faceDetails));
        }

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

}

}
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
import java.util.List;

//snippet-start:[rekognition.java2.detect_labels.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.DetectFacesRequest;
import software.amazon.awssdk.services.rekognition.model.DetectFacesResponse;
```

```
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceDetail;
import software.amazon.awssdk.services.rekognition.model.AgeRange;

//snippet-end:[rekognition.java2.detect_labels.import]

public class DetectFaces {

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

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

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

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

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

        try {
            S3Object s3Object = S3Object.builder()
                .bucket(bucket)
```

```

        .name(image)
        .build() ;

Image myImage = Image.builder()
    .s3object(s3object)
    .build();

DetectFacesRequest facesRequest = DetectFacesRequest.builder()
    .attributes(Attribute.ALL)
    .image(myImage)
    .build();

DetectFacesResponse facesResponse =
rekClient.detectFaces(facesRequest);
List<FaceDetail> faceDetails = facesResponse.faceDetails();
for (FaceDetail face : faceDetails) {
    AgeRange ageRange = face.ageRange();
    System.out.println("The detected face is estimated to be
between "
                        + ageRange.low().toString() + " and " +
ageRange.high().toString()
                        + " years old.");

    System.out.println("There is a smile :
"+face.smile().value().toString());
}

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

```

AWS CLI

Contoh ini menampilkan output JSON dari detect-faces AWS CLI operasi. Ganti file dengan nama file citra. Ganti bucket dengan nama bucket Amazon S3 yang berisi file citra.

```
aws rekognition detect-faces --image '{"S3object":{"Bucket":"bucket-
name"},"Name":"image-name"}'\
```

```

--attributes "ALL" --profile profile-name --region
region-name

```

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda alih-alih tanda kutip tunggal dan hindari tanda kutip ganda bagian dalam dengan garis miring terbalik (yaitu \) untuk mengatasi kesalahan parser yang mungkin Anda temui. Sebagai contoh, lihat berikut ini:

```

aws rekognition detect-faces --image "{\"S3Object\":{\"Bucket\":\"bucket-name\",
\"Name\":\"image-name\"}}" --attributes "ALL"
--profile profile-name --region region-name

```

Python

Contoh ini menampilkan perkiraan rentang usia dan atribut lain untuk wajah yang terdeteksi, serta daftar JSON untuk semua atribut wajah yang terdeteksi. Ubah nilai `photo` ke nama file citra. Ubah nilai `bucket` ke bucket Amazon S3 tempat citra disimpan. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```

import boto3
import json

def detect_faces(photo, bucket, region):

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

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

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

```



```
print('Here are the other attributes:')
print(json.dumps(faceDetail, indent=4, sort_keys=True))

# Access predictions for individual face details and print them
print("Gender: " + str(faceDetail['Gender']))
print("Smile: " + str(faceDetail['Smile']))
print("Eyeglasses: " + str(faceDetail['Eyeglasses']))
print("Face Occluded: " + str(faceDetail['FaceOccluded']))
print("Emotions: " + str(faceDetail['Emotions'][0]))

return len(response['FaceDetails'])

def main():
    photo='photo'
    bucket='bucket'
    region='region'
    face_count=detect_faces(photo, bucket, region)
    print("Faces detected: " + str(face_count))

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

.NET

Contoh ini menampilkan perkiraan rentang usia untuk wajah yang terdeteksi, dan daftar JSON untuk semua atribut wajah yang terdeteksi. Ubah nilai photo ke nama file citra. Ubah nilai bucket ke bucket Amazon S3 tempat citra disimpan.

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

using System;
using System.Collections.Generic;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

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

```
String bucket = "bucket";

AmazonRekognitionClient rekognitionClient = new
AmazonRekognitionClient();

DetectFacesRequest detectFacesRequest = new DetectFacesRequest()
{
    Image = new Image()
    {
        S3Object = new S3Object()
        {
            Name = photo,
            Bucket = bucket
        },
    },
    // Attributes can be "ALL" or "DEFAULT".
    // "DEFAULT": BoundingBox, Confidence, Landmarks, Pose, and Quality.
    // "ALL": See https://docs.aws.amazon.com/sdkfornet/v3/apidocs/
items/Rekognition/TFaceDetail.html
    Attributes = new List<String>() { "ALL" }
};

try
{
    DetectFacesResponse detectFacesResponse =
rekognitionClient.DetectFaces(detectFacesRequest);
    bool hasAll = detectFacesRequest.Attributes.Contains("ALL");
    foreach(FaceDetail face in detectFacesResponse.FaceDetails)
    {
        Console.WriteLine("BoundingBox: top={0} left={1} width={2}
height={3}", face.BoundingBox.Left,
            face.BoundingBox.Top, face.BoundingBox.Width,
face.BoundingBox.Height);
        Console.WriteLine("Confidence: {0}\nLandmarks: {1}\nPose:
pitch={2} roll={3} yaw={4}\nQuality: {5}",
            face.Confidence, face.Landmarks.Count, face.Pose.Pitch,
            face.Pose.Roll, face.Pose.Yaw, face.Quality);
        if (hasAll)
            Console.WriteLine("The detected face is estimated to be
between " +
                face.AgeRange.Low + " and " + face.AgeRange.High + "
years old.");
    }
}
```

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

Ruby

Contoh ini menampilkan perkiraan rentang usia untuk wajah yang terdeteksi, serta daftar berbagai atribut wajah. Ubah nilai photo ke nama file citra. Ubah nilai bucket ke bucket Amazon S3 tempat citra disimpan.

```
# Add to your Gemfile
# gem 'aws-sdk-rekognition'
require 'aws-sdk-rekognition'
credentials = Aws::Credentials.new(
  ENV['AWS_ACCESS_KEY_ID'],
  ENV['AWS_SECRET_ACCESS_KEY']
)
bucket = 'bucket' # the bucketname without s3://
photo = 'input.jpg' # the name of file
client = Aws::Rekognition::Client.new credentials: credentials
attrs = {
  image: {
    s3_object: {
      bucket: bucket,
      name: photo
    },
  },
  attributes: ['ALL']
}
response = client.detect_faces attrs
puts "Detected faces for: #{photo}"
response.face_details.each do |face_detail|
  low = face_detail.age_range.low
  high = face_detail.age_range.high
  puts "The detected face is between: #{low} and #{high} years old"
  puts "All other attributes:"
  puts "  bounding_box.width:      #{face_detail.bounding_box.width}"
  puts "  bounding_box.height:     #{face_detail.bounding_box.height}"
```

```

puts " bounding_box.left:      #{face_detail.bounding_box.left}"
puts " bounding_box.top:       #{face_detail.bounding_box.top}"
puts " age.range.low:          #{face_detail.age_range.low}"
puts " age.range.high:         #{face_detail.age_range.high}"
puts " smile.value:            #{face_detail.smile.value}"
puts " smile.confidence:       #{face_detail.smile.confidence}"
puts " eyeglasses.value:       #{face_detail.eyeglasses.value}"
puts " eyeglasses.confidence:  #{face_detail.eyeglasses.confidence}"
puts " sunglasses.value:       #{face_detail.sunglasses.value}"
puts " sunglasses.confidence:  #{face_detail.sunglasses.confidence}"
puts " gender.value:           #{face_detail.gender.value}"
puts " gender.confidence:      #{face_detail.gender.confidence}"
puts " beard.value:            #{face_detail.beard.value}"
puts " beard.confidence:       #{face_detail.beard.confidence}"
puts " mustache.value:         #{face_detail.mustache.value}"
puts " mustache.confidence:    #{face_detail.mustache.confidence}"
puts " eyes_open.value:        #{face_detail.eyes_open.value}"
puts " eyes_open.confidence:   #{face_detail.eyes_open.confidence}"
puts " mout_open.value:        #{face_detail.mouth_open.value}"
puts " mout_open.confidence:   #{face_detail.mouth_open.confidence}"
puts " emotions[0].type:       #{face_detail.emotions[0].type}"
puts " emotions[0].confidence: #{face_detail.emotions[0].confidence}"
puts " landmarks[0].type:     #{face_detail.landmarks[0].type}"
puts " landmarks[0].x:        #{face_detail.landmarks[0].x}"
puts " landmarks[0].y:        #{face_detail.landmarks[0].y}"
puts " pose.roll:              #{face_detail.pose.roll}"
puts " pose.yaw:               #{face_detail.pose.yaw}"
puts " pose.pitch:             #{face_detail.pose.pitch}"
puts " quality.brightness:    #{face_detail.quality.brightness}"
puts " quality.sharpness:     #{face_detail.quality.sharpness}"
puts " confidence:            #{face_detail.confidence}"
puts "-----"
puts ""
end

```

Node.js

Contoh ini menampilkan perkiraan rentang usia untuk wajah yang terdeteksi, serta daftar berbagai atribut wajah. Ubah nilai photo ke nama file citra. Ubah nilai bucket ke bucket Amazon S3 tempat citra disimpan.

Ganti nilai profile_name di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

Jika Anda menggunakan TypeScript definisi, Anda mungkin perlu menggunakan `import AWS from 'aws-sdk'` alih-alih `const AWS = require('aws-sdk')`, untuk menjalankan program dengan Node.js. Anda dapat melihat [AWS SDK for Javascript](#) untuk detailnya selengkapnya. Tergantung bagaimana konfigurasi diatur, Anda juga mungkin perlu menentukan wilayah Anda dengan `AWS.config.update({region: region});`.

```
// Load the SDK
var AWS = require('aws-sdk');
const bucket = 'bucket-name' // the bucketname without s3://
const photo = 'photo-name' // the name of file

var credentials = new AWS.SharedIniFileCredentials({profile: 'profile-name'});
AWS.config.credentials = credentials;
AWS.config.update({region: 'region-name'});

const client = new AWS.Rekognition();
const params = {
  Image: {
    S3Object: {
      Bucket: bucket,
      Name: photo
    },
  },
  Attributes: ['ALL']
}

client.detectFaces(params, function(err, response) {
  if (err) {
    console.log(err, err.stack); // an error occurred
  } else {
    console.log(`Detected faces for: ${photo}`)
    response.FaceDetails.forEach(data => {
      let low = data.AgeRange.Low
      let high = data.AgeRange.High
      console.log(`The detected face is between: ${low} and ${high} years
old`)
      console.log("All other attributes:")
      console.log(`  BoundingBox.Width:      ${data.BoundingBox.Width}`)
      console.log(`  BoundingBox.Height:     ${data.BoundingBox.Height}`)
      console.log(`  BoundingBox.Left:       ${data.BoundingBox.Left}`)
    })
  }
})
```

```

console.log(` BoundingBox.Top:      ${data.BoundingBox.Top}`)
console.log(` Age.Range.Low:         ${data.AgeRange.Low}`)
console.log(` Age.Range.High:        ${data.AgeRange.High}`)
console.log(` Smile.Value:              ${data.Smile.Value}`)
console.log(` Smile.Confidence:          ${data.Smile.Confidence}`)
console.log(` Eyeglasses.Value:         ${data.Eyeglasses.Value}`)
console.log(` Eyeglasses.Confidence:    ${data.Eyeglasses.Confidence}`)
console.log(` Sunglasses.Value:         ${data.Sunglasses.Value}`)
console.log(` Sunglasses.Confidence:    ${data.Sunglasses.Confidence}`)
console.log(` Gender.Value:             ${data.Gender.Value}`)
console.log(` Gender.Confidence:        ${data.Gender.Confidence}`)
console.log(` Beard.Value:              ${data.Beard.Value}`)
console.log(` Beard.Confidence:         ${data.Beard.Confidence}`)
console.log(` Mustache.Value:          ${data.Mustache.Value}`)
console.log(` Mustache.Confidence:     ${data.Mustache.Confidence}`)
console.log(` EyesOpen.Value:          ${data.EyesOpen.Value}`)
console.log(` EyesOpen.Confidence:     ${data.EyesOpen.Confidence}`)
console.log(` MouthOpen.Value:         ${data.MouthOpen.Value}`)
console.log(` MouthOpen.Confidence:    ${data.MouthOpen.Confidence}`)
console.log(` Emotions[0].Type:        ${data.Emotions[0].Type}`)
console.log(` Emotions[0].Confidence:  ${data.Emotions[0].Confidence}`)
console.log(` Landmarks[0].Type:      ${data.Landmarks[0].Type}`)
console.log(` Landmarks[0].X:         ${data.Landmarks[0].X}`)
console.log(` Landmarks[0].Y:         ${data.Landmarks[0].Y}`)
console.log(` Pose.Roll:              ${data.Pose.Roll}`)
console.log(` Pose.Yaw:               ${data.Pose.Yaw}`)
console.log(` Pose.Pitch:             ${data.Pose.Pitch}`)
console.log(` Quality.Brightness:     ${data.Quality.Brightness}`)
console.log(` Quality.Sharpness:      ${data.Quality.Sharpness}`)
console.log(` Confidence:             ${data.Confidence}`)
console.log("-----")
console.log("")
    }) // for response.faceDetails
  } // if
});

```

DetectFaces permintaan operasi

Input ke DetectFaces adalah citra. Pada contoh ini, citra dimuat dari bucket Amazon S3. Parameter `Attributes` menentukan bahwa semua atribut wajah harus dikembalikan. Untuk informasi selengkapnya, lihat [Bekerja dengan citra](#).

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

DetectFaces respon operasi

DetectFaces mengembalikan informasi berikut untuk setiap wajah yang terdeteksi:

- Kotak batas – Koordinat kotak batas yang mengelilingi wajah.
- Kepercayaan – Tingkat kepercayaan yang dimiliki kotak batas berisi wajah.
- Tengara wajah – Array penanda wajah. Untuk setiap penanda (seperti mata kiri, mata kanan, dan mulut), respons memberikan koordinat x dan y.
- Atribut wajah — Satu set atribut wajah, seperti apakah wajah tersumbat, dikembalikan sebagai `FaceDetail` objek. Set meliputi: `AgeRange`, `Jenggot`, `Emosi`, `Kacamata`, `EyeDirection`, `Jenis Kelamin`, `EyesOpen`, `FaceOccluded`, `Kumis`, `MouthOpen`, `Senyum`, dan `Kacamata Hitam`. Untuk setiap atribut tersebut, respons memberikan nilai. Nilainya bisa dari berbagai jenis, seperti tipe Boolean (apakah seseorang mengenakan kacamata hitam), tali (apakah orang tersebut pria atau wanita), atau nilai derajat sudut (untuk nada/menguap arah pandangan mata). Selain itu, untuk sebagian besar atribut, respons juga memberikan kepercayaan pada nilai yang terdeteksi untuk atribut. Perhatikan bahwa sementara `FaceOccluded` dan `EyeDirection` atribut didukung saat menggunakan `DetectFaces`, atribut tersebut tidak didukung saat menganalisis video dengan `StartFaceDetection` dan `GetFaceDetection`.
- Kualitas – Menggambarkan tingkat kecerahan dan ketajaman wajah. Untuk informasi tentang memastikan deteksi wajah sebaik mungkin, lihat [Rekomendasi untuk citra input perbandingan wajah](#).
- Pose – Menggambarkan rotasi wajah di dalam citra.

Permintaan dapat menggambarkan array atribut wajah yang ingin Anda kembalikan. DEFAULTSubset atribut wajah -BoundingBox, Confidence, PoseQuality, dan Landmarks - akan selalu dikembalikan. Anda dapat meminta pengembalian atribut wajah tertentu (selain daftar default) - dengan menggunakan ["DEFAULT", "FACE_OCCLUDED", "EYE_DIRECTION"] atau hanya satu atribut, seperti["FACE_OCCLUDED"]. Anda dapat meminta semua atribut wajah dengan menggunakan["ALL"]. Meminta lebih banyak atribut dapat meningkatkan waktu respons.

Berikut ini adalah contoh respons panggilan DetectFaces API:

```
{
  "FaceDetails": [
    {
      "BoundingBox": {
        "Width": 0.7919622659683228,
        "Height": 0.7510867118835449,
        "Left": 0.08881539851427078,
        "Top": 0.151064932346344
      },
      "AgeRange": {
        "Low": 18,
        "High": 26
      },
      "Smile": {
        "Value": false,
        "Confidence": 89.77348327636719
      },
      "Eyeglasses": {
        "Value": true,
        "Confidence": 99.99996948242188
      },
      "Sunglasses": {
        "Value": true,
        "Confidence": 93.65237426757812
      },
      "Gender": {
        "Value": "Female",
        "Confidence": 99.85968780517578
      },
      "Beard": {
        "Value": false,
        "Confidence": 77.52591705322266
      },
      "Mustache": {
```



```
    "Value": false,
    "Confidence": 94.48904418945312
  },
  "EyesOpen": {
    "Value": true,
    "Confidence": 98.57169342041016
  },
  "MouthOpen": {
    "Value": false,
    "Confidence": 74.33953094482422
  },
  "Emotions": [
    {
      "Type": "SAD",
      "Confidence": 65.56403350830078
    },
    {
      "Type": "CONFUSED",
      "Confidence": 31.277774810791016
    },
    {
      "Type": "DISGUSTED",
      "Confidence": 15.553778648376465
    },
    {
      "Type": "ANGRY",
      "Confidence": 8.012762069702148
    },
    {
      "Type": "SURPRISED",
      "Confidence": 7.621500015258789
    },
    {
      "Type": "FEAR",
      "Confidence": 7.243380546569824
    },
    {
      "Type": "CALM",
      "Confidence": 5.8196024894714355
    },
    {
      "Type": "HAPPY",
      "Confidence": 2.2830512523651123
    }
  ]
}
```

```
],
"Landmarks": [
  {
    "Type": "eyeLeft",
    "X": 0.30225440859794617,
    "Y": 0.41018882393836975
  },
  {
    "Type": "eyeRight",
    "X": 0.6439348459243774,
    "Y": 0.40341562032699585
  },
  {
    "Type": "mouthLeft",
    "X": 0.343580037355423,
    "Y": 0.6951127648353577
  },
  {
    "Type": "mouthRight",
    "X": 0.6306480765342712,
    "Y": 0.6898072361946106
  },
  {
    "Type": "nose",
    "X": 0.47164231538772583,
    "Y": 0.5763645172119141
  },
  {
    "Type": "leftEyeBrowLeft",
    "X": 0.1732882857322693,
    "Y": 0.34452149271965027
  },
  {
    "Type": "leftEyeBrowRight",
    "X": 0.3655243515968323,
    "Y": 0.33231860399246216
  },
  {
    "Type": "leftEyeBrowUp",
    "X": 0.2671719491481781,
    "Y": 0.31669262051582336
  },
  {
    "Type": "rightEyeBrowLeft",
```

```
    "X": 0.5613729953765869,
    "Y": 0.32813435792922974
  },
  {
    "Type": "rightEyeBrowRight",
    "X": 0.7665090560913086,
    "Y": 0.3318614959716797
  },
  {
    "Type": "rightEyeBrowUp",
    "X": 0.6612788438796997,
    "Y": 0.3082450032234192
  },
  {
    "Type": "leftEyeLeft",
    "X": 0.2416982799768448,
    "Y": 0.4085965156555176
  },
  {
    "Type": "leftEyeRight",
    "X": 0.36943578720092773,
    "Y": 0.41230902075767517
  },
  {
    "Type": "leftEyeUp",
    "X": 0.29974061250686646,
    "Y": 0.3971870541572571
  },
  {
    "Type": "leftEyeDown",
    "X": 0.30360740423202515,
    "Y": 0.42347756028175354
  },
  {
    "Type": "rightEyeLeft",
    "X": 0.5755768418312073,
    "Y": 0.4081145226955414
  },
  {
    "Type": "rightEyeRight",
    "X": 0.7050536870956421,
    "Y": 0.39924031496047974
  },
  {
```

```
    "Type": "rightEyeUp",
    "X": 0.642906129360199,
    "Y": 0.39026668667793274
  },
  {
    "Type": "rightEyeDown",
    "X": 0.6423097848892212,
    "Y": 0.41669243574142456
  },
  {
    "Type": "noseLeft",
    "X": 0.4122826159000397,
    "Y": 0.5987403392791748
  },
  {
    "Type": "noseRight",
    "X": 0.5394935011863708,
    "Y": 0.5960900187492371
  },
  {
    "Type": "mouthUp",
    "X": 0.478581964969635,
    "Y": 0.6660456657409668
  },
  {
    "Type": "mouthDown",
    "X": 0.483366996049881,
    "Y": 0.7497162818908691
  },
  {
    "Type": "leftPupil",
    "X": 0.30225440859794617,
    "Y": 0.41018882393836975
  },
  {
    "Type": "rightPupil",
    "X": 0.6439348459243774,
    "Y": 0.40341562032699585
  },
  {
    "Type": "upperJawlineLeft",
    "X": 0.11031254380941391,
    "Y": 0.3980775475502014
  },
  },
```

```
{
  "Type": "midJawlineLeft",
  "X": 0.19301874935626984,
  "Y": 0.7034031748771667
},
{
  "Type": "chinBottom",
  "X": 0.4939905107021332,
  "Y": 0.8877836465835571
},
{
  "Type": "midJawlineRight",
  "X": 0.7990140914916992,
  "Y": 0.6899225115776062
},
{
  "Type": "upperJawlineRight",
  "X": 0.8548634648323059,
  "Y": 0.38160091638565063
}
],
"Pose": {
  "Roll": -5.83309268951416,
  "Yaw": -2.4244730472564697,
  "Pitch": 2.6216139793395996
},
"Quality": {
  "Brightness": 96.16363525390625,
  "Sharpness": 95.51618957519531
},
"Confidence": 99.99872589111328,
"FaceOccluded": {
  "Value": true,
  "Confidence": 99.99726104736328
},
"EyeDirection": {
  "Yaw": 16.299732,
  "Pitch": -6.407457,
  "Confidence": 99.968704
}
},
"ResponseMetadata": {
  "RequestId": "8bf02607-70b7-4f20-be55-473fe1bba9a2",
```

```
"HTTPStatusCode": 200,  
"HTTPHeaders": {  
  "x-amzn-requestid": "8bf02607-70b7-4f20-be55-473fe1bba9a2",  
  "content-type": "application/x-amz-json-1.1",  
  "content-length": "3409",  
  "date": "Wed, 26 Apr 2023 20:18:50 GMT"  
},  
"RetryAttempts": 0  
}  
}
```

Perhatikan hal berikut:

- Data Pose menggambarkan rotasi wajah yang terdeteksi. Anda dapat menggunakan kombinasi data BoundingBox dan Pose untuk menggambar kotak batas di sekitar wajah yang ditampilkan aplikasi Anda.
- Parameter Quality menggambarkan tingkat kecerahan dan ketajaman wajah. Mungkin ini berguna untuk membandingkan wajah di seluruh citra dan menemukan wajah terbaik.
- Respons sebelumnya menunjukkan semua landmarks wajah yang dapat dideteksi layanan, semua atribut wajah dan emosi. Untuk mendapatkan semua ini dalam respons, Anda harus menentukan parameter attributes dengan nilai ALL. Secara default, API DetectFaces hanya mengembalikan lima atribut wajah berikut: BoundingBox, Confidence, Pose, Quality dan landmarks. Penanda default yang dikembalikan adalah: eyeLeft, eyeRight, nose, mouthLeft, dan mouthRight.

Membandingkan wajah dalam citra

Dengan Rekognition Anda dapat membandingkan wajah antara dua gambar menggunakan operasi. [CompareFaces](#) Fitur ini berguna untuk aplikasi seperti verifikasi identitas atau pencocokan foto.

CompareFaces membandingkan wajah pada gambar sumber dengan setiap wajah di gambar target. Gambar diteruskan ke CompareFaces sebagai:

- Representasi gambar yang disandikan base64.
- Objek Amazon S3.

Deteksi Wajah vs Perbandingan Wajah

Perbandingan wajah berbeda dengan deteksi wajah. Deteksi wajah (yang menggunakan DetectFaces) hanya mengidentifikasi keberadaan dan lokasi wajah dalam gambar atau video. Sebaliknya, perbandingan wajah melibatkan membandingkan wajah yang terdeteksi dalam gambar sumber dengan wajah dalam gambar target untuk menemukan kecocokan.

Ambang kesamaan

Gunakan `similarityThreshold` parameter untuk menentukan tingkat kepercayaan minimum untuk kecocokan yang akan dimasukkan dalam respons. Secara default, hanya wajah dengan skor kemiripan yang lebih besar dari atau sama dengan 80% yang dikembalikan dalam bentuk respons.

Note

`CompareFaces` menggunakan algoritma pembelajaran mesin, yang probabilistik. Negatif palsu merupakan prediksi yang keliru saat wajah dalam citra target memiliki skor kepercayaan kemiripan yang rendah bila dibandingkan dengan wajah dalam citra sumber. Untuk mengurangi kemungkinan negatif palsu, sebaiknya Anda membandingkan citra target dengan beberapa citra sumber. Jika Anda berencana untuk menggunakan `CompareFaces` pada pengambilan keputusan yang memengaruhi hak, privasi, atau akses seseorang ke layanan, sebaiknya hasil dari `CompareFaces` diteruskan kepada manusia untuk dilakukan peninjauan dan validasi lebih lanjut sebelum mengambil tindakan.

Contoh kode berikut menunjukkan cara menggunakan `CompareFaces` operasi untuk berbagai AWS SDK. Dalam AWS CLI contoh, Anda mengunggah dua gambar JPEG ke bucket Amazon S3 Anda dan menentukan nama kunci objek. Dalam contoh lain, Anda memuat dua file dari sistem file lokal dan memasukkannya sebagai array bit citra.

Untuk membandingkan wajah

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan `AmazonRekognitionFullAccess` dan `AmazonS3ReadOnlyAccess` (hanya AWS CLI contoh) izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan kode contoh berikut untuk memanggil operasi `CompareFaces`.

Java

Contoh ini menampilkan informasi tentang pencocokan wajah pada citra sumber dan target yang dimuat dari sistem file lokal.

Ganti nilai `sourceImage` dan `targetImage` dengan alur dan nama file citra sumber dan target.

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

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.CompareFacesMatch;
import com.amazonaws.services.rekognition.model.CompareFacesRequest;
import com.amazonaws.services.rekognition.model.CompareFacesResult;
import com.amazonaws.services.rekognition.model.ComparedFace;
import java.util.List;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import com.amazonaws.util.IOUtils;

public class CompareFaces {

    public static void main(String[] args) throws Exception{
        Float similarityThreshold = 70F;
        String sourceImage = "source.jpg";
        String targetImage = "target.jpg";
        ByteBuffer sourceImageBytes=null;
        ByteBuffer targetImageBytes=null;

        AmazonRekognition rekognitionClient =
        AmazonRekognitionClientBuilder.defaultClient();

        //Load source and target images and create input parameters
```



```
try (InputStream inputStream = new FileInputStream(new
File(sourceImage))) {
    sourceImageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
}
catch(Exception e)
{
    System.out.println("Failed to load source image " + sourceImage);
    System.exit(1);
}
try (InputStream inputStream = new FileInputStream(new
File(targetImage))) {
    targetImageBytes =
ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
}
catch(Exception e)
{
    System.out.println("Failed to load target images: " + targetImage);
    System.exit(1);
}

Image source=new Image()
    .withBytes(sourceImageBytes);
Image target=new Image()
    .withBytes(targetImageBytes);

CompareFacesRequest request = new CompareFacesRequest()
    .withSourceImage(source)
    .withTargetImage(target)
    .withSimilarityThreshold(similarityThreshold);

// Call operation
CompareFacesResult
compareFacesResult=rekognitionClient.compareFaces(request);

// Display results
List <CompareFacesMatch> faceDetails =
compareFacesResult.getFaceMatches();
for (CompareFacesMatch match: faceDetails){
    ComparedFace face= match.getFace();
    BoundingBox position = face.getBoundingBox();
    System.out.println("Face at " + position.getLeft().toString()
        + " " + position.getTop()
        + " matches with " + match.getSimilarity().toString())
```

```
        + "% confidence.");

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

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

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
import java.util.List;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import software.amazon.awssdk.services.rekognition.model.CompareFacesMatch;
import software.amazon.awssdk.services.rekognition.model.CompareFacesRequest;
import software.amazon.awssdk.services.rekognition.model.CompareFacesResponse;
import software.amazon.awssdk.services.rekognition.model.ComparedFace;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;

// snippet-end:[rekognition.java2.detect_faces.import]

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

```
public class CompareFaces {

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "  <pathSource> <pathTarget>\n\n" +
            "Where:\n" +
            "  pathSource - The path to the source image (for example, C:\\AWS\\
\\pic1.png). \n " +
            "  pathTarget - The path to the target image (for example, C:\\AWS\\
\\pic2.png). \n\n";

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

        Float similarityThreshold = 70F;
        String sourceImage = args[0];
        String targetImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        compareTwoFaces(rekClient, similarityThreshold, sourceImage,
targetImage);
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.compare_faces.main]
    public static void compareTwoFaces(RekognitionClient rekClient, Float
similarityThreshold, String sourceImage, String targetImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            InputStream tarStream = new FileInputStream(targetImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            SdkBytes targetBytes = SdkBytes.fromInputStream(tarStream);

            // Create an Image object for the source image.
            Image souImage = Image.builder()
```

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

    Image tarImage = Image.builder()
        .bytes(targetBytes)
        .build();

    CompareFacesRequest facesRequest = CompareFacesRequest.builder()
        .sourceImage(souImage)
        .targetImage(tarImage)
        .similarityThreshold(similarityThreshold)
        .build();

    // Compare the two images.
    CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
    List<CompareFacesMatch> faceDetails =
compareFacesResult.faceMatches();
    for (CompareFacesMatch match: faceDetails){
        ComparedFace face= match.face();
        BoundingBox position = face.boundingBox();
        System.out.println("Face at " + position.left().toString()
            + " " + position.top()
            + " matches with " + face.confidence().toString()
            + "% confidence.");
    }
    List<ComparedFace> uncompaired = compareFacesResult.unmatchedFaces();
    System.out.println("There was " + uncompaired.size() + " face(s) that
did not match");
    System.out.println("Source image rotation: " +
compareFacesResult.sourceImageOrientationCorrection());
    System.out.println("target image rotation: " +
compareFacesResult.targetImageOrientationCorrection());

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println("Failed to load source image " + sourceImage);
        System.exit(1);
    }
}
// snippet-end:[rekognition.java2.compare_faces.main]
}
```

AWS CLI

Contoh ini menampilkan output JSON dari `compare-faces` AWS CLI operasi.

Ganti `bucket-name` dengan nama bucket Amazon S3 yang berisi citra sumber dan target. Ganti `source.jpg` dan `target.jpg` dengan nama file untuk citra sumber dan target.

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

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda alih-alih tanda kutip tunggal dan hindari tanda kutip ganda bagian dalam dengan garis miring terbalik (yaitu `\`) untuk mengatasi kesalahan parser yang mungkin Anda temui. Sebagai contoh, lihat berikut ini:

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

Python

Contoh ini menampilkan informasi tentang pencocokan wajah pada citra sumber dan target yang dimuat dari sistem file lokal.

Ganti nilai `source_file` dan `target_file` dengan alur dan nama file citra sumber dan target. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/  
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
import boto3  
  
def compare_faces(sourceFile, targetFile):
```

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

imageSource = open(sourceFile, 'rb')
imageTarget = open(targetFile, 'rb')

response = client.compare_faces(SimilarityThreshold=80,
                               SourceImage={'Bytes': imageSource.read()},
                               TargetImage={'Bytes': imageTarget.read()})

for faceMatch in response['FaceMatches']:
    position = faceMatch['Face']['BoundingBox']
    similarity = str(faceMatch['Similarity'])
    print('The face at ' +
          str(position['Left']) + ' ' +
          str(position['Top']) +
          ' matches with ' + similarity + '% confidence')

imageSource.close()
imageTarget.close()
return len(response['FaceMatches'])

def main():
    source_file = 'source-file-name'
    target_file = 'target-file-name'
    face_matches = compare_faces(source_file, target_file)
    print("Face matches: " + str(face_matches))

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

.NET

Contoh ini menampilkan informasi tentang pencocokan wajah pada citra sumber dan target yang dimuat dari sistem file lokal.

Ganti nilai `sourceImage` dan `targetImage` dengan alur dan nama file citra sumber dan target.

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

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

public class CompareFaces
{
    public static void Example()
    {
        float similarityThreshold = 70F;
        String sourceImage = "source.jpg";
        String targetImage = "target.jpg";

        AmazonRekognitionClient rekognitionClient = new
AmazonRekognitionClient();

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

        Amazon.Rekognition.Model.Image imageTarget = new
Amazon.Rekognition.Model.Image();
        try
        {
            using (FileStream fs = new FileStream(targetImage, FileMode.Open,
FileAccess.Read))
            {
                byte[] data = new byte[fs.Length];
                data = new byte[fs.Length];
            }
        }
    }
}
```

```
        fs.Read(data, 0, (int)fs.Length);
        imageTarget.Bytes = new MemoryStream(data);
    }
}
catch (Exception)
{
    Console.WriteLine("Failed to load target image: " + targetImage);
    return;
}

CompareFacesRequest compareFacesRequest = new CompareFacesRequest()
{
    SourceImage = imageSource,
    TargetImage = imageTarget,
    SimilarityThreshold = similarityThreshold
};

// Call operation
CompareFacesResponse compareFacesResponse =
rekognitionClient.CompareFaces(compareFacesRequest);

// Display results
foreach(CompareFacesMatch match in compareFacesResponse.FaceMatches)
{
    ComparedFace face = match.Face;
    BoundingBox position = face.BoundingBox;
    Console.WriteLine("Face at " + position.Left
        + " " + position.Top
        + " matches with " + match.Similarity
        + "% confidence.");
}

Console.WriteLine("There was " +
compareFacesResponse.UnmatchedFaces.Count + " face(s) that did not match");
}
}
```

Ruby

Contoh ini menampilkan informasi tentang pencocokan wajah pada citra sumber dan target yang dimuat dari sistem file lokal.

Ganti nilai `photo_source` dan `photo_target` dengan alur dan nama file citra sumber dan target.

```
# Add to your Gemfile
# gem 'aws-sdk-rekognition'
require 'aws-sdk-rekognition'
credentials = Aws::Credentials.new(
  ENV['AWS_ACCESS_KEY_ID'],
  ENV['AWS_SECRET_ACCESS_KEY']
)
bucket      = 'bucket' # the bucketname without s3://
photo_source = 'source.jpg'
photo_target = 'target.jpg'
client      = Aws::Rekognition::Client.new credentials: credentials
attrs = {
  source_image: {
    s3_object: {
      bucket: bucket,
      name: photo_source
    },
  },
  target_image: {
    s3_object: {
      bucket: bucket,
      name: photo_target
    },
  },
  similarity_threshold: 70
}
response = client.compare_faces attrs
response.face_matches.each do |face_match|
  position = face_match.face.bounding_box
  similarity = face_match.similarity
  puts "The face at: #{position.left}, #{position.top} matches with
#{similarity} % confidence"
end
```

Node.js

Contoh ini menampilkan informasi tentang pencocokan wajah pada citra sumber dan target yang dimuat dari sistem file lokal.

Ganti nilai `photo_source` dan `photo_target` dengan alur dan nama file citra sumber dan target. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
// Load the SDK
var AWS = require('aws-sdk');
const bucket = 'bucket-name' // the bucket name without s3://
const photo_source = 'photo-source-name' // path and the name of file
const photo_target = 'photo-target-name'

var credentials = new AWS.SharedIniFileCredentials({profile: 'profile-name'});
AWS.config.credentials = credentials;
AWS.config.update({region: 'region-name'});

const client = new AWS.Rekognition();
const params = {
  SourceImage: {
    S3Object: {
      Bucket: bucket,
      Name: photo_source
    },
  },
  TargetImage: {
    S3Object: {
      Bucket: bucket,
      Name: photo_target
    },
  },
  SimilarityThreshold: 70
}
client.compareFaces(params, function(err, response) {
  if (err) {
    console.log(err, err.stack); // an error occurred
  } else {
    response.FaceMatches.forEach(data => {
      let position = data.Face.BoundingBox
      let similarity = data.Similarity
      console.log(`The face at: ${position.Left}, ${position.Top} matches
with ${similarity} % confidence`)
    }) // for response.faceDetails
  } // if
});
```

CompareFaces permintaan operasi

Input ke CompareFaces adalah citra. Pada contoh ini, citra sumber dan target dimuat dari sistem file lokal. Parameter input `SimilarityThreshold` menentukan kepercayaan minimum bahwa wajah yang dibandingkan harus cocok agar dapat dimasukkan dalam respons. Untuk informasi selengkapnya, lihat [Bekerja dengan citra](#).

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

CompareFaces respon operasi

Respons meliputi:

- Array kecocokan wajah: Daftar wajah yang cocok dengan skor kesamaan dan metadata untuk setiap wajah yang cocok. Jika beberapa wajah cocok, `faceMatches` array mencakup semua kecocokan wajah.
- Detail kecocokan wajah: Setiap wajah yang cocok juga menyediakan kotak pembatas, nilai kepercayaan, lokasi tengara, dan skor kesamaan.
- Daftar wajah yang tak tertandingi: Respons juga mencakup wajah dari gambar target yang tidak cocok dengan wajah gambar sumber. Termasuk kotak pembatas untuk setiap wajah yang tak tertandingi.
- Informasi wajah sumber: Termasuk informasi tentang wajah dari gambar sumber yang digunakan untuk perbandingan, termasuk kotak pembatas dan nilai kepercayaan.

Contoh menunjukkan bahwa satu kecocokan wajah ditemukan pada gambar target. Untuk kecocokan wajah tersebut, diberikan kotak batas dan nilai kepercayaan diri (tingkat kepercayaan yang dimiliki Amazon Rekognition bahwa kotak batas berisi wajah). Skor kesamaan 99,99 menunjukkan seberapa mirip wajah. Contoh ini juga menunjukkan satu wajah yang ditemukan Amazon Rekognition pada gambar target yang tidak cocok dengan wajah yang dianalisis dalam gambar sumber.

```
{
  "FaceMatches": [{
    "Face": {
      "BoundingBox": {
        "Width": 0.5521978139877319,
        "Top": 0.1203877404332161,
        "Left": 0.23626373708248138,
        "Height": 0.3126954436302185
      },
      "Confidence": 99.98751068115234,
      "Pose": {
        "Yaw": -82.36799621582031,
        "Roll": -62.13221740722656,
        "Pitch": 0.8652129173278809
      },
      "Quality": {
        "Sharpness": 99.99880981445312,
        "Brightness": 54.49755096435547
      },
      "Landmarks": [{
        "Y": 0.2996366024017334,
        "X": 0.41685718297958374,
        "Type": "eyeLeft"
      },
      {
        "Y": 0.2658946216106415,
        "X": 0.4414493441581726,
        "Type": "eyeRight"
      },
      {
        "Y": 0.3465650677680969,
        "X": 0.48636093735694885,
        "Type": "nose"
      },
      {
        "Y": 0.30935320258140564,
        "X": 0.6251809000968933,
        "Type": "mouthLeft"
      },
      {
        "Y": 0.26942989230155945,
        "X": 0.6454493403434753,
        "Type": "mouthRight"
      }
    ]
  }
}
```

```
    }
  ]
},
"Similarity": 100.0
]],
"SourceImageOrientationCorrection": "ROTATE_90",
"TargetImageOrientationCorrection": "ROTATE_90",
"UnmatchedFaces": [{
  "BoundingBox": {
    "Width": 0.4890109896659851,
    "Top": 0.6566604375839233,
    "Left": 0.10989011079072952,
    "Height": 0.278298944234848
  },
  "Confidence": 99.99992370605469,
  "Pose": {
    "Yaw": 51.51519012451172,
    "Roll": -110.32493591308594,
    "Pitch": -2.322134017944336
  },
  "Quality": {
    "Sharpness": 99.99671173095703,
    "Brightness": 57.23163986206055
  },
  "Landmarks": [{
    "Y": 0.8288310766220093,
    "X": 0.3133862614631653,
    "Type": "eyeLeft"
  },
  {
    "Y": 0.7632885575294495,
    "X": 0.28091415762901306,
    "Type": "eyeRight"
  },
  {
    "Y": 0.7417283654212952,
    "X": 0.3631140887737274,
    "Type": "nose"
  },
  {
    "Y": 0.8081989884376526,
    "X": 0.48565614223480225,
    "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
}
}
```

Mendeteksi wajah dalam video yang tersimpan

Amazon Rekognition Video dapat mendeteksi wajah dalam video yang disimpan di bucket Amazon S3 dan memberikan informasi seperti:

- Waktu atau berapa kali wajah terdeteksi dalam video.
- Lokasi wajah dalam frame video saat terdeteksi.
- Penanda wajah seperti posisi mata kiri.
- Atribut tambahan seperti yang dijelaskan pada [the section called “Pedoman atribut wajah”](#) halaman.

Deteksi wajah dalam video yang tersimpan oleh Amazon Rekognition Video adalah operasi tidak sinkron. Untuk memulai deteksi wajah dalam video, hubungi [StartFaceDetection](#). Amazon Rekognition Video menerbitkan status penyelesaian analisis video ke topik Amazon Simple Notification Service (Amazon SNS). Jika analisis video berhasil, Anda dapat menelepon [GetFaceDetection](#) untuk mendapatkan hasil analisis video. Untuk informasi selengkapnya tentang memulai analisis video dan mendapatkan hasilnya, lihat [Memanggil operasi Amazon Rekognition Video](#).

Prosedur ini melebar ke kode di [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#), yang menggunakan antrean Amazon Simple Queue Service (Amazon SQS) untuk mendapatkan status penyelesaian permintaan analisis video.

Untuk mendeteksi wajah dalam video yang disimpan di bucket Amazon S3 (SDK)

1. Lakukan [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#).
2. Tambahkan kode berikut ke kelas VideoDetect yang Anda buat di langkah 1.

AWS CLI

- Dalam contoh kode berikut, ubah bucket-name dan video-name ke nama bucket Amazon S3 dan nama file yang Anda tentukan di langkah 2.
- Ubah region-name ke wilayah AWS yang Anda gunakan. Ganti nilai profile_name dengan nama profil pengembang Anda.
- Ubah TopicARN ke ARN dari topik Amazon SNS yang Anda buat pada langkah 3 [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#).
- Perubahkan RoleARN ke ARN dari peran layanan IAM yang Anda buat di langkah 7 [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#).

```
aws rekognition start-face-detection --video '{"S3Object":{"Bucket":"Bucket-Name","Name":"Video-Name"}}' --notification-channel \
'{"SNSTopicArn":"Topic-ARN","RoleArn":"Role-ARN"}' --region region-name --
profile profile-name
```

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda alih-alih tanda kutip tunggal dan hindari tanda kutip ganda bagian dalam dengan garis miring terbalik (yaitu\) untuk mengatasi kesalahan parser yang mungkin Anda temui. Sebagai contoh, lihat berikut ini:

```
aws rekognition start-face-detection --video "{\\"S3Object\\":{\\"Bucket\\":
\\"Bucket-Name\\",\\"Name\\":\\"Video-Name\\"}}" --notification-channel \
"{\\"SNSTopicArn\\":\\"Topic-ARN\\",\\"RoleArn\\":\\"Role-ARN\\"}" --region region-name
--profile profile-name
```

Setelah menjalankan `StartFaceDetection` operasi dan mendapatkan nomor ID pekerjaan, jalankan `GetFaceDetection` operasi berikut dan berikan nomor ID pekerjaan:

```
aws rekognition get-face-detection --job-id job-id-number --profile profile-name
```

Java

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

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

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

    StartFaceDetectionRequest req = new StartFaceDetectionRequest()
        .withVideo(new Video()
            .withS3Object(new S3Object()
                .withBucket(bucket)
                .withName(video)))
        .withNotificationChannel(channel);

    StartFaceDetectionResult startLabelDetectionResult =
rek.startFaceDetection(req);
    startJobId=startLabelDetectionResult.getJobId();
}

private static void GetFaceDetectionResults() throws Exception{

    int maxResults=10;
```



```
String paginationToken=null;
GetFaceDetectionResult faceDetectionResult=null;

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

    faceDetectionResult = rek.getFaceDetection(new
GetFaceDetectionRequest()
        .withJobId(startJobId)
        .withNextToken(paginationToken)
        .withMaxResults(maxResults));

    VideoMetadata videoMetaData=faceDetectionResult.getVideoMetadata();

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

    //Show faces, confidence and detection times
    List<FaceDetection> faces= faceDetectionResult.getFaces();

    for (FaceDetection face: faces) {
        long seconds=face.getTimestamp()/1000;
        System.out.print("Sec: " + Long.toString(seconds) + " ");
        System.out.println(face.getFace().toString());
        System.out.println();
    }
} while (faceDetectionResult !=null && faceDetectionResult.getNextToken() !=
null);

}
```

Dalam fungsi main, ganti baris:

```
StartLabelDetection(bucket, video);

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

dengan:

```
StartFaceDetection(bucket, video);

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

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
//snippet-start:[rekognition.java2.recognize_video_faces.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import java.util.List;
//snippet-end:[rekognition.java2.recognize_video_faces.import]

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

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

        final String usage = "\n" +
            "Usage: " +
            "  <bucket> <video> <topicArn> <roleArn>\n\n" +
            "Where:\n" +
            "  bucket - The name of the bucket in which the video is located (for
            example, (for example, myBucket). \n\n"+
            "  video - The name of video (for example, people.mp4). \n\n" +
```

```
        "    topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic. \n\n" +
        "    roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use. \n\n" ;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];

    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
        .build();

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

    StartFaceDetection(rekClient, channel, bucket, video);
    GetFaceResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

// snippet-start:[rekognition.java2.recognize_video_faces.main]
public static void StartFaceDetection(RekognitionClient rekClient,
                                     NotificationChannel channel,
                                     String bucket,
                                     String video) {

    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();
```

```
        Video vid0b = Video.builder()
            .s3Object(s3obj)
            .build();

        StartFaceDetectionRequest faceDetectionRequest =
StartFaceDetectionRequest.builder()
            .jobTag("Faces")
            .faceAttributes(FaceAttributes.ALL)
            .notificationChannel(channel)
            .video(vid0b)
            .build();

        StartFaceDetectionResponse startLabelDetectionResult =
rekClient.startFaceDetection(faceDetectionRequest);
        startJobId=startLabelDetectionResult.jobId();

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

public static void GetFaceResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetFaceDetectionResponse faceDetectionResponse=null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (faceDetectionResponse !=null)
                paginationToken = faceDetectionResponse.nextToken();

            GetFaceDetectionRequest recognitionRequest =
GetFaceDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {
```

```
        faceDetectionResponse =
rekClient.getFaceDetection(recognitionRequest);
        status = faceDetectionResponse.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null
    VideoMetadata videoMeta=faceDetectionResponse.videoMetadata();
    System.out.println("Format: " + videoMeta.format());
    System.out.println("Codec: " + videoMeta.codec());
    System.out.println("Duration: " + videoMeta.durationMillis());
    System.out.println("FrameRate: " + videoMeta.frameRate());
    System.out.println("Job");

    // Show face information
    List<FaceDetection> faces= faceDetectionResponse.faces();

    for (FaceDetection face: faces) {
        String age = face.face().ageRange().toString();
        String smile = face.face().smile().toString();
        System.out.println("The detected face is estimated to be"
            + age + " years old.");
        System.out.println("There is a smile : "+smile);
    }

    } while (faceDetectionResponse !=null &&
faceDetectionResponse.nextToken() != null);

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

```
}
```

Python

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

# ===== Faces=====
def StartFaceDetection(self):
    response=self.rek.start_face_detection(Video={'S3Object': {'Bucket':
self.bucket, 'Name': self.video}},
        NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
self.snsTopicArn})

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

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

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

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

        for faceDetection in response['Faces']:
            print('Face: ' + str(faceDetection['Face']))
            print('Confidence: ' + str(faceDetection['Face']['Confidence']))
            print('Timestamp: ' + str(faceDetection['Timestamp']))
            print()

        if 'NextToken' in response:
            paginationToken = response['NextToken']
```

```
else:
    finished = True
```

Dalam fungsi main, ganti baris:

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

dengan:

```
analyzer.StartFaceDetection()
if analyzer.GetSQSMessageSuccess()==True:
    analyzer.GetFaceDetectionResults()
```

Note

Jika Anda sudah menjalankan contoh video selain [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#), nama fungsi yang akan diganti berbeda.

3. Jalankan kode tersebut. Informasi tentang wajah yang terdeteksi dalam video ditampilkan.

GetFaceDetection respon operasi

GetFaceDetection mengembalikan array (Faces) yang berisi informasi tentang wajah yang terdeteksi dalam video. Elemen array [FaceDetection](#), ada untuk setiap kali wajah terdeteksi dalam video. Elemen array yang dikembalikan diurutkan berdasarkan waktu, dalam hitungan milidetik sejak awal video.

Contoh berikut adalah respons JSON parsial dari GetFaceDetection. Dalam respons, perhatikan hal berikut:

- Kotak batas – Koordinat kotak batas yang mengelilingi wajah.
- Kepercayaan – Tingkat kepercayaan yang dimiliki kotak batas berisi wajah.
- Tengara wajah – Array penanda wajah. Untuk setiap tengara (seperti mata kiri, mata kanan, dan mulut), respons memberikan x dan y koordinat.

- **Atribut wajah** — Satu set atribut wajah, yang meliputi: Jenggot AgeRange, Emosi, Kacamata,, Jenis Kelamin, EyesOpen, Kumis MouthOpen, Senyum, dan Kacamata Hitam. Nilai dapat terdiri dari berbagai tipe, seperti tipe Boolean (apakah seseorang memakai kacamata hitam) atau string (apakah orang tersebut laki-laki atau perempuan). Selain itu, untuk sebagian besar atribut, respons juga memberikan kepercayaan pada nilai yang terdeteksi untuk atribut. Perhatikan bahwa sementara FaceOccluded dan EyeDirection atribut didukung saat menggunakan DetectFaces, atribut tersebut tidak didukung saat menganalisis video dengan StartFaceDetection dan GetFaceDetection.
- **Timestamp** — Waktu ketika wajah terdeteksi dalam video.
- **Informasi halaman** – Contoh menunjukkan satu halaman informasi deteksi wajah. Anda dapat menentukan jumlah elemen orang yang akan dikembalikan pada parameter input MaxResults untuk GetFaceDetection. Jika hasil melebihi MaxResults, GetFaceDetection mengembalikan token (NextToken) yang digunakan untuk mendapatkan halaman hasil berikutnya. Untuk informasi selengkapnya, lihat [Mendapatkan hasil analisis Amazon Rekognition Video](#).
- **Informasi video** – Respons mencakup informasi tentang format video (VideoMetadata) di setiap halaman informasi yang dikembalikan oleh GetFaceDetection.
- **Kualitas** – Menggambarkan tingkat kecerahan dan ketajaman wajah.
- **Pose** — Menjelaskan rotasi wajah.

```
{
  "Faces": [
    {
      "Face": {
        "BoundingBox": {
          "Height": 0.23000000417232513,
          "Left": 0.42500001192092896,
          "Top": 0.16333332657814026,
          "Width": 0.12937499582767487
        },
        "Confidence": 99.97504425048828,
        "Landmarks": [
          {
            "Type": "eyeLeft",
            "X": 0.46415066719055176,
            "Y": 0.2572723925113678
          },
          {

```



```
        "Type": "eyeRight",
        "X": 0.5068183541297913,
        "Y": 0.23705792427062988
    },
    {
        "Type": "nose",
        "X": 0.49765899777412415,
        "Y": 0.28383663296699524
    },
    {
        "Type": "mouthLeft",
        "X": 0.487221896648407,
        "Y": 0.3452930748462677
    },
    {
        "Type": "mouthRight",
        "X": 0.5142884850502014,
        "Y": 0.33167609572410583
    }
],
"Pose": {
    "Pitch": 15.966927528381348,
    "Roll": -15.547388076782227,
    "Yaw": 11.34195613861084
},
"Quality": {
    "Brightness": 44.80223083496094,
    "Sharpness": 99.95819854736328
}
},
"Timestamp": 0
},
{
    "Face": {
        "BoundingBox": {
            "Height": 0.20000000298023224,
            "Left": 0.029999999329447746,
            "Top": 0.2199999988079071,
            "Width": 0.11249999701976776
        },
        "Confidence": 99.85971069335938,
        "Landmarks": [
            {
                "Type": "eyeLeft",
```

```
        "X": 0.06842322647571564,
        "Y": 0.3010137975215912
    },
    {
        "Type": "eyeRight",
        "X": 0.10543643683195114,
        "Y": 0.29697132110595703
    },
    {
        "Type": "nose",
        "X": 0.09569807350635529,
        "Y": 0.33701086044311523
    },
    {
        "Type": "mouthLeft",
        "X": 0.0732642263174057,
        "Y": 0.3757539987564087
    },
    {
        "Type": "mouthRight",
        "X": 0.10589495301246643,
        "Y": 0.3722417950630188
    }
  ],
  "Pose": {
    "Pitch": -0.5589138865470886,
    "Roll": -5.1093974113464355,
    "Yaw": 18.69594955444336
  },
  "Quality": {
    "Brightness": 43.052337646484375,
    "Sharpness": 99.68138885498047
  }
},
"Timestamp": 0
},
{
  "Face": {
    "BoundingBox": {
      "Height": 0.2177777737379074,
      "Left": 0.7593749761581421,
      "Top": 0.13333334028720856,
      "Width": 0.12250000238418579
    },

```

```
"Confidence": 99.63436889648438,
"Landmarks": [
  {
    "Type": "eyeLeft",
    "X": 0.8005779385566711,
    "Y": 0.20915353298187256
  },
  {
    "Type": "eyeRight",
    "X": 0.8391435146331787,
    "Y": 0.21049551665782928
  },
  {
    "Type": "nose",
    "X": 0.8191410899162292,
    "Y": 0.2523227035999298
  },
  {
    "Type": "mouthLeft",
    "X": 0.8093273043632507,
    "Y": 0.29053622484207153
  },
  {
    "Type": "mouthRight",
    "X": 0.8366993069648743,
    "Y": 0.29101791977882385
  }
],
"Pose": {
  "Pitch": 3.165884017944336,
  "Roll": 1.4182015657424927,
  "Yaw": -11.151537895202637
},
"Quality": {
  "Brightness": 28.910892486572266,
  "Sharpness": 97.61507415771484
}
},
"Timestamp": 0
}.....

],
"JobStatus": "SUCCEEDED",
```

```
"NextToken": "i7fj5XPV/
fwviXqz0eag90w332Jd5G8ZGwf7hooirD/6V1qFmjKF0QZ6QPWUiqv29HbyuhMNqQ==",
  "VideoMetadata": {
    "Codec": "h264",
    "DurationMillis": 67301,
    "FileExtension": "mp4",
    "Format": "QuickTime / MOV",
    "FrameHeight": 1080,
    "FrameRate": 29.970029830932617,
    "FrameWidth": 1920
  }
}
```

Mencari wajah dalam koleksi

Amazon Rekognition memungkinkan Anda menggunakan wajah masukan untuk mencari kecocokan dalam kumpulan wajah yang disimpan. Anda mulai dengan menyimpan informasi tentang wajah yang terdeteksi dalam wadah sisi server yang disebut "koleksi". Koleksi menyimpan wajah dan pengguna individu (beberapa wajah dari orang yang sama). Wajah individu disimpan sebagai vektor wajah, representasi matematis dari wajah (bukan gambar wajah yang sebenarnya). Gambar yang berbeda dari orang yang sama dapat digunakan untuk membuat dan menyimpan beberapa vektor wajah dalam koleksi yang sama. Anda kemudian dapat menggabungkan beberapa vektor wajah dari orang yang sama untuk membuat vektor pengguna. Vektor pengguna dapat menawarkan akurasi pencarian wajah yang lebih tinggi dengan penggambaran yang lebih kuat, berisi berbagai tingkat pencahayaan, ketajaman, pose, penampilan, dll.

Setelah Anda membuat koleksi, Anda dapat menggunakan wajah input untuk mencari vektor pengguna yang cocok atau vektor wajah dalam koleksi. Pencarian terhadap vektor pengguna dapat secara signifikan meningkatkan akurasi dibandingkan dengan mencari terhadap vektor wajah individu. Anda dapat menggunakan wajah yang terdeteksi dalam gambar, video yang disimpan, dan video streaming untuk mencari vektor wajah yang disimpan. Anda dapat menggunakan wajah yang terdeteksi dalam gambar untuk mencari vektor pengguna yang disimpan.

Untuk menyimpan informasi wajah, Anda harus melakukan hal berikut:

1. **Buat Koleksi** - Untuk menyimpan informasi wajah, Anda harus terlebih dahulu membuat ([CreateCollection](#)) koleksi wajah di salah satu AWS Wilayah di akun Anda. Koleksi wajah ini dapat ditentukan ketika Anda memanggil operasi `IndexFaces`.
2. **Index Faces** - [IndexFaces](#) Operasi mendeteksi wajah dalam gambar, mengekstrak, dan menyimpan vektor wajah dalam koleksi. Anda dapat menggunakan operasi ini untuk mendeteksi wajah dalam citra dan menyimpan informasi fitur wajah yang terdeteksi ke dalam koleksi. Ini adalah contoh operasi API berbasis penyimpanan karena layanan menyimpan informasi vektor wajah di server.

Untuk membuat pengguna dan mengaitkan beberapa vektor wajah dengan pengguna, Anda harus melakukan hal berikut:


1. **Buat Pengguna** - Anda harus terlebih dahulu membuat pengguna dengan [CreateUser](#). Anda dapat meningkatkan akurasi pencocokan wajah dengan menggabungkan beberapa vektor wajah dari

orang yang sama ke dalam vektor pengguna. Anda dapat mengaitkan hingga 100 vektor wajah dengan vektor pengguna.

2. Associate Faces - Setelah membuat pengguna, Anda dapat menambahkan vektor wajah yang ada ke pengguna tersebut dengan [AssociateFaces](#) operasi. Vektor wajah harus berada dalam koleksi yang sama dengan vektor pengguna agar dapat dikaitkan dengan vektor pengguna tersebut.

Setelah membuat koleksi dan menyimpan vektor wajah dan pengguna, Anda dapat menggunakan operasi berikut untuk mencari kecocokan wajah:

- [SearchFacesByImage](#)- Untuk mencari wajah individu yang disimpan dengan wajah dari gambar.
- [SearchFaces](#)- Untuk mencari wajah individu yang disimpan dengan ID wajah yang disediakan.
- [SearchUsers](#)- Untuk mencari terhadap pengguna yang disimpan dengan ID wajah atau ID pengguna yang disediakan.
- [SearchUsersByImage](#)- Untuk mencari terhadap pengguna yang disimpan dengan wajah dari gambar.
- [StartFaceSearch](#)- Untuk mencari wajah dalam video yang disimpan.
- [CreateStreamProcessor](#)- Untuk mencari wajah dalam video streaming.

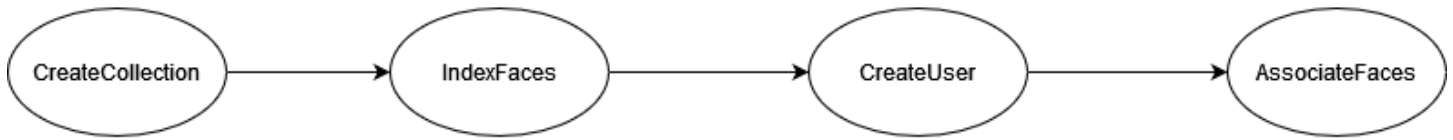
 Note

Koleksi menyimpan vektor wajah, yang merupakan representasi matematis dari wajah. Koleksi tidak menyimpan gambar wajah.

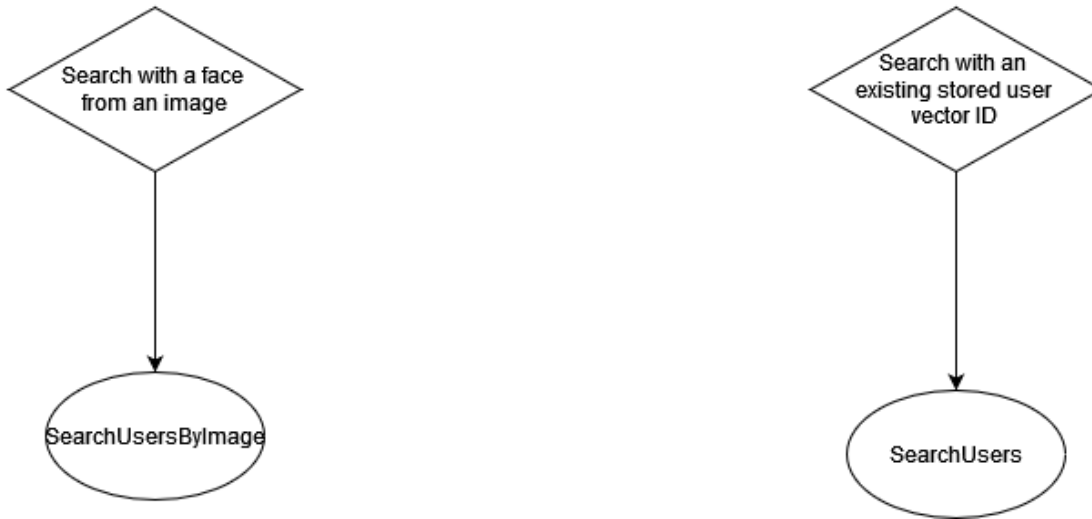
Diagram berikut menunjukkan urutan operasi panggilan, berdasarkan tujuan Anda untuk menggunakan koleksi:

Untuk pencocokan akurasi maksimum dengan Vektor Pengguna:

**Storing user vectors
in a collection**

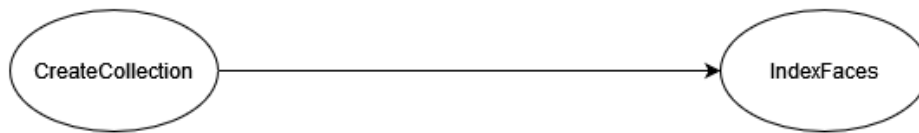


**Searching user
vectors in a collection**

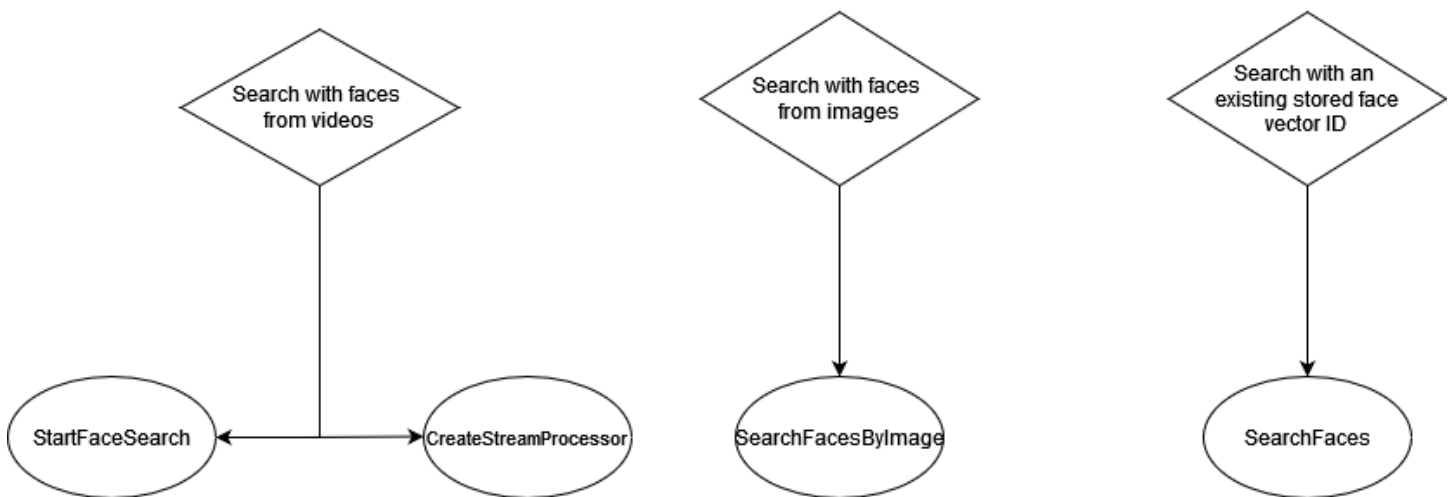


Untuk pencocokan akurasi tinggi dengan Vektor Wajah individual:

Storing faces in a collection



Searching faces in a collection



Anda dapat menggunakan koleksi dalam berbagai skenario. Misalnya, Anda dapat membuat koleksi wajah yang menyimpan wajah yang terdeteksi dari gambar lencana karyawan yang dipindai dan ID yang dikeluarkan pemerintah dengan menggunakan `IndexFaces` dan `AssociateFaces` operasi. Ketika seorang karyawan memasuki gedung, citra wajah karyawan tersebut ditangkap dan dikirim ke operasi `SearchUsersByImage`. Jika kecocokan wajah menghasilkan skor kemiripan yang cukup tinggi (katakanlah 99%), Anda dapat mengautentikasi karyawan tersebut.

Mengelola koleksi

Koleksi wajah adalah sumber daya Amazon Rekognition utama, dan setiap koleksi wajah yang Anda buat memiliki Amazon Resource Name (ARN) yang unik. Anda membuat setiap koleksi wajah di AWS Wilayah tertentu di akun Anda. Saat suatu koleksi dibuat, koleksi tersebut terkait dengan model pendeteksian wajah versi terbaru. Untuk informasi selengkapnya, lihat [Versioning model](#).

Anda dapat melakukan operasi manajemen berikut pada koleksi:

- Membuat koleksi dengan [CreateCollection](#). Untuk informasi selengkapnya, lihat [Membuat koleksi](#).

- Cantumkan koleksi yang tersedia dengan [ListCollections](#). Untuk informasi selengkapnya, lihat [Mencantumkan koleksi](#).
- Jelaskan koleksi dengan [DescribeCollection](#). Untuk informasi selengkapnya, lihat [Menjelaskan koleksi](#).
- Hapus koleksi dengan [DeleteCollection](#). Untuk informasi selengkapnya, lihat [Menghapus koleksi](#).

Mengelola wajah dalam koleksi

Setelah membuat koleksi wajah, wajah dapat disimpan dalam koleksi tersebut. Amazon Rekognition menyediakan operasi berikut untuk mengelola wajah dalam koleksi:

- [IndexFaces](#) Operasi mendeteksi wajah dalam gambar input (JPEG atau PNG), dan menambahkannya ke koleksi wajah yang ditentukan. ID wajah unik dikembalikan untuk setiap wajah yang terdeteksi dalam citra. Setelah wajah disimpan, Anda dapat mencari kecocokan wajah di koleksi wajah. Untuk informasi selengkapnya, lihat [Menambahkan wajah ke koleksi](#).
- [ListFaces](#) Operasi mencantumkan wajah-wajah dalam koleksi. Untuk informasi selengkapnya, lihat [Menambahkan wajah ke koleksi](#).
- [DeleteFaces](#) Operasi menghapus wajah dari koleksi. Untuk informasi selengkapnya, lihat [Menghapus wajah dari koleksi](#).

Mengelola pengguna dalam koleksi

Setelah Anda menyimpan beberapa vektor wajah dari orang yang sama, Anda dapat meningkatkan akurasi dengan mengaitkan semua vektor wajah tersebut ke dalam satu vektor pengguna. Anda dapat menggunakan operasi berikut untuk mengelola pengguna Anda:

- [CreateUser](#)- Operasi membuat pengguna baru dalam koleksi dengan ID pengguna unik yang disediakan.
- [AssociateUsers](#)- Tambahkan 1 - 100 ID wajah unik ke ID pengguna. Setelah mengaitkan setidaknya satu ID wajah ke pengguna, Anda dapat mencari kecocokan terhadap pengguna tersebut dalam koleksi Anda.
- [ListUsers](#)- Daftar pengguna dalam koleksi.
- [DeleteUsers](#)- Menghapus pengguna dari koleksi dengan ID pengguna yang disediakan.
- [DisassociateFaces](#)- Menghapus satu atau lebih ID wajah dari pengguna.

Menggunakan ambang kesamaan untuk mengaitkan wajah

Penting untuk memastikan bahwa wajah yang dikaitkan dengan pengguna semuanya berasal dari orang yang sama. Untuk membantu, `UserMatchThreshold` parameter menentukan kepercayaan kecocokan pengguna minimum yang diperlukan untuk wajah baru yang akan dikaitkan dengan yang `UserID` mengandung setidaknya satu `FaceID` sudah. Ini membantu memastikan bahwa `FaceIds` yang terkait dengan `hakUserID`. Nilai berkisar dari 0-100 dan nilai default adalah 75.

Panduan untuk menggunakan `IndexFaces`

Berikut ini adalah panduan untuk menggunakan `IndexFaces` dalam skenario umum.

Aplikasi keamanan kritis atau publik

- Panggil [IndexFaces](#) dengan gambar yang hanya berisi satu wajah di setiap gambar dan kaitkan ID Wajah yang dikembalikan dengan pengenalan untuk subjek gambar.
- Anda dapat menggunakan [DetectFaces](#) sebelum pengindeksan untuk memverifikasi hanya ada satu wajah dalam gambar. Jika terdeteksi lebih dari satu wajah, lakukan peninjauan lalu kirim ulang citra tersebut dengan citra yang hanya menampilkan satu wajah. Tindakan ini mencegah pengindeksan yang tidak disengaja pada beberapa wajah dan mengaitkannya dengan orang yang sama.

Aplikasi berbagi foto dan media sosial

- Anda harus memanggil `IndexFaces` tanpa batasan citra yang menampilkan beberapa wajah dalam kasus penggunaan seperti album keluarga. Dalam kasus tersebut, Anda perlu mengidentifikasi setiap orang di setiap foto dan menggunakan informasi tersebut untuk mengelompokkan foto berdasarkan orang-orang yang ditampilkan dalam citra tersebut.

Penggunaan umum

- Indeks beberapa gambar berbeda dari orang yang sama, terutama dengan atribut wajah yang berbeda (pose wajah, rambut wajah, dll), buat pengguna, dan kaitkan wajah yang berbeda dengan pengguna tersebut untuk meningkatkan kualitas pencocokan.
- Sertakan proses tinjauan agar pencocokan yang gagal dapat diindeks dengan pengenalan wajah yang benar untuk meningkatkan kemampuan pencocokan wajah berikutnya.

- Untuk informasi tentang kualitas citra, lihat [Rekomendasi untuk citra input perbandingan wajah](#).

Mencari wajah dan pengguna dalam koleksi

Setelah Anda membuat koleksi wajah dan menyimpan vektor wajah dan/atau vektor pengguna, Anda dapat mencari koleksi wajah untuk kecocokan wajah. Dengan Amazon Rekognition, Anda dapat mencari wajah dalam koleksi yang cocok dengan:

- ID wajah yang disediakan ([SearchFaces](#)). Untuk informasi selengkapnya, lihat [Mencari wajah dengan ID wajah](#).
- Wajah terbesar dalam gambar yang disediakan ([SearchFacesByImage](#)). Untuk informasi selengkapnya, lihat [Mencari wajah dengan gambar](#).
- Wajah dalam video yang tersimpan. Untuk informasi selengkapnya, lihat [Mencari video yang disimpan untuk wajah](#).
- Wajah dalam video streaming. Untuk informasi selengkapnya, lihat [Bekerja dengan acara video streaming](#).

Anda dapat menggunakan CompareFaces operasi untuk membandingkan wajah dalam gambar sumber dengan wajah di gambar target. Ruang lingkup perbandingan ini terbatas pada wajah yang terdeteksi pada citra target. Untuk informasi selengkapnya, lihat [Membandingkan wajah dalam gambar](#).

Berbagai operasi Penelusuran yang terlihat dalam daftar berikut membandingkan wajah (diidentifikasi baik oleh gambar input FaceId atau gambar) dengan semua wajah yang disimpan dalam koleksi wajah tertentu:

- [SearchFaces](#)
- [SearchFacesByImage](#)
- [SearchUsers](#)
- [SearchUsersByImage](#)

Menggunakan ambang batas kemiripan untuk mencocokkan wajah

Kami memungkinkan Anda untuk mengontrol hasil dari semua operasi pencarian ([CompareFaces](#), [SearchFaces](#), [SearchFacesByImage](#), [SearchUsers](#), [SearchUsersByImage](#)) dengan memberikan ambang kesamaan sebagai parameter input.

`FaceMatchThreshold`, adalah atribut input ambang kesamaan untuk `SearchFaces` dan `SearchFacesByImage`, dan mengontrol berapa banyak hasil yang dikembalikan berdasarkan kesamaan dengan wajah yang dicocokkan. Atribut ambang kesamaan untuk `SearchUsers` dan `SearchUsersByImage` adalah `UserMatchThreshold`, dan mengontrol berapa banyak hasil yang dikembalikan berdasarkan kesamaan dengan vektor pengguna yang dicocokkan. Atribut ambang batas adalah `SimilarityThreshold` untuk `CompareFaces`.

Respons dengan nilai atribut respons `Similarity` yang lebih rendah dari ambang batas tidak dikirimkan. Ambang batas ini penting untuk dikalibrasi untuk mengkalibrasi kasus penggunaan karena dapat menentukan jumlah positif palsu yang disertakan dalam hasil kecocokan Anda. Hal ini mengendalikan penarikan ulang hasil pencarian Anda—semakin rendah ambang batas, semakin tinggi penarikan ulangnya.

Semua sistem machine learning bersifat probabilistik. Anda harus menggunakan penilaian Anda dalam mengatur ambang batas kemiripan yang tepat, tergantung pada kasus penggunaan Anda. Misalnya, jika Anda ingin membangun aplikasi foto untuk mengidentifikasi anggota keluarga yang mirip, Anda dapat memilih ambang batas yang lebih rendah (misalnya 80%). Di sisi lain, untuk kasus penggunaan penegak hukum yang banyak, sebaiknya gunakan nilai ambang batas tinggi 99% atau lebih untuk mengurangi kesalahan identifikasi yang tidak disengaja.

Selain `FaceMatchThreshold` dan `UserMatchThreshold`, Anda dapat menggunakan atribut `Similarity` respons sebagai sarana untuk mengurangi kesalahan identifikasi yang tidak disengaja. Misalnya, Anda dapat memilih untuk menggunakan ambang batas rendah (seperti 80%) untuk mendapatkan hasil yang lebih banyak. Kemudian Anda dapat menggunakan atribut respons Kesamaan (persentase kesamaan) untuk mempersempit pilihan dan memfilter respons yang tepat dalam aplikasi Anda. Sekali lagi, dengan menggunakan kemiripan yang lebih tinggi (seperti 99% dan lebih) dapat mengurangi risiko kesalahan identifikasi.

Kasus penggunaan yang melibatkan keselamatan publik

Selain rekomendasi yang tercantum dalam [Praktik terbaik untuk sensor, citra input, dan video](#) dan [Panduan untuk menggunakan IndexFaces](#), Anda harus menggunakan praktik terbaik berikut

saat men-deploy sistem pendeteksian dan perbandingan wajah dalam kasus penggunaan yang melibatkan keselamatan publik. Pertama, Anda harus menggunakan ambang kepercayaan 99% atau lebih tinggi untuk mengurangi kesalahan dan positif palsu. Kedua, Anda harus melibatkan peninjau manusia untuk memverifikasi hasil yang diterima dari deteksi wajah atau sistem perbandingan, dan Anda tidak harus membuat keputusan berdasarkan output sistem tanpa tinjauan manusia tambahan. Sistem pendeteksian dan perbandingan wajah harus berfungsi sebagai alat untuk membantu mempersempit bidang dan memungkinkan manusia untuk secara cepat meninjau dan mempertimbangkan opsi. Ketiga, kami menyarankan agar Anda terbuka tentang penggunaan sistem deteksi dan perbandingan wajah dalam kasus penggunaan ini, termasuk, sedapat mungkin, menginformasikan pengguna akhir dan subjek tentang penggunaan sistem ini, memperoleh persetujuan untuk penggunaan tersebut, dan menyediakan mekanisme agar pengguna akhir dan subjek dapat memberikan umpan balik untuk memperbaiki sistem.

Jika Anda adalah lembaga penegak hukum yang menggunakan fitur perbandingan wajah Amazon Rekognition sehubungan dengan investigasi kriminal, Anda harus mengikuti persyaratan yang tercantum dalam [Persyaratan layanan AWS](#). Hal ini termasuk skenario berikut.

- Minta manusia yang terlatih dengan baik meninjau semua keputusan untuk mengambil tindakan yang mungkin berdampak pada kebebasan sipil seseorang atau hak asasi manusia yang setara.
- Latih personil untuk menggunakan sistem pengenalan wajah yang bertanggung jawab.
- Berikan pengungkapan publik tentang penggunaan sistem pengenalan wajah.
- Jangan menggunakan Amazon Rekognition untuk mengawasi seseorang secara berkelanjutan tanpa adanya tinjauan independen atau keadaan darurat.

Dalam semua kasus, pencocokan perbandingan wajah harus dilihat dalam konteks bukti meyakinkan lainnya, dan tidak boleh digunakan sebagai penentu tunggal untuk mengambil tindakan. Namun, jika perbandingan wajah digunakan untuk non-law-enforcement skenario (misalnya, untuk membuka kunci telepon atau mengautentikasi identitas karyawan untuk mengakses gedung kantor pribadi yang aman), keputusan ini tidak memerlukan audit manual karena tidak akan memengaruhi kebebasan sipil seseorang atau hak asasi manusia yang setara.

Jika Anda berencana menggunakan sistem deteksi wajah atau perbandingan wajah untuk kasus penggunaan yang melibatkan keselamatan publik, Anda harus menggunakan praktik terbaik yang disebutkan sebelumnya. Selain itu, Anda harus memeriksa sumber daya yang dipublikasikan tentang penggunaan perbandingan wajah. Hal ini mencakup [Templat Pengembangan Kebijakan Pengenalan Wajah Untuk Digunakan Dalam Aktivitas Intelijen dan Investigasi Pidana](#) yang disediakan oleh Biro Bantuan Hukum Departemen Kehakiman. Templat ini menyediakan beberapa perbandingan wajah

dan sumber daya terkait biometrik dan dirancang untuk menyediakan kerangka kerja pada penegak hukum dan lembaga keselamatan publik dengan mengembangkan kebijakan perbandingan wajah yang mematuhi hukum yang berlaku, mengurangi risiko privasi, dan menetapkan akuntabilitas dan pengawasan entitas. Sumber daya tambahan mencakup [Praktik Privasi Terbaik untuk Penggunaan Pengenalan Wajah Komersial](#) oleh Telekomunikasi dan Administrasi Informasi Nasional dan [Praktik Terbaik untuk Penggunaan Umum Pengenalan Wajah](#) oleh staf Komisi Perdagangan Federal. Sumber daya lain dapat dikembangkan dan dipublikasikan di kemudian hari, dan Anda harus terus mengedukasi diri sendiri tentang topik penting ini.

Sebagai pengingat, Anda harus mematuhi semua hukum yang berlaku saat menggunakan layanan AWS, dan Anda tidak dapat menggunakan layanan AWS dengan cara yang melanggar hak-hak orang lain atau mungkin berbahaya bagi orang lain. Ini berarti bahwa Anda tidak dapat menggunakan layanan AWS untuk kasus penggunaan keselamatan publik dengan cara yang secara ilegal mendiskriminasi seseorang atau melanggar proses jatuh tempo, privasi, atau kebebasan sipil seseorang. Anda harus mendapatkan saran legal yang sesuai yang diperlukan untuk meninjau persyaratan legal atau pertanyaan terkait kasus penggunaan Anda.

Menggunakan Amazon Rekognition untuk membantu keselamatan publik

Amazon Rekognition dapat membantu dalam skenario keselamatan publik dan penegakan hukum—seperti menemukan anak-anak yang hilang, memerangi perdagangan manusia, atau mencegah kejahatan. Dalam skenario keamanan publik dan penegakan hukum, pertimbangkan hal berikut:

- Gunakan Amazon Rekognition sebagai langkah pertama dalam menemukan kecocokan yang memungkinkan. Respons dari operasi wajah Amazon Rekognition memungkinkan Anda mendapatkan serangkaian potensi kecocokan untuk pertimbangan lebih lanjut dengan cepat.
- Jangan gunakan respons Amazon Rekognition untuk membuat keputusan mandiri untuk skenario yang memerlukan analisis manusia. Jika Anda adalah lembaga penegak hukum yang menggunakan Amazon Rekognition untuk membantu mengidentifikasi seseorang sehubungan dengan penyelidikan kriminal, dan tindakan yang akan diambil berdasarkan identifikasi yang dapat memengaruhi kebebasan sipil atau hak asasi manusia yang setara, keputusan untuk mengambil tindakan tersebut harus dilakukan oleh seseorang yang terlatih dengan tepat berdasarkan pemeriksaan independen mereka terhadap bukti identifikasi.
- Gunakan ambang batas kemiripan 99% untuk skenario saat memerlukan kecocokan kemiripan wajah yang sangat akurat. Contohnya adalah mengautentikasi akses ke gedung.
- Ketika hak-hak sipil diutamakan, seperti kasus penggunaan yang melibatkan penegakan hukum, gunakan ambang batas kepercayaan 99% atau lebih tinggi dan terapkan tinjauan manusia

terhadap prediksi perbandingan wajah untuk memastikan bahwa hak-hak sipil seseorang tidak dilanggar.

- Gunakan ambang batas kemiripan yang lebih rendah dari 99% untuk skenario yang mendapat manfaat dari serangkaian kecocokan potensial yang lebih besar. Contohnya adalah menemukan orang hilang. Jika perlu, Anda dapat menggunakan atribut respons Kemiripan untuk menentukan seberapa mirip potensi kecocokan dengan orang yang ingin Anda kenali.
- Memiliki rencana untuk kecocokan wajah positif palsu yang dihasilkan oleh Amazon Rekognition. Misalnya, tingkatkan pencocokan dengan menggunakan beberapa gambar dari orang yang sama saat Anda membuat indeks dengan [IndexFaces](#) operasi. Untuk informasi selengkapnya, lihat [Panduan untuk menggunakan IndexFaces](#).

Dalam kasus penggunaan lain (seperti media sosial), kami sarankan untuk menggunakan penilaian terbaik Anda untuk menilai apakah hasil Amazon Rekognition memerlukan tinjauan manusia. Tergantung pada persyaratan aplikasi Anda, ambang batas kemiripan juga bisa lebih rendah.

Membuat koleksi

Anda dapat menggunakan [CreateCollection](#) operasi untuk membuat koleksi.

Untuk informasi selengkapnya, lihat [Mengelola koleksi](#).

Membuat koleksi (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan AmazonRekognitionFullAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi CreateCollection.

Java

Contoh berikut membuat koleksi dan menampilkan Amazon Resource Name (ARN)nya.

Ubah nilai `collectionId` untuk nama koleksi yang ingin Anda buat.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)
```

```
package aws.example.rekognition.image;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.CreateCollectionRequest;
import com.amazonaws.services.rekognition.model.CreateCollectionResult;

public class CreateCollection {

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

        AmazonRekognition rekognitionClient =
        AmazonRekognitionClientBuilder.defaultClient();

        String collectionId = "MyCollection";
        System.out.println("Creating collection: " +
            collectionId );

        CreateCollectionRequest request = new CreateCollectionRequest()
            .withCollectionId(collectionId);

        CreateCollectionResult createCollectionResult =
        rekognitionClient.createCollection(request);
        System.out.println("CollectionArn : " +
            createCollectionResult.getCollectionArn());
        System.out.println("Status code : " +
            createCollectionResult.getStatusCode().toString());

    }
}
```


Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
//snippet-start:[rekognition.java2.create_collection.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import
    software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
//snippet-end:[rekognition.java2.create_collection.import]

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

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "  <collectionName> \n\n" +
            "Where:\n" +
            "  collectionName - The name of the collection. \n\n";

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

```
String collectionId = args[0];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
    .build();

System.out.println("Creating collection: " +collectionId);
createMyCollection(rekClient, collectionId );
rekClient.close();
}

// snippet-start:[rekognition.java2.create_collection.main]
public static void createMyCollection(RekognitionClient rekClient,String
collectionId ) {

    try {
        CreateCollectionRequest collectionRequest =
CreateCollectionRequest.builder()
            .collectionId(collectionId)
            .build();

        CreateCollectionResponse collectionResponse =
rekClient.createCollection(collectionRequest);
        System.out.println("CollectionArn: " +
collectionResponse.collectionArn());
        System.out.println("Status code: " +
collectionResponse.statusCode().toString());

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

AWS CLI

AWS CLI Perintah ini menampilkan output JSON untuk operasi create-collection CLI.

Ganti nilai collection-id menjadi nama koleksi yang ingin Anda buat.

Ganti nilai `profile_name` dengan nama profil pengembang Anda.

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

Python

Contoh berikut membuat koleksi dan menampilkan Amazon Resource Name (ARN)nya.

Ubah nilai `collection_id` menjadi nama koleksi yang ingin Anda buat. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/  
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
import boto3  
  
def create_collection(collection_id):  
    session = boto3.Session(profile_name='profile-name')  
    client = session.client('rekognition')  
  
    # Create a collection  
    print('Creating collection:' + collection_id)  
    response = client.create_collection(CollectionId=collection_id)  
    print('Collection ARN: ' + response['CollectionArn'])  
    print('Status code: ' + str(response['StatusCode']))  
    print('Done...')  
  
def main():  
    collection_id = "collection-id"  
    create_collection(collection_id)  
  
if __name__ == "__main__":  
    main()
```

.NET

Contoh berikut membuat koleksi dan menampilkan Amazon Resource Name (ARN)nya.

Ubah nilai `collectionId` menjadi nama koleksi yang ingin Anda buat.

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

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

public class CreateCollection
{
    public static void Example()
    {
        AmazonRekognitionClient rekognitionClient = new
AmazonRekognitionClient();

        String collectionId = "MyCollection";
        Console.WriteLine("Creating collection: " + collectionId);

        CreateCollectionRequest createCollectionRequest = new
CreateCollectionRequest()
        {
            CollectionId = collectionId
        };

        CreateCollectionResponse createCollectionResponse =
rekognitionClient.CreateCollection(createCollectionRequest);
        Console.WriteLine("CollectionArn : " +
createCollectionResponse.CollectionArn);
        Console.WriteLine("Status code : " +
createCollectionResponse.StatusCode);

    }
}
```

Node.JS

Dalam contoh berikut, ganti nilai `region` dengan nama wilayah yang terkait dengan akun Anda dan ganti nilai `collectionName` dengan nama koleksi yang diinginkan.

Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import { CreateCollectionCommand} from "@aws-sdk/client-rekognition";
import { RekognitionClient } from "@aws-sdk/client-rekognition";
import {fromIni} from '@aws-sdk/credential-providers';

// Set the AWS Region.
const REGION = "region-name"; //e.g. "us-east-1"
// Set the profile name
const profileName = "profile-name"
// Name the collection
const collectionName = "collection-name"
const rekogClient = new RekognitionClient({region: REGION,
  credentials: fromIni({profile: profileName,}),
});

const createCollection = async (collectionName) => {
  try {
    console.log(`Creating collection: ${collectionName}`)
    const data = await rekogClient.send(new
CreateCollectionCommand({CollectionId: collectionName}));
    console.log("Collection ARN:")
    console.log(data.CollectionARN)
    console.log("Status Code:")
    console.log(String(data.StatusCode))
    console.log("Success.", data);
    return data;
  } catch (err) {
    console.log("Error", err.stack);
  }
};

createCollection(collectionName)
```

CreateCollection permintaan operasi

Input pada `CreateCollection` adalah nama koleksi yang ingin Anda buat.

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

CreateCollection respon operasi

Amazon Rekognition membuat koleksi dan mengembalikan Amazon Resource Name (ARN) dari koleksi yang baru dibuat.

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

Penandaan koleksi

Anda dapat mengidentifikasi, mengatur, mencari, dan memfilter koleksi Amazon Rekognition menggunakan tanda. Setiap tanda adalah label yang terdiri dari kunci dan nilai yang ditentukan pengguna.

Anda juga dapat menggunakan tanda untuk mengendalikan akses untuk koleksi dengan menggunakan Identity and Access Management (IAM). Untuk informasi selengkapnya, lihat [Mengontrol akses ke AWS sumber daya menggunakan tag sumber daya](#).

Topik

- [Tambahkan tanda ke koleksi baru](#)
- [Tambahkan tanda ke koleksi yang sudah ada](#)
- [Daftar tanda dalam koleksi](#)
- [Hapus tanda dari koleksi](#)

Tambahkan tanda ke koleksi baru

Anda dapat menambahkan tanda ke koleksi karena Anda membuatnya menggunakan operasi `CreateCollection`. Tentukan satu tanda atau lebih di parameter input array `Tags`.

AWS CLI

Ganti nilai `profile_name` dengan nama profil pengembang Anda.

```
aws rekognition create-collection --collection-id "collection-name" --tags
{"key1":"value1","key2":"value2"} --profile profile-name
```

Untuk perangkat Windows:

```
aws rekognition create-collection --collection-id "collection-name" --tags
{"key1\\":\\"value1\\",\\"key2\\":\\"value2\\"} --profile profile-name
```

Python

Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
import boto3

def create_collection(collection_id):
    client = boto3.client('rekognition')

    # Create a collection
    print('Creating collection:' + collection_id)
    response = client.create_collection(CollectionId=collection_id)
    print('Collection ARN: ' + response['CollectionArn'])
    print('Status code: ' + str(response['StatusCode']))
    print('Done...')

def main():
    collection_id = 'NewCollectionName'
    create_collection(collection_id)

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

Tambahkan tanda ke koleksi yang sudah ada

Untuk menambahkan satu tanda atau lebih ke koleksi yang sudah ada, gunakan operasi `TagResource`. Tentukan Amazon Resource Name (ARN) koleksi (`ResourceArn`) dan tanda (`Tags`) yang ingin Anda tambahkan. Contoh berikut menunjukkan cara menambahkan dua tanda.

AWS CLI

Ganti nilai `profile_name` dengan nama profil pengembang Anda.

```
aws rekognition tag-resource --resource-arn collection-arn --tags
{"key1":"value1","key2":"value2"} --profile profile-name
```

Untuk perangkat Windows:

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

Python

Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import boto3

def create_tag(collection_id):
    session = boto3.Session(profile_name='default')
    client = session.client('rekognition')
    response = client.tag_resource(ResourceArn=collection_id,
                                  Tags={
                                      "KeyName": "ValueName"
                                  })

    print(response)
    if "'HTTPStatusCode': 200" in str(response):
        print("Success!!")
```



```
def main():
    collection_arn = "collection-arn"
    create_tag(collection_arn)

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

Note

Jika Anda tidak tahu koleksi Nama Sumber Daya Amazon, Anda dapat menggunakan operasi DescribeCollection.

Daftar tanda dalam koleksi

Untuk mencantumkan tanda terlampir di koleksi, gunakan operasi ListTagsForResource dan tentukan ARN koleksi (ResourceArn). Respons adalah peta kunci dan nilai-nilai tanda yang terlampir pada koleksi tertentu.

AWS CLI

Ganti nilai profile_name dengan nama profil pengembang Anda.

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

Python

Ganti nilai profile_name di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
import boto3

def list_tags():
    client = boto3.client('rekognition')
    response =
    client.list_tags_for_resource(ResourceArn="arn:aws:rekognition:region-
name:5498347593847598:collection/NewCollectionName")
    print(response)
```

```
def main():
    list_tags()

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

Output menampilkan daftar tanda yang terlampir pada koleksi:

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

Hapus tanda dari koleksi

Untuk menghapus satu tanda atau lebih dari koleksi, gunakan operasi `UntagResource`. Tentukan ARN (`ResourceArn`) model dan kunci tanda (`Tag-Keys`) yang ingin Anda hapus.

AWS CLI

Ganti nilai `profile_name` dengan nama profil pengembang Anda.

```
aws rekognition untag-resource --resource-arn resource-arn --profile profile-name --
tag-keys "key1" "key2"
```

Atau, Anda dapat menentukan tanda-kunci dalam format ini:

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

Python

Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
import boto3
```

```
def list_tags():
    client = boto3.client('rekognition')
    response = client.untag_resource(ResourceArn="arn:aws:rekognition:region-
name:5498347593847598:collection/NewCollectionName", TagKeys=['KeyName'])
    print(response)

def main():
    list_tags()

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

Mencantumkan koleksi

Anda dapat menggunakan [ListCollections](#) operasi untuk membuat daftar koleksi di wilayah yang Anda gunakan.

Untuk informasi selengkapnya, lihat [Mengelola koleksi](#).

Untuk daftar koleksi (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan AmazonRekognitionFullAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi ListCollections.

Java

Contoh berikut mencantumkan koleksi di wilayah saat ini.

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

package aws.example.rekognition.image;

import java.util.List;
```

```
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.ListCollectionsRequest;
import com.amazonaws.services.rekognition.model.ListCollectionsResult;

public class ListCollections {

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

        AmazonRekognition amazonRekognition =
        AmazonRekognitionClientBuilder.defaultClient();

        System.out.println("Listing collections");
        int limit = 10;
        ListCollectionsResult listCollectionsResult = null;
        String paginationToken = null;
        do {
            if (listCollectionsResult != null) {
                paginationToken = listCollectionsResult.getNextToken();
            }
            ListCollectionsRequest listCollectionsRequest = new
ListCollectionsRequest()
                .withMaxResults(limit)
                .withNextToken(paginationToken);

listCollectionsResult=amazonRekognition.listCollections(listCollectionsRequest);

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

    }
}
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
//snippet-start:[rekognition.java2.list_collections.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import
    software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;
//snippet-end:[rekognition.java2.list_collections.import]

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

    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
            .build();

        System.out.println("Listing collections");
        listAllCollections(rekClient);
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.list_collections.main]
    public static void listAllCollections(RekognitionClient rekClient) {
        try {
```

```
ListCollectionsRequest listCollectionsRequest =
ListCollectionsRequest.builder()
    .maxResults(10)
    .build();

ListCollectionsResponse response =
rekClient.listCollections(listCollectionsRequest);
List<String> collectionIds = response.collectionIds();
for (String resultId : collectionIds) {
    System.out.println(resultId);
}

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

AWS CLI

AWS CLI Perintah ini menampilkan output JSON untuk operasi `list-collections` CLI. Ganti nilai `profile_name` dengan nama profil pengembang Anda.

```
aws rekognition list-collections --profile profile-name
```

Python

Contoh berikut mencantumkan koleksi di wilayah saat ini.

Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import boto3

def list_collections():
```

```
max_results=2

client=boto3.client('rekognition')

#Display all the collections
print('Displaying collections...')
response=client.list_collections(MaxResults=max_results)
collection_count=0
done=False

while done==False:
    collections=response['CollectionIds']

    for collection in collections:
        print (collection)
        collection_count+=1
    if 'NextToken' in response:
        nextToken=response['NextToken']

response=client.list_collections(NextToken=nextToken,MaxResults=max_results)

    else:
        done=True

    return collection_count

def main():

    collection_count=list_collections()
    print("collections: " + str(collection_count))
if __name__ == "__main__":
    main()
```

.NET

Contoh berikut mencantumkan koleksi di wilayah saat ini.

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

using System;
using Amazon.Rekognition;
```

```
using Amazon.Rekognition.Model;

public class ListCollections
{
    public static void Example()
    {
        AmazonRekognitionClient rekognitionClient = new
AmazonRekognitionClient();

        Console.WriteLine("Listing collections");
        int limit = 10;

        ListCollectionsResponse listCollectionsResponse = null;
        String paginationToken = null;
        do
        {
            if (listCollectionsResponse != null)
                paginationToken = listCollectionsResponse.NextToken;

            ListCollectionsRequest listCollectionsRequest = new
ListCollectionsRequest()
            {
                MaxResults = limit,
                NextToken = paginationToken
            };

            listCollectionsResponse =
rekognitionClient.ListCollections(listCollectionsRequest);

            foreach (String resultId in listCollectionsResponse.CollectionIds)
                Console.WriteLine(resultId);
        } while (listCollectionsResponse != null &&
listCollectionsResponse.NextToken != null);
    }
}
```

Node.js

Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```



```
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import { ListCollectionsCommand } from "@aws-sdk/client-rekognition";
import { RekognitionClient } from "@aws-sdk/client-rekognition";
import {fromIni} from '@aws-sdk/credential-providers';

// Set the AWS Region.
const REGION = "region-name"; //e.g. "us-east-1"
// Set the profile name
const profileName = "profile-name"
// Name the collection
const rekogClient = new RekognitionClient({region: REGION,
  credentials: fromIni({profile: profileName,}),
});

const listCollection = async () => {
  var max_results = 10
  console.log("Displaying collections:")
  var response = await rekogClient.send(new ListCollectionsCommand({MaxResults:
max_results}))
  var collection_count = 0
  var done = false
  while (done == false){
    var collections = response.CollectionIds
    collections.forEach(collection => {
      console.log(collection)
      collection_count += 1
    });
    return collection_count
  }
}

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

ListCollections permintaan operasi

Input ke ListCollections adalah jumlah maksimum koleksi yang akan dikembalikan.

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

```
}
```

Jika respons memiliki lebih banyak koleksi daripada yang diminta oleh `MaxResults`, token dikembalikan yang dapat Anda gunakan untuk mendapatkan set hasil berikutnya, dalam panggilan berikutnya `ListCollections`. Sebagai contoh:

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

ListCollections respon operasi

Amazon Rekognition mengembalikan array koleksi (`CollectionIds`). Array (`FaceModelVersions`) terpisah menyediakan versi model wajah yang digunakan untuk menganalisis wajah di setiap koleksi. Misalnya, dalam respons JSON berikut, koleksi `MyCollection` menganalisis wajah menggunakan versi 2.0 model wajah. Koleksi `AnotherCollection` menggunakan versi 3.0 model wajah. Untuk informasi selengkapnya, lihat [Versioning model](#).

`NextToken` adalah token yang digunakan untuk mendapatkan serangkaian hasil berikutnya, dalam panggilan berikutnya untuk `ListCollections`.

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

Menjelaskan koleksi

Anda dapat menggunakan [DescribeCollection](#) operasi untuk mendapatkan informasi berikut tentang koleksi:

- Jumlah wajah yang diindeks ke dalam koleksi, .
- Versi model yang digunakan dengan koleksi. Untuk informasi selengkapnya, lihat [the section called “Versioning model”](#).
- Amazon Resource Name (ARN) dari koleksi.
- Tanggal dan waktu pembuatan koleksi.

Menjelaskan koleksi (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan AmazonRekognitionFullAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi DescribeCollection.

Java

Contoh ini menjelaskan suatu koleksi.

Ubah nilai `collectionId` ke ID koleksi yang diinginkan.

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

package com.amazonaws.samples;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.DescribeCollectionRequest;
import com.amazonaws.services.rekognition.model.DescribeCollectionResult;

public class DescribeCollection {

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

        String collectionId = "CollectionID";
```

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

System.out.println("Describing collection: " +
    collectionId );

DescribeCollectionRequest request = new DescribeCollectionRequest()
    .withCollectionId(collectionId);

DescribeCollectionResult describeCollectionResult =
rekognitionClient.describeCollection(request);
System.out.println("Collection Arn : " +
    describeCollectionResult.getCollectionARN());
System.out.println("Face count : " +
    describeCollectionResult.getFaceCount().toString());
System.out.println("Face model version : " +
    describeCollectionResult.getFaceModelVersion());
System.out.println("Created : " +
    describeCollectionResult.getCreationTimestamp().toString());

}

}
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
//snippet-end:[rekognition.java2.describe_collection.import]
```

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

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "  <collectionName>\n\n" +
            "Where:\n" +
            "  collectionName - The name of the Amazon Rekognition collection. \n
\n";

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

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
            .build();

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.describe_collection.main]
    public static void describeColl(RekognitionClient rekClient, String
collectionName) {

        try {
            DescribeCollectionRequest describeCollectionRequest =
DescribeCollectionRequest.builder()
                .collectionId(collectionName)
```

```
        .build();

        DescribeCollectionResponse describeCollectionResponse =
rekClient.describeCollection(describeCollectionRequest);
        System.out.println("Collection Arn : " +
describeCollectionResponse.collectionARN());
        System.out.println("Created : " +
describeCollectionResponse.creationTimestamp().toString());

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

AWS CLI

AWS CLI Perintah ini menampilkan output JSON untuk operasi `describe-collection` CLI. Ubah nilai `collection-id` menjadi ID koleksi yang diinginkan. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
aws rekognition describe-collection --collection-id collection-name --profile
profile-name
```

Python

Contoh ini menjelaskan suatu koleksi.

Ubah nilai `collection_id` ke ID koleksi yang diinginkan. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import boto3
from botocore.exceptions import ClientError
```

```
def describe_collection(collection_id):

    print('Attempting to describe collection ' + collection_id)

    session = boto3.Session(profile_name='default')
    client = session.client('rekognition')

    try:
        response = client.describe_collection(CollectionId=collection_id)
        print("Collection Arn: " + response['CollectionARN'])
        print("Face Count: " + str(response['FaceCount']))
        print("Face Model Version: " + response['FaceModelVersion'])
        print("Timestamp: " + str(response['CreationTimestamp']))

    except ClientError as e:
        if e.response['Error']['Code'] == 'ResourceNotFoundException':
            print('The collection ' + collection_id + ' was not found ')
        else:
            print('Error other than Not Found occurred: ' + e.response['Error']
                  ['Message'])
            print('Done...')

def main():
    collection_id = 'collection-name'
    describe_collection(collection_id)

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

.NET

Contoh ini menjelaskan suatu koleksi.

Ubah nilai `collectionId` ke ID koleksi yang diinginkan.

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

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

```
public class DescribeCollection
{
    public static void Example()
    {
        AmazonRekognitionClient rekognitionClient = new
AmazonRekognitionClient();

        String collectionId = "CollectionID";
        Console.WriteLine("Describing collection: " + collectionId);

        DescribeCollectionRequest describeCollectionRequest = new
DescribeCollectionRequest()
        {
            CollectionId = collectionId
        };

        DescribeCollectionResponse describeCollectionResponse =
rekognitionClient.DescribeCollection(describeCollectionRequest);
        Console.WriteLine("Collection ARN: " +
describeCollectionResponse.CollectionARN);
        Console.WriteLine("Face count: " +
describeCollectionResponse.FaceCount);
        Console.WriteLine("Face model version: " +
describeCollectionResponse.FaceModelVersion);
        Console.WriteLine("Created: " +
describeCollectionResponse.CreationTimestamp);
    }
}
```

Node.js

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

import { DescribeCollectionCommand } from "@aws-sdk/client-rekognition";
import { RekognitionClient } from "@aws-sdk/client-rekognition";
import {fromIni} from '@aws-sdk/credential-providers';

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



```
const profileName = "profile-name"
// Name the collection
const rekogClient = new RekognitionClient({region: REGION,
  credentials: fromIni({profile: profileName,}),
});

// Name the collection
const collection_name = "collection-name"

const describeCollection = async (collectionName) => {
  try {
    console.log(`Attempting to describe collection named - ${collectionName}`)
    var response = await rekogClient.send(new
DescribeCollectionCommand({CollectionId: collectionName}))
    console.log('Collection Arn:')
    console.log(response.CollectionARN)
    console.log('Face Count:')
    console.log(response.FaceCount)
    console.log('Face Model Version:')
    console.log(response.FaceModelVersion)
    console.log('Timestamp:')
    console.log(response.CreationTimestamp)
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};

describeCollection(collection_name)
```

DescribeCollection permintaan operasi

Input di `DescribeCollection` adalah ID koleksi yang diinginkan, seperti yang ditunjukkan pada contoh JSON berikut.

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

DescribeCollectionrespon operasi

Respons meliputi:

- Jumlah wajah yang diindeks ke dalam koleksi, `FaceCount`.
- Versi model wajah yang digunakan dengan koleksi, `FaceModelVersion`. Untuk informasi selengkapnya, lihat [the section called "Versioning model"](#).
- Nama Sumber Daya Amazon koleksi, `CollectionARN`.
- Waktu dan tanggal pembuatan koleksi, `CreationTimestamp`. Nilai `CreationTimestamp` adalah jumlah milidetik sejak periode jangka waktu Unix hingga pembuatan koleksi. Jangka waktu periode Unix adalah 00:00:00 Coordinated Universal Time (UTC), Kamis, 1 Januari 1970. Untuk informasi selengkapnya, lihat [Waktu Unix](#).

```
{
  "CollectionARN": "arn:aws:rekognition:us-east-1:nnnnnnnnnnn:collection/MyCollection",
  "CreationTimestamp": 1.533422155042E9,
  "FaceCount": 200,
  "UserCount" : 20,
  "FaceModelVersion": "1.0"
}
```

Menghapus koleksi

Anda dapat menggunakan [DeleteCollection](#) operasi untuk menghapus koleksi.

Untuk informasi selengkapnya, lihat [Mengelola koleksi](#).

Untuk menghapus koleksi (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan `AmazonRekognitionFullAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi `DeleteCollection`.

Java

Contoh ini menjelaskan suatu koleksi.

Ubah nilai `collectionId` menjadi koleksi yang ingin Anda hapus.

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

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.DeleteCollectionRequest;
import com.amazonaws.services.rekognition.model.DeleteCollectionResult;

public class DeleteCollection {

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

        AmazonRekognition rekognitionClient =
        AmazonRekognitionClientBuilder.defaultClient();

        String collectionId = "MyCollection";

        System.out.println("Deleting collections");

        DeleteCollectionRequest request = new DeleteCollectionRequest()
            .withCollectionId(collectionId);
        DeleteCollectionResult deleteCollectionResult =
        rekognitionClient.deleteCollection(request);

        System.out.println(collectionId + ": " +
        deleteCollectionResult.getStatusCode()
            .toString());

    }

}
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
// snippet-start:[rekognition.java2.delete_collection.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
// snippet-end:[rekognition.java2.delete_collection.import]

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

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "  <collectionId> \n\n" +
            "Where:\n" +
            "  collectionId - The id of the collection to delete. \n\n";

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

        String collectionId = args[0];
```

```
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.delete_collection.main]
    public static void deleteMyCollection(RekognitionClient rekClient, String
collectionId ) {

        try {
            DeleteCollectionRequest deleteCollectionRequest =
DeleteCollectionRequest.builder()
                .collectionId(collectionId)
                .build();

            DeleteCollectionResponse deleteCollectionResponse =
rekClient.deleteCollection(deleteCollectionRequest);
            System.out.println(collectionId + ": " +
deleteCollectionResponse.statusCode().toString());

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

AWS CLI

AWS CLI Perintah ini menampilkan output JSON untuk operasi delete-collection CLI. Ganti nilai collection-id dengan nama koleksi yang ingin Anda hapus. Ganti nilai profile_name di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
aws rekognition delete-collection --collection-id collection-name --profile
profile-name
```

Python

Contoh ini menjelaskan suatu koleksi.

Ubah nilai `collection_id` menjadi koleksi yang ingin Anda hapus. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import boto3
from botocore.exceptions import ClientError

def delete_collection(collection_id):

    print('Attempting to delete collection ' + collection_id)
    session = boto3.Session(profile_name='default')
    client = session.client('rekognition')

    status_code = 0

    try:
        response = client.delete_collection(CollectionId=collection_id)
        status_code = response['StatusCode']

    except ClientError as e:
        if e.response['Error']['Code'] == 'ResourceNotFoundException':
            print('The collection ' + collection_id + ' was not found ')
        else:
            print('Error other than Not Found occurred: ' + e.response['Error']
['Message'])
            status_code = e.response['ResponseMetadata']['HTTPStatusCode']
    return (status_code)

def main():

    collection_id = 'collection-name'
```

```
status_code = delete_collection(collection_id)
print('Status code: ' + str(status_code))

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

.NET

Contoh ini menjelaskan suatu koleksi.

Ubah nilai `collectionId` menjadi koleksi yang ingin Anda hapus.

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

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

public class DeleteCollection
{
    public static void Example()
    {
        AmazonRekognitionClient rekognitionClient = new
AmazonRekognitionClient();

        String collectionId = "MyCollection";
        Console.WriteLine("Deleting collection: " + collectionId);

        DeleteCollectionRequest deleteCollectionRequest = new
DeleteCollectionRequest()
        {
            CollectionId = collectionId
        };

        DeleteCollectionResponse deleteCollectionResponse =
rekognitionClient.DeleteCollection(deleteCollectionRequest);
        Console.WriteLine(collectionId + ": " +
deleteCollectionResponse.StatusCode);
    }
}
```

Node.js

Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import { DeleteCollectionCommand } from "@aws-sdk/client-rekognition";
import { RekognitionClient } from "@aws-sdk/client-rekognition";
import {fromIni} from '@aws-sdk/credential-providers';

// Set the AWS Region.
const REGION = "region-name"; //e.g. "us-east-1"
// Set the profile name
const profileName = "profile-name"
// Name the collection
const rekogClient = new RekognitionClient({region: REGION,
  credentials: fromIni({profile: profileName,}),
});

// Name the collection
const collection_name = "collection-name"

const deleteCollection = async (collectionName) => {
  try {
    console.log(`Attempting to delete collection named - ${collectionName}`)
    var response = await rekogClient.send(new
DeleteCollectionCommand({CollectionId: collectionName}))
    var status_code = response.StatusCode
    if (status_code = 200){
      console.log("Collection successfully deleted.")
    }
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};

deleteCollection(collection_name)
```


DeleteCollection permintaan operasi

Input pada DeleteCollection adalah ID koleksi yang akan dihapus, seperti yang ditunjukkan pada contoh JSON berikut.

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

DeleteCollection respon operasi

Respon DeleteCollection berisi kode status HTTP yang menunjukkan keberhasilan atau kegagalan operasi. 200 dikirimkan jika koleksi berhasil dihapus.

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

Menambahkan wajah ke koleksi

Anda dapat menggunakan [IndexFaces](#) operasi untuk mendeteksi wajah dalam gambar dan menambahkannya ke koleksi. Untuk setiap wajah yang terdeteksi, Amazon Rekognition mengekstraksi fitur wajah dan menyimpan informasi fitur dalam basis data. Selain itu, perintah tersebut menyimpan metadata untuk setiap wajah yang terdeteksi dalam koleksi wajah tertentu. Amazon Rekognition tidak menyimpan bit citra yang sebenarnya.

Untuk informasi tentang memberikan wajah yang cocok untuk pengindeksan, lihat [Rekomendasi untuk citra input perbandingan wajah](#).

Untuk setiap wajah, operasi IndexFaces menyimpan informasi berikut:

- **Fitur wajah multidimensi** — IndexFaces menggunakan analisis wajah untuk mengekstraksi informasi multidimensi tentang fitur wajah dan menyimpan informasi tersebut dalam koleksi wajah. Anda tidak dapat mengakses informasi ini secara langsung. Namun, Amazon Rekognition menggunakan informasi ini ketika mencari koleksi wajah untuk mencocokkan wajah.
- **Metadata** — Metadata untuk setiap wajah mencakup kotak pembatas, tingkat kepercayaan (bahwa kotak pembatas berisi wajah), ID yang ditetapkan oleh Amazon Rekognition (ID wajah dan ID citra), dan ID citra eksternal (jika Anda memberikannya) dalam permintaan. Informasi ini dikembalikan

kepada Anda sebagai respons atas panggilan API `IndexFaces`. Misalnya, lihat elemen `face` dalam respons contoh berikut.

Layanan akan mengembalikan metadata ini sebagai respons terhadap panggilan API berikut:

- [ListFaces](#)
- Operasi pencarian wajah — Respons untuk [SearchFaces](#) dan [SearchFacesByImage](#) mengembalikan kepercayaan pada kecocokan untuk setiap wajah yang cocok, bersama dengan metadata wajah yang cocok ini.

Jumlah muka yang terindeks oleh `IndexFaces` tergantung pada versi model deteksi wajah yang terkait dengan koleksi input. Untuk informasi selengkapnya, lihat [Versioning model](#).

Informasi tentang wajah yang diindeks dikembalikan dalam array objek. [FaceRecord](#)

Anda mungkin ingin mengaitkan wajah yang terindeks dengan citra yang terdeteksi dengan wajah tersebut. Misalnya, Anda mungkin ingin mempertahankan indeks sisi klien citra dan wajah dalam citra. Untuk mengaitkan wajah dengan citra, tentukan ID citra di parameter permintaan `ExternalImageId`. ID citra dapat berupa nama file atau ID lain yang Anda buat.

Selain informasi sebelumnya bahwa API tetap ada di koleksi wajah, API juga mengembalikan detail wajah yang tidak tersimpan dalam koleksi. (Lihat elemen `faceDetail` dalam respons contoh berikut).

Note

`DetectFaces` menampilkan informasi yang sama, sehingga Anda tidak perlu memanggil `DetectFaces` dan `IndexFaces` untuk citra yang sama.

Memfilter wajah

`IndexFaces` Operasi ini memungkinkan Anda untuk memfilter wajah yang diindeks dari gambar. Dengan `IndexFaces` Anda dapat menentukan jumlah maksimum wajah untuk diberi indeks, atau Anda dapat memilih untuk hanya mengindeks wajah terdeteksi dengan kualitas tinggi.

Anda dapat menentukan jumlah maksimum wajah yang diindeks oleh `IndexFaces` dengan menggunakan parameter input `MaxFaces`. Hal ini berguna apabila Anda ingin mengindeks wajah

terbesar dalam citra dan tidak ingin mengindeks wajah yang lebih kecil, seperti wajah orang yang berdiri di baliknya.

Secara default, IndexFaces memilih bilah kualitas yang digunakan untuk memfilter wajah. Anda dapat menggunakan parameter input `QualityFilter` untuk secara eksplisit mengatur bilah kualitas. Nilainya adalah:

- `AUTO` — Amazon Rekognition memilih bar kualitas yang digunakan untuk memfilter wajah (nilai default).
- `LOW` — Semua kecuali wajah kualitas terendah diindeks.
- `MEDIUM`
- `HIGH` — Hanya wajah kualitas tertinggi yang diindeks.
- `NONE` - Tidak ada wajah yang disaring berdasarkan kualitas.

IndexFaces memfilter wajah karena alasan berikut:

- Wajah terlalu kecil dibandingkan dengan dimensi citra.
- Wajah terlalu buram.
- Citra terlalu gelap.
- Wajah memiliki pose yang ekstrem.
- Wajah tidak memiliki cukup detail agar cocok untuk pencarian wajah.

Note

Untuk menggunakan penyaringan berkualitas, Anda memerlukan koleksi yang terkait dengan model wajah versi 3 atau lebih tinggi. Untuk mendapatkan versi model wajah yang terkait dengan koleksi, hubungi [DescribeCollection](#).

Informasi tentang wajah yang tidak diindeks oleh IndexFaces dikembalikan dalam array objek. [UnindexedFace](#) Array Reasons berisi daftar alasan mengapa wajah tidak diindeks. Misalnya, nilai `EXCEEDS_MAX_FACES` adalah wajah yang tidak diindeks karena jumlah wajah yang ditentukan oleh `MaxFaces` telah terdeteksi.

Untuk informasi selengkapnya, lihat [Mengelola wajah dalam koleksi](#).

Menambahkan wajah ke koleksi (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan `AmazonRekognitionFullAccess` dan `AmazonS3ReadOnlyAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Unggah citra (berisi satu wajah atau lebih) ke bucket Amazon S3.

Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

3. Gunakan contoh berikut untuk memanggil operasi `IndexFaces`.

Java

Contoh ini menampilkan pengenalan wajah untuk wajah yang ditambahkan ke koleksi.

Ubah nilai `collectionId` untuk nama koleksi yang ingin ditambahkan wajah. Ganti nilai-nilai `bucket` dan `photo` dengan nama bucket Amazon S3 dan citra yang Anda gunakan di langkah 2. Parameter `.withMaxFaces(1)` membatasi jumlah wajah yang diindeks menjadi 1. Hapus atau ubah nilainya sesuai dengan kebutuhan Anda.

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

package aws.example.rekognition.image;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.FaceRecord;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.IndexFacesRequest;
import com.amazonaws.services.rekognition.model.IndexFacesResult;
import com.amazonaws.services.rekognition.model.QualityFilter;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.UnindexedFace;
import java.util.List;
```

```
public class AddFacesToCollection {
    public static final String collectionId = "MyCollection";
    public static final String bucket = "bucket";
    public static final String photo = "input.jpg";

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

        AmazonRekognition rekognitionClient =
AmazonRekognitionClientBuilder.defaultClient();

        Image image = new Image()
            .withS3Object(new S3Object()
                .withBucket(bucket)
                .withName(photo));

        IndexFacesRequest indexFacesRequest = new IndexFacesRequest()
            .withImage(image)
            .withQualityFilter(QualityFilter.AUTO)
            .withMaxFaces(1)
            .withCollectionId(collectionId)
            .withExternalImageId(photo)
            .withDetectionAttributes("DEFAULT");

        IndexFacesResult indexFacesResult =
rekognitionClient.indexFaces(indexFacesRequest);

        System.out.println("Results for " + photo);
        System.out.println("Faces indexed:");
        List<FaceRecord> faceRecords = indexFacesResult.getFaceRecords();
        for (FaceRecord faceRecord : faceRecords) {
            System.out.println("  Face ID: " +
faceRecord.getFace().getFaceId());
            System.out.println("  Location:" +
faceRecord.getFaceDetail().getBoundingBox().toString());
        }

        List<UnindexedFace> unindexedFaces =
indexFacesResult.getUnindexedFaces();
        System.out.println("Faces not indexed:");
        for (UnindexedFace unindexedFace : unindexedFaces) {
            System.out.println("  Location:" +
unindexedFace.getFaceDetail().getBoundingBox().toString());
            System.out.println("  Reasons:");
            for (String reason : unindexedFace.getReasons()) {
```

```
        System.out.println("    " + reason);
    }
}
}
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
//snippet-start:[rekognition.java2.add_faces_collection.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.IndexFacesResponse;
import software.amazon.awssdk.services.rekognition.model.IndexFacesRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.QualityFilter;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceRecord;
import software.amazon.awssdk.services.rekognition.model.UnindexedFace;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Reason;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
//snippet-end:[rekognition.java2.add_faces_collection.import]

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

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

```
final String usage = "\n" +
    "Usage: " +
    "  <collectionId> <sourceImage>\n\n" +
    "Where:\n" +
    "  collectionName - The name of the collection.\n" +
    "  sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png). \n\n";

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

String collectionId = args[0];
String sourceImage = args[1];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
    .build();

addToCollection(rekClient, collectionId, sourceImage);
rekClient.close();
}

// snippet-start:[rekognition.java2.add_faces_collection.main]
public static void addToCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        IndexFacesRequest facesRequest = IndexFacesRequest.builder()
            .collectionId(collectionId)
            .image(souImage)
            .maxFaces(1)
            .qualityFilter(QualityFilter.AUTO)
            .detectionAttributes(Attribute.DEFAULT)
            .build();
```

```

    IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
    System.out.println("Results for the image");
    System.out.println("\n Faces indexed:");
    List<FaceRecord> faceRecords = facesResponse.faceRecords();
    for (FaceRecord faceRecord : faceRecords) {
        System.out.println("  Face ID: " + faceRecord.face().faceId());
        System.out.println("  Location:" +
faceRecord.faceDetail().boundingBox().toString());
    }

    List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
    System.out.println("Faces not indexed:");
    for (UnindexedFace unindexedFace : unindexedFaces) {
        System.out.println("  Location:" +
unindexedFace.faceDetail().boundingBox().toString());
        System.out.println("  Reasons:");
        for (Reason reason : unindexedFace.reasons()) {
            System.out.println("Reason:  " + reason);
        }
    }

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

```

AWS CLI

AWS CLI Perintah ini menampilkan output JSON untuk operasi `index-faces` CLI.

Ganti nilai `collection-id` dengan nama koleksi tempat Anda ingin wajah itu disimpan. Mengganti nilai-nilai Bucket dan Name dengan bucket Amazon S3 dan file citra yang Anda gunakan pada langkah 2. Parameter `max-faces` membatasi jumlah wajah yang diindeks menjadi 1. Hapus atau ubah nilainya sesuai dengan kebutuhan Anda. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.


```
aws rekognition index-faces --image '{"S3object":{"Bucket":"bucket-
name","Name":"file-name"}}' --collection-id "collection-id" \
                                --max-faces 1 --quality-filter "AUTO" --
detection-attributes "ALL" \
                                --external-image-id "example-image.jpg" --
profile profile-name
```

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda alih-alih tanda kutip tunggal dan hindari tanda kutip ganda bagian dalam dengan garis miring terbalik (yaitu\) untuk mengatasi kesalahan parser yang mungkin Anda temui. Sebagai contoh, lihat berikut ini:

```
aws rekognition index-faces --image "{\"S3object\":{\"Bucket\": \"bucket-name\",
\\Name\": \"image-name\"}}\" \
--collection-id "collection-id" --max-faces 1 --quality-filter "AUTO" --
detection-attributes "ALL" \
--external-image-id "example-image.jpg" --profile profile-name
```

Python

Contoh ini menampilkan pengenalan wajah untuk wajah yang ditambahkan ke koleksi.

Ubah nilai `collectionId` untuk nama koleksi yang ingin ditambahkan wajah. Ganti nilai-nilai `bucket` dan `photo` dengan nama bucket Amazon S3 dan citra yang Anda gunakan di langkah 2. Parameter input `MaxFaces` membatasi jumlah wajah yang diindeks menjadi 1. Hapus atau ubah nilainya sesuai dengan kebutuhan Anda. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import boto3

def add_faces_to_collection(bucket, photo, collection_id):

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

```

response = client.index_faces(CollectionId=collection_id,
                             Image={'S3Object': {'Bucket': bucket, 'Name':
photo}},
                             ExternalImageId=photo,
                             MaxFaces=1,
                             QualityFilter="AUTO",
                             DetectionAttributes=['ALL'])

print('Results for ' + photo)
print('Faces indexed:')
for faceRecord in response['FaceRecords']:
    print(' Face ID: ' + faceRecord['Face']['FaceId'])
    print(' Location: {}'.format(faceRecord['Face']['BoundingBox']))

print('Faces not indexed:')
for unindexedFace in response['UnindexedFaces']:
    print(' Location: {}'.format(unindexedFace['FaceDetail']
['BoundingBox']))
    print(' Reasons:')
    for reason in unindexedFace['Reasons']:
        print(' ' + reason)
return len(response['FaceRecords'])

def main():
    bucket = 'bucket-name'
    collection_id = 'collection-id'
    photo = 'photo-name'

    indexed_faces_count = add_faces_to_collection(bucket, photo, collection_id)
    print("Faces indexed count: " + str(indexed_faces_count))

if __name__ == "__main__":
    main()

```

.NET

Contoh ini menampilkan pengenalan wajah untuk wajah yang ditambahkan ke koleksi.

Ubah nilai `collectionId` untuk nama koleksi yang ingin ditambahkan wajah. Ganti nilai-nilai `bucket` dan `photo` dengan nama bucket Amazon S3 dan citra yang Anda gunakan di langkah 2.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)
```

```
using System;
using System.Collections.Generic;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class AddFaces
{
    public static void Example()
    {
        String collectionId = "MyCollection";
        String bucket = "bucket";
        String photo = "input.jpg";

        AmazonRekognitionClient rekognitionClient = new
AmazonRekognitionClient();

        Image image = new Image()
        {
            S3Object = new S3Object()
            {
                Bucket = bucket,
                Name = photo
            }
        };

        IndexFacesRequest indexFacesRequest = new IndexFacesRequest()
        {
            Image = image,
            CollectionId = collectionId,
            ExternalImageId = photo,
            DetectionAttributes = new List<String>(){ "ALL" }
        };

        IndexFacesResponse indexFacesResponse =
rekognitionClient.IndexFaces(indexFacesRequest);

        Console.WriteLine(photo + " added");
        foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
            Console.WriteLine("Face detected: Faceid is " +
                faceRecord.Face.FaceId);
    }
}
```

```
}
```

IndexFaces permintaan operasi

Input ke IndexFaces adalah citra yang akan diindeks dan koleksi untuk ditambahkan wajah atau beberapa wajah.

```
{
  "CollectionId": "MyCollection",
  "Image": {
    "S3Object": {
      "Bucket": "bucket",
      "Name": "input.jpg"
    }
  },
  "ExternalImageId": "input.jpg",
  "DetectionAttributes": [
    "DEFAULT"
  ],
  "MaxFaces": 1,
  "QualityFilter": "AUTO"
}
```

IndexFaces respon operasi

IndexFaces mengembalikan informasi tentang wajah yang terdeteksi dalam citra. Misalnya, respons JSON berikut mencakup atribut deteksi default untuk wajah yang terdeteksi dalam citra input. Contoh tersebut juga menunjukkan wajah yang tidak terindeks karena nilai parameter input MaxFaces telah terlampaui — array Reasons berisi EXCEEDS_MAX_FACES. Jika wajah tidak diindeks karena alasan kualitas, Reasons berisi nilai-nilai seperti KETAJAMAN_RENDAH atau KECERAHAN_RENDAH. Untuk informasi lebih lanjut, lihat [UnindexedFace](#).

```
{
  "FaceModelVersion": "3.0",
  "FaceRecords": [
    {
      "Face": {
        "BoundingBox": {
          "Height": 0.3247932195663452,
          "Left": 0.5055555701255798,
```

```
        "Top": 0.2743072211742401,
        "Width": 0.21444444358348846
    },
    "Confidence": 99.99998474121094,
    "ExternalImageId": "input.jpg",
    "FaceId": "b86e2392-9da1-459b-af68-49118dc16f87",
    "ImageId": "09f43d92-02b6-5cea-8fbd-9f187db2050d"
},
"FaceDetail": {
    "BoundingBox": {
        "Height": 0.3247932195663452,
        "Left": 0.5055555701255798,
        "Top": 0.2743072211742401,
        "Width": 0.21444444358348846
    },
    "Confidence": 99.99998474121094,
    "Landmarks": [
        {
            "Type": "eyeLeft",
            "X": 0.5751981735229492,
            "Y": 0.4010535478591919
        },
        {
            "Type": "eyeRight",
            "X": 0.6511467099189758,
            "Y": 0.4017036259174347
        },
        {
            "Type": "nose",
            "X": 0.6314528584480286,
            "Y": 0.4710812568664551
        },
        {
            "Type": "mouthLeft",
            "X": 0.5879443287849426,
            "Y": 0.5171778798103333
        },
        {
            "Type": "mouthRight",
            "X": 0.6444502472877502,
            "Y": 0.5164633989334106
        }
    ],
    "Pose": {
```

```
        "Pitch": -10.313642501831055,
        "Roll": -1.0316886901855469,
        "Yaw": 18.079818725585938
    },
    "Quality": {
        "Brightness": 71.2919921875,
        "Sharpness": 78.74752044677734
    }
}
],
"OrientationCorrection": "",
"UnindexedFaces": [
    {
        "FaceDetail": {
            "BoundingBox": {
                "Height": 0.1329464465379715,
                "Left": 0.5611110925674438,
                "Top": 0.6832437515258789,
                "Width": 0.08777777850627899
            },
            "Confidence": 92.37225341796875,
            "Landmarks": [
                {
                    "Type": "eyeLeft",
                    "X": 0.5796897411346436,
                    "Y": 0.7452847957611084
                },
                {
                    "Type": "eyeRight",
                    "X": 0.6078574657440186,
                    "Y": 0.742687463760376
                },
                {
                    "Type": "nose",
                    "X": 0.597953200340271,
                    "Y": 0.7620673179626465
                },
                {
                    "Type": "mouthLeft",
                    "X": 0.5884202122688293,
                    "Y": 0.7920381426811218
                },
                {
```

```
        "Type": "mouthRight",
        "X": 0.60627681016922,
        "Y": 0.7919750809669495
      }
    ],
    "Pose": {
      "Pitch": 15.658954620361328,
      "Roll": -4.583454608917236,
      "Yaw": 10.558992385864258
    },
    "Quality": {
      "Brightness": 42.54612350463867,
      "Sharpness": 86.93206024169922
    }
  },
  "Reasons": [
    "EXCEEDS_MAX_FACES"
  ]
}
]
```

Untuk mendapatkan semua informasi wajah, tentukan 'ALL' untuk parameter permintaan `DetectionAttributes`. Sebagai contoh, dalam respons contoh berikut, perhatikan informasi tambahan dalam elemen `faceDetail`, yang tidak tersimpan pada server:

- 25 penanda wajah (dibandingkan dengan hanya lima pada contoh sebelumnya)
- Sepuluh atribut wajah (kacamata, jenggot, oklusi, arah tatapan mata, dan sebagainya)
- Emosi (lihat elemen `emotion`)

Elemen `face` menyediakan metadata yang tersimpan di server.

`FaceModelVersion` adalah versi model wajah yang terkait dengan koleksi. Untuk informasi selengkapnya, lihat [Versioning model](#).

`OrientationCorrection` adalah perkiraan orientasi citra. Informasi koreksi orientasi tidak dikembalikan jika Anda menggunakan versi model deteksi wajah yang lebih tinggi dari versi 3. Untuk informasi selengkapnya, lihat [Mendapatkan orientasi citra dan koordinat kotak pembatas](#).

Respons sampel berikut menunjukkan JSON yang dikembalikan saat menentukan ["SEMUA"]:

```
{
  "FaceModelVersion": "3.0",
  "FaceRecords": [
    {
      "Face": {
        "BoundingBox": {
          "Height": 0.06333333253860474,
          "Left": 0.17185185849666595,
          "Top": 0.7366666793823242,
          "Width": 0.11061728745698929
        },
        "Confidence": 99.99999237060547,
        "ExternalImageId": "input.jpg",
        "FaceId": "578e2e1b-d0b0-493c-aa39-ba476a421a34",
        "ImageId": "9ba38e68-35b6-5509-9d2e-fcffa75d1653"
      },
      "FaceDetail": {
        "AgeRange": {
          "High": 25,
          "Low": 15
        },
        "Beard": {
          "Confidence": 99.98077392578125,
          "Value": false
        },
        "BoundingBox": {
          "Height": 0.06333333253860474,
          "Left": 0.17185185849666595,
          "Top": 0.7366666793823242,
          "Width": 0.11061728745698929
        },
        "Confidence": 99.99999237060547,
        "Emotions": [
          {
            "Confidence": 95.40877532958984,
            "Type": "HAPPY"
          },
          {
            "Confidence": 6.6088080406188965,
            "Type": "CALM"
          },
          {
            "Confidence": 0.7385611534118652,
```



```
        "Type": "SAD"
      }
    ],
    "EyeDirection": {
      "yaw": 16.299732,
      "pitch": -6.407457,
      "confidence": 99.968704
    },
    "Eyeglasses": {
      "Confidence": 99.96795654296875,
      "Value": false
    },
    "EyesOpen": {
      "Confidence": 64.0671157836914,
      "Value": true
    },
    "Gender": {
      "Confidence": 100,
      "Value": "Female"
    },
    "Landmarks": [
      {
        "Type": "eyeLeft",
        "X": 0.21361233294010162,
        "Y": 0.757106363773346
      },
      {
        "Type": "eyeRight",
        "X": 0.2518567442893982,
        "Y": 0.7599404454231262
      },
      {
        "Type": "nose",
        "X": 0.2262365221977234,
        "Y": 0.7711842060089111
      },
      {
        "Type": "mouthLeft",
        "X": 0.2050037682056427,
        "Y": 0.7801263332366943
      },
      {
        "Type": "mouthRight",
        "X": 0.2430567592382431,
```

```
        "Y": 0.7836716771125793
    },
    {
        "Type": "leftPupil",
        "X": 0.2161938101053238,
        "Y": 0.756662905216217
    },
    {
        "Type": "rightPupil",
        "X": 0.2523181438446045,
        "Y": 0.7603650689125061
    },
    {
        "Type": "leftEyeBrowLeft",
        "X": 0.20066319406032562,
        "Y": 0.7501518130302429
    },
    {
        "Type": "leftEyeBrowUp",
        "X": 0.2130996286869049,
        "Y": 0.7480520606040955
    },
    {
        "Type": "leftEyeBrowRight",
        "X": 0.22584207355976105,
        "Y": 0.7504606246948242
    },
    {
        "Type": "rightEyeBrowLeft",
        "X": 0.24509544670581818,
        "Y": 0.7526801824569702
    },
    {
        "Type": "rightEyeBrowUp",
        "X": 0.2582615911960602,
        "Y": 0.7516844868659973
    },
    {
        "Type": "rightEyeBrowRight",
        "X": 0.26881539821624756,
        "Y": 0.7554477453231812
    },
    {
        "Type": "leftEyeLeft",
```

```
        "X": 0.20624476671218872,  
        "Y": 0.7568746209144592  
    },  
    {  
        "Type": "leftEyeRight",  
        "X": 0.22105035185813904,  
        "Y": 0.7582521438598633  
    },  
    {  
        "Type": "leftEyeUp",  
        "X": 0.21401576697826385,  
        "Y": 0.7553104162216187  
    },  
    {  
        "Type": "leftEyeDown",  
        "X": 0.21317370235919952,  
        "Y": 0.7584449648857117  
    },  
    {  
        "Type": "rightEyeLeft",  
        "X": 0.24393919110298157,  
        "Y": 0.7600628137588501  
    },  
    {  
        "Type": "rightEyeRight",  
        "X": 0.2598416209220886,  
        "Y": 0.7605880498886108  
    },  
    {  
        "Type": "rightEyeUp",  
        "X": 0.2519053518772125,  
        "Y": 0.7582084536552429  
    },  
    {  
        "Type": "rightEyeDown",  
        "X": 0.25177454948425293,  
        "Y": 0.7612871527671814  
    },  
    {  
        "Type": "noseLeft",  
        "X": 0.2185886949300766,  
        "Y": 0.774715781211853  
    },  
    {
```

```
        "Type": "noseRight",
        "X": 0.23328955471515656,
        "Y": 0.7759330868721008
    },
    {
        "Type": "mouthUp",
        "X": 0.22446128726005554,
        "Y": 0.7805567383766174
    },
    {
        "Type": "mouthDown",
        "X": 0.22087252140045166,
        "Y": 0.7891407608985901
    }
],
"MouthOpen": {
    "Confidence": 95.87068939208984,
    "Value": false
},
"Mustache": {
    "Confidence": 99.9828109741211,
    "Value": false
},
"Pose": {
    "Pitch": -0.9409101605415344,
    "Roll": 7.233824253082275,
    "Yaw": -2.3602254390716553
},
"Quality": {
    "Brightness": 32.01998519897461,
    "Sharpness": 93.67259216308594
},
"Smile": {
    "Confidence": 86.7142105102539,
    "Value": true
},
"Sunglasses": {
    "Confidence": 97.38925170898438,
    "Value": false
}
}
},
"OrientationCorrection": "ROTATE_0"
```

```
"UnindexedFaces": []  
}
```

Daftar wajah dan pengguna terkait dalam koleksi

Anda dapat menggunakan [ListFaces](#) operasi untuk membuat daftar wajah dan pengguna terkait mereka dalam koleksi.

Untuk informasi selengkapnya, lihat [Mengelola wajah dalam koleksi](#).

Untuk mendaftarkan wajah dalam koleksi (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan AmazonRekognitionFullAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi ListFaces.

Java

Contoh ini menampilkan daftar wajah dalam koleksi.

Ubah nilai `collectionId` menjadi ID koleksi yang diinginkan.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/  
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
package aws.example.rekognition.image;  
import com.amazonaws.services.rekognition.AmazonRekognition;  
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;  
import com.amazonaws.services.rekognition.model.Face;  
import com.amazonaws.services.rekognition.model.ListFacesRequest;  
import com.amazonaws.services.rekognition.model.ListFacesResult;  
import java.util.List;  
import com.fasterxml.jackson.databind.ObjectMapper;
```

```
public class ListFacesInCollection {
    public static final String collectionId = "MyCollection";

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

        AmazonRekognition rekognitionClient =
        AmazonRekognitionClientBuilder.defaultClient();

        ObjectMapper objectMapper = new ObjectMapper();

        ListFacesResult listFacesResult = null;
        System.out.println("Faces in collection " + collectionId);

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

            ListFacesRequest listFacesRequest = new ListFacesRequest()
                .withCollectionId(collectionId)
                .withMaxResults(1)
                .withNextToken(paginationToken);

            listFacesResult = rekognitionClient.listFaces(listFacesRequest);
            List < Face > faces = listFacesResult.getFaces();
            for (Face face: faces) {
                System.out.println(objectMapper.writerWithDefaultPrettyPrinter()
                    .writeValueAsString(face));
            }
        } while (listFacesResult != null && listFacesResult.getNextToken() !=
            null);
    }
}
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
// snippet-start:[rekognition.java2.list_faces_collection.import]
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;
// snippet-end:[rekognition.java2.list_faces_collection.import]

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

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "  <collectionId>\n\n" +
            "Where:\n" +
            "  collectionId - The name of the collection. \n\n";

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

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        System.out.println("Faces in collection " + collectionId);
        listFacesCollection(rekClient, collectionId);
    }
}
```

```
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.list_faces_collection.main]
    public static void listFacesCollection(RekognitionClient rekClient, String
collectionId ) {
        try {
            ListFacesRequest facesRequest = ListFacesRequest.builder()
                .collectionId(collectionId)
                .maxResults(10)
                .build();

            ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
            List<Face> faces = facesResponse.faces();
            for (Face face: faces) {
                System.out.println("Confidence level there is a face:
"+face.confidence());
                System.out.println("The face Id value is "+face.faceId());
            }

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

AWS CLI

AWS CLI Perintah ini menampilkan output JSON untuk operasi `list-faces` CLI. Ganti nilai `collection-id` dengan nama koleksi yang ingin Anda cantumkan. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
aws rekognition list-faces --collection-id "collection-id" --profile profile-
name
```

Python

Contoh ini menampilkan daftar wajah dalam koleksi.

Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import boto3

def list_faces_in_collection(collection_id):
    maxResults = 2
    faces_count = 0
    tokens = True

    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')
    response = client.list_faces(CollectionId=collection_id,
                                MaxResults=maxResults)

    print('Faces in collection ' + collection_id)

    while tokens:

        faces = response['Faces']

        for face in faces:
            print(face)
            faces_count += 1
        if 'NextToken' in response:
            nextToken = response['NextToken']
            response = client.list_faces(CollectionId=collection_id,
                                        NextToken=nextToken,
MaxResults=maxResults)
        else:
            tokens = False
    return faces_count

def main():
    collection_id = 'collection-id'
    faces_count = list_faces_in_collection(collection_id)
    print("faces count: " + str(faces_count))
```

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

.NET

Contoh ini menampilkan daftar wajah dalam koleksi.

Ubah nilai `collectionId` menjadi ID koleksi yang diinginkan.

```
//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 permintaan operasi

Input ke ListFaces adalah ID koleksi yang ingin Anda cantumi wajah. MaxResults adalah jumlah maksimum wajah yang dikirimkan. ListFaces juga mengambil daftar ID wajah untuk memfilter hasilnya, dan ID pengguna yang disediakan untuk mencantumkan hanya wajah yang terkait dengan pengguna yang diberikan.

```
{
  "CollectionId": "MyCollection",
  "MaxResults": 1
}
```

Jika respons tersebut memiliki wajah lebih dari yang diminta oleh MaxResults, token dikembalikan sehingga Anda dapat menggunakannya untuk mendapatkan serangkaian hasil berikutnya, dalam panggilan berikutnya untuk ListFaces. Sebagai contoh:

```
{
  "CollectionId": "MyCollection",
  "NextToken": "sm+5ythT3aeEVIR4WA....",
  "MaxResults": 1
}
```

ListFaces respon operasi

Respons dari ListFaces adalah informasi tentang metadata wajah yang disimpan dalam koleksi tertentu.

- FaceModelVersion— Versi model wajah yang terkait dengan koleksi. Untuk informasi selengkapnya, lihat [Versioning model](#).
- Wajah — Informasi tentang wajah dalam koleksi. Ini termasuk informasi tentang [BoundingBox](#), kepercayaan diri, pengidentifikasi gambar, dan ID wajah. Untuk informasi lebih lanjut, lihat [Wajah](#).
- NextToken— Token yang digunakan untuk mendapatkan set hasil berikutnya.

```
{
  "FaceModelVersion": "6.0",
  "Faces": [
    {
      "Confidence": 99.76940155029297,
      "IndexFacesModelVersion": "6.0",
      "UserId": "demoUser2",
      "ImageId": "56a0ca74-1c83-39dd-b363-051a64168a65",
      "BoundingBox": {
        "Width": 0.03177810087800026,
        "Top": 0.36568498611450195,
        "Left": 0.3453829884529114,
        "Height": 0.056759100407361984
      },
      "FaceId": "c92265d4-5f9c-43af-a58e-12be0ce02bc3"
    },
    {
      "BoundingBox": {
        "Width": 0.03254450112581253,
        "Top": 0.6080359816551208,
        "Left": 0.5160620212554932,
        "Height": 0.06347999721765518
      },
      "IndexFacesModelVersion": "6.0",
      "FaceId": "851cb847-dccc-4fea-9309-9f4805967855",
      "Confidence": 99.94369506835938,
      "ImageId": "a8aed589-ceec-35f7-9c04-82e0b546b024"
    },
    {
      "BoundingBox": {
        "Width": 0.03094629943370819,
        "Top": 0.4218429923057556,
        "Left": 0.6513839960098267,
        "Height": 0.05266290158033371
      },
      "IndexFacesModelVersion": "6.0",
      "FaceId": "c0eb3b65-24a0-41e1-b23a-1908b1aaeac1",
      "Confidence": 99.82969665527344,
      "ImageId": "56a0ca74-1c83-39dd-b363-051a64168a65"
    }
  ]
}
```

Menghapus wajah dari koleksi

Anda dapat menggunakan [DeleteFaces](#) operasi untuk menghapus wajah dari koleksi. Untuk informasi selengkapnya, lihat [Mengelola wajah dalam koleksi](#).

Menghapus wajah dari koleksi

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan AmazonRekognitionFullAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi DeleteFaces.

Java

Contoh ini menghapus satu wajah dari koleksi.

Ubah nilai `collectionId` menjadi koleksi yang berisi wajah yang ingin Anda hapus. Ubah nilai `faces` menjadi ID wajah yang ingin Anda hapus. Untuk menghapus beberapa wajah, tambahkan ID wajah pada array `faces`.

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

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.DeleteFacesRequest;
import com.amazonaws.services.rekognition.model.DeleteFacesResult;

import java.util.List;

public class DeleteFacesFromCollection {
    public static final String collectionId = "MyCollection";
    public static final String faces[] = {"xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx"};
```

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

    AmazonRekognition rekognitionClient =
AmazonRekognitionClientBuilder.defaultClient();

    DeleteFacesRequest deleteFacesRequest = new DeleteFacesRequest()
        .withCollectionId(collectionId)
        .withFaceIds(faces);

    DeleteFacesResult
deleteFacesResult=rekognitionClient.deleteFaces(deleteFacesRequest);

    List < String > faceRecords = deleteFacesResult.getDeletedFaces();
    System.out.println(Integer.toString(faceRecords.size()) + " face(s)
deleted:");
    for (String face: faceRecords) {
        System.out.println("FaceID: " + face);
    }
}
}
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
// snippet-end:[rekognition.java2.delete_faces_collection.import]

/**
```

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

        final String usage = "\n" +
            "Usage: " +
            "  <collectionId> <faceId> \n\n" +
            "Where:\n" +
            "  collectionId - The id of the collection from which faces are
deleted. \n\n" +
            "  faceId - The id of the face to delete. \n\n";

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

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.delete_faces_collection.main]
    public static void deleteFacesCollection(RekognitionClient rekClient,
        String collectionId,
        String faceId) {

        try {
```

```
        DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
            .collectionId(collectionId)
            .faceIds(faceId)
            .build();

        rekClient.deleteFaces(deleteFacesRequest);
        System.out.println("The face was deleted from the collection.");

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

AWS CLI

AWS CLI Perintah ini menampilkan output JSON untuk operasi `delete-faces` CLI. Ganti nilai `collection-id` dengan nama koleksi yang berisi wajah yang ingin Anda hapus. Ganti nilai `face-ids` dengan array ID wajah yang ingin Anda hapus. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
aws rekognition delete-faces --collection-id "collection-id" --face-ids "faceid"
--profile profile-name
```

Python

Contoh ini menghapus satu wajah dari koleksi.

Ubah nilai `collectionId` menjadi koleksi yang berisi wajah yang ingin Anda hapus. Ubah nilai `faces` menjadi ID wajah yang ingin Anda hapus. Untuk menghapus beberapa wajah, tambahkan ID wajah pada array `faces`. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import boto3
```



```
def delete_faces_from_collection(collection_id, faces):

    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')
    response = client.delete_faces(CollectionId=collection_id,
                                   FaceIds=faces)

    print(str(len(response['DeletedFaces'])) + ' faces deleted:')
    for faceId in response['DeletedFaces']:
        print(faceId)
    return len(response['DeletedFaces'])

def main():
    collection_id = 'collection-id'
    faces = []
    faces.append("xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx")

    faces_count = delete_faces_from_collection(collection_id, faces)
    print("deleted faces count: " + str(faces_count))

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

.NET

Contoh ini menghapus satu wajah dari koleksi.

Ubah nilai `collectionId` menjadi koleksi yang berisi wajah yang ingin Anda hapus. Ubah nilai `faces` menjadi ID wajah yang ingin Anda hapus. Untuk menghapus beberapa wajah, tambahkan ID wajah dalam daftar `faces`.

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

using System;
using System.Collections.Generic;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class DeleteFaces
{
```

```
public static void Example()
{
    String collectionId = "MyCollection";
    List<String> faces = new List<String>() { "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxxx" };

    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 permintaan operasi

Input ke DeleteFaces adalah ID koleksi yang berisi wajah, dan array ID wajah untuk wajah yang akan dihapus.

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

DeleteFaces respon operasi

Respon DeleteFaces berisi array ID wajah untuk wajah yang telah dihapus.

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

```
]
}
```

Jika ID wajah yang diberikan dalam input saat ini dikaitkan dengan Pengguna, mereka akan dikembalikan sebagai bagian dari `UnsuccessfulFaceDeletions` dengan alasan yang sah.

```
{
  "DeletedFaces": [
    "daf29cac-f910-41e9-851f-6eeb0e08f973"
  ],
  "UnsuccessfulFaceDeletions" : [
    {
      "FaceId" : "0b683aed-a0f1-48b2-9b5e-139e9cc2a757",
      "UserId" : "demoUser1",
      "Reason" : ["ASSOCIATED_TO_AN_EXISTING_USER"]
    }
  ]
}
```

Membuat pengguna

Anda dapat menggunakan [CreateUser](#) operasi untuk membuat pengguna baru dalam koleksi menggunakan ID pengguna unik yang Anda berikan. Anda kemudian dapat mengaitkan beberapa wajah dengan pengguna yang baru dibuat.

Untuk membuat pengguna (SDK)

1. Jika belum:
 - a. Buat atau perbarui akun pengguna IAM dengan `AmazonRekognitionFullAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi `CreateUser`.

Java

Contoh kode Java ini menciptakan pengguna.

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

```
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.CreateUserRequest;
import com.amazonaws.services.rekognition.model.CreateUserResult;

public class CreateUser {

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

        AmazonRekognition rekognitionClient =
        AmazonRekognitionClientBuilder.defaultClient();

        //Replace collectionId and userId with the name of the user that you
        want to create in that target collection.

        String collectionId = "MyCollection";
        String userId = "demoUser";
        System.out.println("Creating new user: " +
            userId);

        CreateUserRequest request = new CreateUserRequest()
            .withCollectionId(collectionId)
            .withUserId(userId);

        rekognitionClient.createUser(request);
    }
}
```

AWS CLI

AWS CLI Perintah ini menciptakan pengguna, menggunakan operasi create-user CLI.

```
aws rekognition create-user --user-id user-id --collection-id collection-name --
region region-name
--client-request-token request-token
```

Python

Contoh kode Python ini menciptakan pengguna.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3
from botocore.exceptions import ClientError
import logging

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def create_user(collection_id, user_id):
    """
    Creates a new User within a collection specified by CollectionId.
    Takes UserId as a parameter, which is a user provided ID which
    should be unique within the collection.

    :param collection_id: The ID of the collection where the indexed faces will
    be stored at.
    :param user_id: ID for the UserID to be created. This ID needs to be unique
    within the collection.

    :return: The indexFaces response
    """
    try:
        logger.info(f'Creating user: {collection_id}, {user_id}')
        client.create_user(
            CollectionId=collection_id,
            UserId=user_id
        )
    except ClientError:
        logger.exception(f'Failed to create user with given user id:
{user_id}')
        raise

def main():
    collection_id = "collection-id"
    user_id = "user-id"
    create_user(collection_id, user_id)

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

Menghapus pengguna

Anda dapat menggunakan [DeleteUser](#) operasi untuk menghapus pengguna dari koleksi, berdasarkan userID yang disediakan. Perhatikan bahwa setiap wajah yang terkait dengan userID dipisahkan dari userID sebelum userID yang ditentukan dihapus.

Untuk menghapus pengguna (SDK)

1. Jika belum:
 - a. Buat atau perbarui akun pengguna IAM dengan AmazonRekognitionFullAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi DeleteUser.

Java

Contoh kode Java ini menghapus pengguna.

```
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.DeleteUserRequest;
import com.amazonaws.services.rekognition.model.DeleteUserResult;

public class DeleteUser {

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

        AmazonRekognition rekognitionClient =
            AmazonRekognitionClientBuilder.defaultClient();

        //Replace collectionId and userId with the name of the user that you
        want to delete from that target collection.

        String collectionId = "MyCollection";
        String userId = "demoUser";
        System.out.println("Deleting existing user: " +
            userId);
    }
}
```

```

        DeleteUserRequest request = new DeleteUserRequest()
            .withCollectionId(collectionId)
            .withUserId(userId);

        rekognitionClient.deleteUser(request);
    }
}

```

AWS CLI

AWS CLI Perintah ini menghapus pengguna, menggunakan operasi create-user CLI.

```

aws rekognition delete-user --collection-id MyCollection
--user-id user-id --collection-id collection-name --region region-name

```

Python

Contoh kode Python ini menghapus pengguna.

```

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

import boto3
from botocore.exceptions import ClientError
import logging

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def delete_user(collection_id, user_id):
    """
    Delete the user from the given collection

    :param collection_id: The ID of the collection where user is stored.
    :param user_id: The ID of the user in the collection to delete.
    """
    logger.info(f'Deleting user: {collection_id}, {user_id}')
    try:
        client.delete_user(

```

```
        CollectionId=collection_id,
        UserId=user_id
    )
except ClientError:
    logger.exception(f'Failed to delete user with given user id:
{user_id}')
    raise

def main():
    collection_id = "collection-id"
    user_id = "user-id"
    delete_user(collection_id, user_id)

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

Mengaitkan wajah dengan pengguna

Anda dapat menggunakan [AssociateFaces](#) operasi untuk mengaitkan beberapa wajah individual dengan satu pengguna. Untuk mengaitkan wajah dengan pengguna, Anda harus terlebih dahulu membuat koleksi dan pengguna. Perhatikan bahwa vektor wajah harus berada dalam koleksi yang sama di mana vektor pengguna berada.

Untuk mengaitkan wajah (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan AmazonRekognitionFullAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi AssociateFaces.

Java

Contoh kode Java ini mengaitkan wajah dengan pengguna.

```
import java.util.Arrays;
import java.util.List;
```



```
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AssociateFacesRequest;
import com.amazonaws.services.rekognition.model.AssociateFacesResult;

public class AssociateFaces {

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

        AmazonRekognition rekognitionClient =
        AmazonRekognitionClientBuilder.defaultClient();

        /* Replace the below configurations to allow you successfully run the
        example

        @collectionId: The collection where user and faces are stored
        @userId: The user which faces will get associated to
        @faceIds: The list of face IDs that will get associated to the given
        user
        @userMatchThreshold: Minimum User match confidence required for the
        face to
        be associated with a User that has at least one
        faceID already associated
        */

        String collectionId = "MyCollection";
        String userId = "demoUser";
        String faceId1 = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx";
        String faceId2 = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx";
        List<String> faceIds = Arrays.asList(faceid1,faceid2);

        float userMatchThreshold = 0f;
        System.out.println("Associating faces to the existing user: " +
            userId);

        AssociateFacesRequest request = new AssociateFacesRequest()
            .withCollectionId(collectionId)
            .withUserId(userId)
            .withFaceIds(faceIds)
            .withUserMatchThreshold(userMatchThreshold);

        AssociateFacesResult result = rekognitionClient.associateFaces(request);
```

```
        System.out.println("Successful face associations: " +
result.getAssociatedFaces().size());
        System.out.println("Unsuccessful face associations: " +
result.getUnsuccessfulFaceAssociations().size());
    }
}
```

AWS CLI

AWS CLI Perintah ini mengaitkan wajah dengan pengguna, menggunakan operasi `associate-faces` CLI.

```
aws rekognition associate-faces --user-id user-id --face-ids face-id-1 face-id-2
--collection-id collection-name
--region region-name
```

Python

Contoh kode Python ini mengaitkan wajah dengan pengguna.

```
from botocore.exceptions import ClientError
import boto3
import logging

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def associate_faces(collection_id, user_id, face_ids):
    """
    Associate stored faces within collection to the given user

    :param collection_id: The ID of the collection where user and faces are
stored.
    :param user_id: The ID of the user that we want to associate faces to
    :param face_ids: The list of face IDs to be associated to the given user

    :return: response of AssociateFaces API
    """
    logger.info(f'Associating faces to user: {user_id}, {face_ids}')
```

```
try:
    response = client.associate_faces(
        CollectionId=collection_id,
        UserId=user_id,
        FaceIds=face_ids
    )
    print(f'- associated {len(response["AssociatedFaces"])} faces')
except ClientError:
    logger.exception("Failed to associate faces to the given user")
    raise
else:
    print(response)
    return response

def main():
    face_ids = ["faceId1", "faceId2"]
    collection_id = "collection-id"
    user_id = "user-id"
    associate_faces(collection_id, user_id, face_ids)

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

AssociateFaces respon operasi

Tanggapan untuk AssociateFaces mencakup `UserStatus`, yang merupakan status permintaan diasosiasikan, serta daftar yang akan `FaceIds` dikaitkan. Daftar `UnsuccessfulFaceAssociations` juga dikembalikan. Setelah mengajukan permintaan `AssociateFaces`, operasi mungkin memakan waktu sekitar satu menit untuk menyelesaikannya.

Untuk alasan ini, `UserStatus` dikembalikan, yang dapat memiliki nilai berikut:

- **DIBUAT** - Menunjukkan bahwa 'Pengguna' berhasil dibuat dan tidak ada wajah yang terkait dengannya saat ini. 'Pengguna' akan berada dalam keadaan ini sebelum panggilan 'AssociateFaces' berhasil dilakukan.
- **MEMPERBARUI** - Menunjukkan bahwa 'Pengguna' sedang diperbarui untuk mencerminkan wajah yang baru terkait/terputus dan akan menjadi **AKTIF** dalam beberapa detik. Hasil pencarian mungkin berisi 'Pengguna' dalam keadaan ini dan pelanggan dapat memilih untuk mengabaikannya dari hasil yang dikembalikan.

- **AKTIF** - Menunjukkan bahwa 'Pengguna' diperbarui untuk mencerminkan semua wajah yang terkait/tidak terkait dan berada dalam keadaan yang dapat dicari.

```
{
  "UnsuccessfulFaceAssociations": [
    {
      "Reasons": [
        "LOW_MATCH_CONFIDENCE"
      ],
      "FaceId": "f5817d37-94f6-0000-bfee-1a2b3c4d5e6f",
      "Confidence": 0.9375374913215637
    },
    {
      "Reasons": [
        "ASSOCIATED_TO_A_DIFFERENT_IDENTITY"
      ],
      "FaceId": "851cb847-dccc-1111-bfee-1a2b3c4d5e6f",
      "UserId": "demoUser2"
    }
  ],
  "UserStatus": "UPDATING",
  "AssociatedFaces": [
    {
      "FaceId": "35ebbb41-7f67-2222-bfee-1a2b3c4d5e6f"
    }
  ]
}
```

Memutus wajah dari pengguna

Anda dapat menggunakan [DisassociateFaces](#) operasi untuk menghapus asosiasi antara ID pengguna dan ID wajah.

Untuk memisahkan wajah (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan `AmazonRekognitionFullAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).

- b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi `DisassociateFaces`.

Java

Contoh Java ini menghapus hubungan antara `FaceID` dan `userId` dengan operasi `DisassociateFaces`

```
import java.util.Arrays;
import java.util.List;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.DisassociateFacesRequest;
import com.amazonaws.services.rekognition.model.DisassociateFacesResult;

public class DisassociateFaces {

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

        AmazonRekognition rekognitionClient =
            AmazonRekognitionClientBuilder.defaultClient();

        /* Replace the below configurations to allow you successfully run the
        example

        @collectionId: The collection where user and faces are stored
        @userId: The user which faces will get disassociated from
        @faceIds: The list of face IDs that will get disassociated from the
        given user
        */

        String collectionId = "MyCollection";
        String userId = "demoUser";
        String faceId1 = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";
        String faceId2 = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";
        List<String> faceIds = Arrays.asList(faceId1, faceId2);

        System.out.println("Disassociating faces from existing user: " +
```

```

        userId);

        DisassociateFacesRequest request = new DisassociateFacesRequest()
            .withCollectionId(collectionId)
            .withUserId(userId)
            .withFaceIds(faceIds)

        DisassociateFacesResult result =
rekognitionClient.disassociateFaces(request);

        System.out.println("Successful face disassociations: " +
result.getDisassociatedFaces().size());
        System.out.println("Unsuccessful face disassociations: " +
result.getUnsuccessfulFaceDisassociations().size());
    }
}

```

AWS CLI

AWS CLI Perintah ini menghapus hubungan antara FaceID dan userID dengan operasi.

DisassociateFaces

```
aws rekognition disassociate-faces --face-ids list-of-face-ids
--user-id user-id --collection-id collection-name --region region-name
```

Python

Contoh berikut menghapus hubungan antara FaceID dan userID dengan operasi.

DisassociateFaces

```

from botocore.exceptions import ClientError
import boto3
import logging

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def disassociate_faces(collection_id, user_id, face_ids):
    """
    Disassociate stored faces within collection to the given user

```

```

:param collection_id: The ID of the collection where user and faces are
stored.
:param user_id: The ID of the user that we want to disassociate faces from
:param face_ids: The list of face IDs to be disassociated from the given
user

:return: response of AssociateFaces API
"""
logger.info(f'Dissociating faces from user: {user_id}, {face_ids}')
try:
    response = client.disassociate_faces(
        CollectionId=collection_id,
        UserId=user_id,
        FaceIds=face_ids
    )
    print(f'- disassociated {len(response["DisassociatedFaces"])} faces')
except ClientError:
    logger.exception("Failed to disassociate faces from the given user")
    raise
else:
    print(response)
    return response

def main():
    face_ids = ["faceId1", "faceId2"]
    collection_id = "collection-id"
    user_id = "user-id"
    disassociate_faces(collection_id, user_id, face_ids)

if __name__ == "__main__":
    main()

```

DisassociateFaces respon operasi

Tanggapan untuk DisassociateFaces mencakup UserStatus, yang merupakan status permintaan disosiasi, serta daftar yang tidak FaceIds terkait. Daftar UnsuccessfulFaceDisassociations juga dikembalikan. Setelah mengajukan permintaan DisassociateFaces, operasi mungkin memakan waktu sekitar satu menit untuk menyelesaikannya. Untuk alasan ini, UserStatus dikembalikan, yang dapat memiliki nilai berikut:

- **DIBUAT** - Menunjukkan bahwa 'Pengguna' berhasil dibuat dan tidak ada wajah yang terkait dengannya saat ini. 'Pengguna' akan berada dalam keadaan ini sebelum panggilan 'AssociateFaces' berhasil dilakukan.
- **MEMPERBARUI** - Menunjukkan bahwa 'Pengguna' sedang diperbarui untuk mencerminkan wajah yang baru terkait/terputus dan akan menjadi AKTIF dalam beberapa detik. Hasil pencarian mungkin berisi 'Pengguna' dalam keadaan ini dan pelanggan dapat memilih untuk mengabaikannya dari hasil yang dikembalikan.
- **AKTIF** - Menunjukkan bahwa 'Pengguna' diperbarui untuk mencerminkan semua wajah yang terkait/tidak terkait dan berada dalam keadaan yang dapat dicari.

```
{
  "UserStatus": "UPDATING",
  "DisassociatedFaces": [
    {
      "FaceId": "c92265d4-5f9c-43af-a58e-12be0ce02bc3"
    }
  ],
  "UnsuccessfulFaceDisassociations": [
    {
      "Reasons": [
        "ASSOCIATED_TO_A_DIFFERENT_IDENTITY"
      ],
      "FaceId": "f5817d37-94f6-4335-bfee-6cf79a3d806e",
      "UserId": "demoUser1"
    }
  ]
}
```

Daftar pengguna dalam koleksi

Anda dapat menggunakan [ListUsers](#) operasi untuk daftar UserIds dan UserStatus. Untuk melihat FaceID yang terkait dengan userID, gunakan operasi. [ListFaces](#)

Untuk daftar pengguna (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan AmazonRekognitionFullAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).

- b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi ListUsers.

Java

Contoh Java ini mencantumkan pengguna dalam koleksi menggunakan ListUsers operasi.

```
import java.util.List;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.ListUsersRequest;
import com.amazonaws.services.rekognition.model.ListUsersResult;
import com.amazonaws.services.rekognition.model.User;

public class ListUsers {

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

        AmazonRekognition amazonRekognition =
            AmazonRekognitionClientBuilder.defaultClient();

        System.out.println("Listing users");
        int limit = 10;
        ListUsersResult listUsersResult = null;
        String paginationToken = null;
        do {
            if (listUsersResult != null) {
                paginationToken = listUsersResult.getNextToken();
            }
            ListUsersRequest request = new ListUsersRequest()
                .withCollectionId(collectionId)
                .withMaxResults(limit)
                .withNextToken(paginationToken);
            listUsersResult = amazonRekognition.listUsers(request);

            List<User> users = listUsersResult.getUsers();
            for (User currentUser: users) {
                System.out.println(currentUser.getUserId() + " : " +
                    currentUser.getUserStatus());
            }
        }
    }
}
```

```
    } while (listUsersResult.getNextToken() != null);
  }
}
```

AWS CLI

AWS CLI Perintah ini mencantumkan pengguna dalam koleksi dengan ListUsers operasi.

```
aws rekognition list-users --collection-id collection-id --max-results number-of-max-results
```

Python

Contoh berikut mencantumkan pengguna dalam koleksi dengan ListUsers operasi.

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

import boto3
from botocore.exceptions import ClientError
import logging
from pprint import pprint

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def list_users(collection_id):
    """
    List all users from the given collection

    :param collection_id: The ID of the collection where user is stored.

    :return: response that contains list of Users found within given collection
    """
    logger.info(f'Listing the users in collection: {collection_id}')
    try:
        response = client.list_users(
            CollectionId=collection_id
        )
        pprint(response["Users"])
```

```

except ClientError:
    logger.exception(f'Failed to list all user from given collection:
{collection_id}')
    raise
else:
    return response

def main():
    collection_id = "collection-id"
    list_users(collection_id)

if __name__ == "__main__":
    main()

```

ListUsers respon operasi

Tanggapan atas permintaan untuk ListUsers menyertakan daftar Users dalam koleksi bersama dengan UsedId dan UserStatus Pengguna.

```

{
  "NextToken": "B1asJT3bAb/ttuGgPFV8BZoBZyGQz1UHXbuTNLh48a6enU7kXKw43hp0wizW7L0k/
Gk7Em091znoq6+FcDCcSq2o1rn7A98BLkt5keu+ZRVrUTyrXtT6J7Hmp
+ieQ2an6Zu0qzPfcDPeaJ9eAxG2d0WNrzJgi5hvmjoiSTTfKX3MQz1sduWQkvAAs4hZfhZoKFahFlqWofshCXa/
FHAAY3PL1PjxXbkNeSSMq8V7i1M1KCdrPVykCv9MokpPt7jtNvKPEZGUhxgBTFMxNWLEcFnzAiCWDg91dFy/
La1shPjXA9UVc5Gx9vIJNQ/
e03cQRghAkCT3FOAiXsLAnA0150DTomZpWWVpqB21wKpI3LYmfAVFrDPGzpbTV1RmLsJm41bkmnBBBw9+DHZ1Jn7zW
+qc5Fs3yaHu0f51Xg==",
  "Users": [
    {
      "UserId": "demoUser4",
      "UserStatus": "CREATED"
    },
    {
      "UserId": "demoUser2",
      "UserStatus": "CREATED"
    }
  ]
}

```

Mencari wajah dengan ID wajah

Anda dapat menggunakan [SearchFaces](#) operasi untuk mencari pengguna dalam koleksi yang cocok dengan wajah terbesar dalam gambar yang disediakan.

ID wajah dikembalikan dalam respons [IndexFaces](#) operasi ketika wajah terdeteksi dan ditambahkan ke koleksi. Untuk informasi selengkapnya, lihat [Mengelola wajah dalam koleksi](#).

Untuk mencari wajah dalam koleksi menggunakan ID wajahnya (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan AmazonRekognitionFullAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi SearchFaces.

Java

Contoh ini menampilkan informasi tentang wajah yang cocok dengan wajah yang diidentifikasi berdasarkan ID-nya.

Ubah nilai `collectionID` menjadi koleksi yang berisi wajah yang diperlukan. Ubah nilai `faceId` menjadi pengenalan wajah yang ingin Anda temukan.

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

Jalankan kode contoh. Informasi tentang pencocokan wajah ditampilkan.

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
// snippet-start:[rekognition.java2.match_faces_collection.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;
// snippet-end:[rekognition.java2.match_faces_collection.import]

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

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "  <collectionId> <sourceImage>\n\n" +
            "Where:\n" +
            "  collectionId - The id of the collection. \n" +
            "  sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png). \n\n";

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

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
```

```
        .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
        .build();

    System.out.println("Searching for a face in a collections");
    searchFaceById(rekClient, collectionId, faceId );
    rekClient.close();
}

// snippet-start:[rekognition.java2.match_faces_collection.main]
public static void searchFaceById(RekognitionClient rekClient,String
collectionId, String faceId) {

    try {
        SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
            .collectionId(collectionId)
            .faceId(faceId)
            .faceMatchThreshold(70F)
            .maxFaces(2)
            .build();

        SearchFacesResponse imageResponse =
rekClient.searchFaces(searchFacesRequest) ;
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face: faceImageMatches) {
            System.out.println("The similarity level is
"+face.similarity());
            System.out.println();
        }

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

AWS CLI

AWS CLI Perintah ini menampilkan output JSON untuk operasi search-faces CLI. Ganti nilai face-id dengan pengenal wajah yang ingin Anda cari, dan ganti nilai collection-id

dengan koleksi yang ingin Anda cari. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
aws rekognition search-faces --face-id face-id --collection-id "collection-id"
--profile profile-name
```

Python

Contoh ini menampilkan informasi tentang wajah yang cocok dengan wajah yang diidentifikasi berdasarkan ID-nya.

Ubah nilai `collectionID` menjadi koleksi yang berisi wajah yang diperlukan. Ubah nilai `faceId` menjadi pengenalan wajah yang ingin Anda temukan. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import boto3

def search_face_in_collection(face_id, collection_id):
    threshold = 90
    max_faces = 2

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

    response = client.search_faces(CollectionId=collection_id,
                                   FaceId=face_id,
                                   FaceMatchThreshold=threshold,
                                   MaxFaces=max_faces)

    face_matches = response['FaceMatches']
    print('Matching faces')
    for match in face_matches:
        print('FaceId: ' + match['Face']['FaceId'])
        print('Similarity: ' + "{:.2f}".format(match['Similarity']) + "%")

    return len(face_matches)

def main():
```



```
face_id = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx'
collection_id = 'collection-id'

faces = []
faces.append(face_id)

faces_count = search_face_in_collection(face_id, collection_id)
print("faces found: " + str(faces_count))

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

.NET

Contoh ini menampilkan informasi tentang wajah yang cocok dengan wajah yang diidentifikasi berdasarkan ID-nya.

Ubah nilai `collectionID` menjadi koleksi yang berisi wajah yang diperlukan. Ubah nilai `faceId` menjadi pengenalan wajah yang ingin Anda temukan.

```
//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-xxxxxxxxxxxxx";

        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);
    }
}
```

Jalankan kode contoh. Informasi tentang pencocokan wajah ditampilkan.

SearchFaces permintaan operasi

Mengingat ID wajah (setiap wajah yang disimpan dalam koleksi wajah mempunyai ID wajah), SearchFaces mencari koleksi wajah untuk wajah serupa yang ditentukan. Respons tersebut tidak mencakup wajah yang Anda cari. Respons tersebut hanya mencakup wajah serupa. Secara default, SearchFaces mengembalikan wajah yang algoritmenya mendeteksi kemiripan lebih besar dari 80%. Kemiripan menunjukkan seberapa cocok wajah tersebut dengan wajah input. Secara opsional, Anda dapat menggunakan FaceMatchThreshold untuk menentukan nilai yang berbeda.

```
{
  "CollectionId": "MyCollection",
  "FaceId": "0b683aed-a0f1-48b2-9b5e-139e9cc2a757",
  "MaxFaces": 2,
  "FaceMatchThreshold": 99
}
```

SearchFaces respon operasi

Operasi tersebut mengembalikan array kecocokan wajah yang ditemukan dan ID wajah yang Anda berikan sebagai input.

```
{
  "SearchedFaceId": "7ecf8c19-5274-5917-9c91-1db9ae0449e2",
  "FaceMatches": [ list of face matches found ]
}
```

Untuk setiap kecocokan wajah yang ditemukan, respons meliputi kemiripan dan metadata wajah, seperti yang ditunjukkan dalam respons contoh berikut:

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

Mencari wajah dengan gambar

Anda dapat menggunakan [SearchFacesByImage](#) operasi untuk mencari wajah dalam koleksi yang cocok dengan wajah terbesar dalam gambar yang disediakan.

Untuk informasi selengkapnya, lihat [Mencari wajah dan pengguna dalam koleksi](#).

Untuk mencari wajah dalam koleksi menggunakan citra (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan AmazonRekognitionFullAccess dan AmazonS3ReadOnlyAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Unggah citra (yang berisi satu atau beberapa wajah) ke bucket S3 Anda.

Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

3. Gunakan contoh berikut untuk memanggil operasi SearchFacesByImage.

Java

Contoh ini menampilkan informasi tentang wajah yang cocok dengan wajah terbesar dalam citra. Contoh kode menentukan kedua parameter FaceMatchThreshold dan MaxFaces untuk membatasi hasil yang ditampilkan dalam respons.

Dalam contoh berikut, lakukan perubahan berikut: ubah nilai collectionId dengan koleksi yang ingin Anda cari, ubah nilai bucket dengan bucket yang berisi citra input, dan ubah nilai photo dengan citra input.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/  
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
package aws.example.rekognition.image;  
import com.amazonaws.services.rekognition.AmazonRekognition;  
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
```

```
import com.amazonaws.services.rekognition.model.FaceMatch;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.SearchFacesByImageRequest;
import com.amazonaws.services.rekognition.model.SearchFacesByImageResult;
import java.util.List;
import com.fasterxml.jackson.databind.ObjectMapper;

public class SearchFaceMatchingImageCollection {
    public static final String collectionId = "MyCollection";
    public static final String bucket = "bucket";
    public static final String photo = "input.jpg";

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

        AmazonRekognition rekognitionClient =
AmazonRekognitionClientBuilder.defaultClient();

        ObjectMapper objectMapper = new ObjectMapper();

        // Get an image object from S3 bucket.
        Image image=new Image()
            .withS3Object(new S3Object()
                .withBucket(bucket)
                .withName(photo));

        // Search collection for faces similar to the largest face in the image.
        SearchFacesByImageRequest searchFacesByImageRequest = new
SearchFacesByImageRequest()
            .withCollectionId(collectionId)
            .withImage(image)
            .withFaceMatchThreshold(70F)
            .withMaxFaces(2);

        SearchFacesByImageResult searchFacesByImageResult =
            rekognitionClient.searchFacesByImage(searchFacesByImageRequest);

        System.out.println("Faces matching largest face in image from" + photo);
        List < FaceMatch > faceImageMatches =
searchFacesByImageResult.getFaceMatches();
        for (FaceMatch face: faceImageMatches) {
            System.out.println(objectMapper.writerWithDefaultPrettyPrinter()
                .writeValueAsString(face));
        }
    }
}
```

```
        System.out.println();
    }
}
}
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
// snippet-start:[rekognition.java2.search_faces_collection.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import
    software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
// snippet-end:[rekognition.java2.search_faces_collection.import]

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

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

    final String usage = "\n" +
        "Usage: " +
        "  <collectionId> <sourceImage>\n\n" +
        "Where:\n" +
        "  collectionId - The id of the collection. \n" +
        "  sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png). \n\n";

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

    String collectionId = args[0];
    String sourceImage = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
        .build();

    System.out.println("Searching for a face in a collections");
    searchFaceInCollection(rekClient, collectionId, sourceImage );
    rekClient.close();
}

// snippet-start:[rekognition.java2.search_faces_collection.main]
public static void searchFaceInCollection(RekognitionClient rekClient,String
collectionId, String sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(new
File(sourceImage));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
            .image(souImage)
```

```

        .maxFaces(10)
        .faceMatchThreshold(70F)
        .collectionId(collectionId)
        .build();

        SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest) ;
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face: faceImageMatches) {
            System.out.println("The similarity level is
"+face.similarity());
            System.out.println();
        }

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

```

AWS CLI

AWS CLI Perintah ini menampilkan output JSON untuk operasi `search-faces-by-image` CLI. Ganti nilai `Bucket` dengan bucket S3 yang Anda gunakan pada langkah 2. Ganti nilai `Name` dengan nama file citra yang Anda gunakan pada langkah 2. Ganti nilai `collection-id` dengan koleksi yang ingin Anda lakukan pencarian. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```

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

```

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda alih-alih tanda kutip tunggal dan hindari tanda kutip ganda bagian dalam dengan garis miring terbalik (yaitu `\`) untuk mengatasi kesalahan parser yang mungkin Anda temui. Sebagai contoh, lihat berikut ini:


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

Python

Contoh ini menampilkan informasi tentang wajah yang cocok dengan wajah terbesar dalam citra. Contoh kode menentukan kedua parameter `FaceMatchThreshold` dan `MaxFaces` untuk membatasi hasil yang ditampilkan dalam respons.

Dalam contoh berikut, lakukan perubahan berikut: ubah nilai `collectionId` dengan koleksi yang ingin Anda cari, dan ganti nilai `bucket` dan `photo` dengan nama bucket Amazon S3 dan citra yang Anda gunakan pada Langkah 2. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
#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

Contoh ini menampilkan informasi tentang wajah yang cocok dengan wajah terbesar dalam citra. Contoh kode menentukan kedua parameter `FaceMatchThreshold` dan `MaxFaces` untuk membatasi hasil yang ditampilkan dalam respons.

Dalam contoh berikut, lakukan perubahan berikut: ubah nilai `collectionId` dengan koleksi yang ingin Anda cari, dan ganti nilai-nilai `bucket` dan `photo` dengan nama bucket Amazon S3 dan citra yang Anda gunakan pada langkah 2.

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

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

public class SearchFacesMatchingImage
{
    public static void Example()
    {
        String collectionId = "MyCollection";
        String bucket = "bucket";
        String photo = "input.jpg";

        AmazonRekognitionClient rekognitionClient = new
AmazonRekognitionClient();

        // Get an image object from S3 bucket.
        Image image = new Image()
        {
            S3Object = new S3Object()
            {
                Bucket = bucket,
```

```

        Name = photo
    }
};

SearchFacesByImageRequest searchFacesByImageRequest = new
SearchFacesByImageRequest()
{
    CollectionId = collectionId,
    Image = image,
    FaceMatchThreshold = 70F,
    MaxFaces = 2
};

SearchFacesByImageResponse searchFacesByImageResponse =
rekognitionClient.SearchFacesByImage(searchFacesByImageRequest);

Console.WriteLine("Faces matching largest face in image from " + photo);
foreach (FaceMatch face in searchFacesByImageResponse.FaceMatches)
    Console.WriteLine("FaceId: " + face.Face.FaceId + ", Similarity: " +
face.Similarity);
}
}

```

SearchFacesByImage permintaan operasi

Parameter input pada SearchFacesImageByImage adalah koleksi untuk mencari dan lokasi citra sumber. Dalam contoh ini, citra sumber disimpan dalam sebuah bucket Amazon S3 (S3Object). Yang juga ditentukan adalah wajah maksimal yang dikembalikan (MaxFaces) dan kepercayaan minimal yang harus dicocokkan agar wajah dikirimkan (FaceMatchThreshold).

```

{
  "CollectionId": "MyCollection",
  "Image": {
    "S3Object": {
      "Bucket": "bucket",
      "Name": "input.jpg"
    }
  },
  "MaxFaces": 2,
  "FaceMatchThreshold": 99
}

```

SearchFacesByImage respon operasi

Mengingat citra input (.jpeg atau .png), operasi mendeteksi wajah dalam citra input terlebih dahulu, kemudian mencari koleksi wajah yang ditentukan untuk wajah yang serupa.

Note

Jika layanan mendeteksi beberapa wajah pada citra input, layanan ini menggunakan wajah terbesar yang terdeteksi untuk mencari koleksi wajah.

Operasi tersebut mengirimkan array kecocokan wajah yang ditemukan dan informasi tentang wajah input. Termasuk informasi seperti kotak pembatas, bersama dengan nilai kepercayaan, yang menunjukkan tingkat kepercayaan bahwa kotak pembatas berisi wajah.

Secara default, SearchFacesByImage mengirimkan wajah yang algoritmenya mendeteksi kemiripan lebih dari 80%. Kemiripan menunjukkan seberapa cocok wajah tersebut dengan wajah input. Anda juga dapat menggunakan FaceMatchThreshold untuk menentukan nilai yang berbeda. Untuk setiap kecocokan wajah yang ditemukan, respons tersebut meliputi kemiripan dan metadata wajah, seperti yang ditunjukkan dalam respons contoh berikut:

```
{
  "FaceMatches": [
    {
      "Face": {
        "BoundingBox": {
          "Height": 0.06333330273628235,
          "Left": 0.1718519926071167,
          "Top": 0.7366669774055481,
          "Width": 0.11061699688434601
        },
        "Confidence": 100,
        "ExternalImageId": "input.jpg",
        "FaceId": "578e2e1b-d0b0-493c-aa39-ba476a421a34",
        "ImageId": "9ba38e68-35b6-5509-9d2e-fcffa75d1653"
      },
      "Similarity": 99.9764175415039
    }
  ],
  "FaceModelVersion": "3.0",
  "SearchedFaceBoundingBox": {
```

```
    "Height": 0.06333333253860474,  
    "Left": 0.17185185849666595,  
    "Top": 0.7366666793823242,  
    "Width": 0.11061728745698929  
  },  
  "SearchedFaceConfidence": 99.99999237060547  
}
```

Mencari pengguna (ID wajah/ID pengguna)

Anda dapat menggunakan [SearchUsers](#) operasi untuk mencari pengguna dalam koleksi tertentu yang cocok dengan ID wajah atau ID pengguna yang disediakan. Operasi mencantumkan `UserIds` peringkat yang dikembalikan dengan skor kesamaan tertinggi di atas yang diminta `UserMatchThreshold`. ID pengguna dibuat dalam `CreateUsers` operasi. Untuk informasi selengkapnya, lihat [Mengelola pengguna dalam koleksi](#).

Untuk mencari pengguna (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan `AmazonRekognitionFullAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi `SearchUsers`.

Java

Contoh Java ini mencari pengguna dalam koleksi menggunakan operasi `SearchUsers`

```
import com.amazonaws.services.rekognition.AmazonRekognition;  
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;  
import com.amazonaws.services.rekognition.model.UserMatch;  
import com.amazonaws.services.rekognition.model.SearchUsersRequest;  
import com.amazonaws.services.rekognition.model.SearchUsersResult;  
import com.amazonaws.services.rekognition.model.UserMatch;  
  
public class SearchUsers {  
    //Replace collectionId and faceId with the values you want to use.
```

```
public static final String collectionId = "MyCollection";
public static final String faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";

public static final String userd = 'demo-user';

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

    AmazonRekognition rekognitionClient =
AmazonRekognitionClientBuilder.defaultClient();

    // Search collection for faces matching the user id.
    SearchUsersRequest request = new SearchUsersRequest()
        .withCollectionId(collectionId)
        .withUserId(userId);

    SearchUsersResult result =
        rekognitionClient.searchUsers(request);

    System.out.println("Printing first search result with matched user and
similarity score");
    for (UserMatch match: result.getUserMatches()) {
        System.out.println(match.getUser().getUserId() + " with similarity
score " + match.getSimilarity());
    }

    // Search collection for faces matching the face id.
    SearchUsersRequest request1 = new SearchUsersRequest()
        .withCollectionId(collectionId)
        .withFaceId(faceId);

    SearchUsersResult result1 =
        rekognitionClient.searchUsers(request1);

    System.out.println("Printing second search result with matched user and
similarity score");
    for (UserMatch match: result1.getUserMatches()) {
        System.out.println(match.getUser().getUserId() + " with similarity
score " + match.getSimilarity());
    }
}
```

AWS CLI

AWS CLI Perintah ini mencari pengguna dalam koleksi dengan SearchUsers operasi.

```
aws rekognition search-users --face-id face-id --collection-id collection-id --  
region region-name
```

Python

Contoh berikut mencari pengguna dalam koleksi dengan SearchUsers operasi.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/  
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
import boto3  
from botocore.exceptions import ClientError  
import logging  
  
logger = logging.getLogger(__name__)  
session = boto3.Session(profile_name='profile-name')  
client = session.client('rekognition')  
  
def search_users_by_face_id(collection_id, face_id):  
    """  
    SearchUsers operation with face ID provided as the search source  
  
    :param collection_id: The ID of the collection where user and faces are  
    stored.  
    :param face_id: The ID of the face in the collection to search for.  
  
    :return: response of SearchUsers API  
    """  
    logger.info(f'Searching for users using a face-id: {face_id}')  
    try:  
        response = client.search_users(  
            CollectionId=collection_id,  
            FaceId=face_id  
        )  
        print(f'- found {len(response["UserMatches"])} matches')  
        print([f'- {x["User"]["UserId"]} - {x["Similarity"]}' for x in  
response["UserMatches"]])
```

```
except ClientError:
    logger.exception(f'Failed to perform SearchUsers with given face id:
{face_id}')
    raise
else:
    print(response)
    return response

def search_users_by_user_id(collection_id, user_id):
    """
    SearchUsers operation with user ID provided as the search source

    :param collection_id: The ID of the collection where user and faces are
stored.
    :param user_id: The ID of the user in the collection to search for.

    :return: response of SearchUsers API
    """
    logger.info(f'Searching for users using a user-id: {user_id}')
    try:
        response = client.search_users(
            CollectionId=collection_id,
            UserId=user_id
        )
        print(f'- found {len(response["UserMatches"])} matches')
        print([f'- {x["User"]["UserId"]} - {x["Similarity"]}%' for x in
response["UserMatches"]])
    except ClientError:
        logger.exception(f'Failed to perform SearchUsers with given face id:
{user_id}')
        raise
    else:
        print(response)
        return response

def main():
    collection_id = "collection-id"
    user_id = "user-id"
    face_id = "face-id"
    search_users_by_face_id(collection_id, face_id)
    search_users_by_user_id(collection_id, user_id)

if __name__ == "__main__":
```



```
main()
```

SearchUsers permintaan operasi

Diberikan FaceID atau userId, SearchUsers mencari collectionId yang ditentukan untuk kecocokan pengguna. Secara default, SearchUsers mengembalikan userIds yang skor kesamaannya lebih besar dari 80%. Kesamaan menunjukkan seberapa dekat userId cocok dengan FaceID atau userId yang disediakan. Jika beberapa UserIds dikembalikan, mereka terdaftar dalam urutan skor kesamaan tertinggi ke terendah. Secara opsional, Anda dapat menggunakan UserMatchThreshold untuk menentukan nilai yang berbeda. Untuk informasi selengkapnya, lihat [Mengelola pengguna dalam koleksi](#).

Berikut ini adalah contoh SearchUsers permintaan yang menggunakan UserId:

```
{
  "CollectionId": "MyCollection",
  "UserId": "demoUser1",
  "MaxUsers": 2,
  "UserMatchThreshold": 99
}
```

Berikut ini adalah contoh SearchUsers permintaan yang menggunakan FaceId:

```
{
  "CollectionId": "MyCollection",
  "FaceId": "bff43c40-cfa7-4b94-bed8-8a08b2205107",
  "MaxUsers": 2,
  "UserMatchThreshold": 99
}
```

SearchUsers respon operasi

Jika mencari dengan FaceId, tanggapan untuk SearchUsers mencakup FaceId untuk SearchedFace, serta daftar UserMatches dan UserId dan UserStatus untuk setiap Pengguna.

```
{
  "SearchedFace": {
    "FaceId": "bff43c40-cfa7-4b94-bed8-8a08b2205107"
  },
  "UserMatches": [
    {
      "User": {
        "UserId": "demoUser1",
        "UserStatus": "ACTIVE"
      },
      "Similarity": 100.0
    },
    {
      "User": {
        "UserId": "demoUser2",
        "UserStatus": "ACTIVE"
      },
      "Similarity": 99.97946166992188
    }
  ],
  "FaceModelVersion": "6"
}
```

Jika mencari dengan `userId`, respons untuk `SearchUsers` menyertakan `UserId` untuk `SearchedUser`, selain elemen respons lainnya.

```
{
  "SearchedUser": {
    "UserId": "demoUser1"
  },
  "UserMatches": [
    {
      "User": {
        "UserId": "demoUser2",
        "UserStatus": "ACTIVE"
      },
      "Similarity": 99.97946166992188
    }
  ],
}
```

```
"FaceModelVersion": "6"  
}
```

Mencari pengguna (gambar)

`SearchUsersByImage` mencari `collectionId` yang ditentukan untuk pengguna dalam koleksi yang cocok dengan wajah terbesar yang terdeteksi dalam gambar yang disediakan. Secara default, `SearchUsersByImage` mengembalikan `userIds` yang skor kesamaannya lebih besar dari 80%. Kesamaan menunjukkan seberapa dekat `userId` cocok dengan wajah terbesar yang terdeteksi pada gambar yang disediakan. Jika beberapa `UserIds` dikembalikan, mereka terdaftar dalam urutan skor kesamaan tertinggi ke terendah. Secara opsional, Anda dapat menggunakan `UserMatchThreshold` untuk menentukan nilai yang berbeda. Untuk informasi selengkapnya, lihat [Mengelola pengguna dalam koleksi](#).

Untuk mencari pengguna berdasarkan gambar (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan `AmazonRekognitionFullAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi `SearchUsersByImage`.

Java

Contoh Java ini mencari pengguna dalam koleksi berdasarkan gambar input, menggunakan operasi `SearchUsersByImage`

```
import com.amazonaws.services.rekognition.AmazonRekognition;  
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;  
import com.amazonaws.services.rekognition.model.Image;  
import com.amazonaws.services.rekognition.model.S3Object;  
import com.amazonaws.services.rekognition.model.SearchUsersByImageRequest;  
import com.amazonaws.services.rekognition.model.SearchUsersByImageResult;  
import com.amazonaws.services.rekognition.model.UserMatch;
```

```
public class SearchUsersByImage {
    //Replace bucket, collectionId and photo with your values.
    public static final String collectionId = "MyCollection";
    public static final String s3Bucket = "bucket";
    public static final String s3PhotoFileKey = "input.jpg";

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

        AmazonRekognition rekognitionClient =
        AmazonRekognitionClientBuilder.defaultClient();

        // Get an image object from S3 bucket.
        Image image = new Image()
            .withS3Object(new S3Object()
                .withBucket(s3Bucket)
                .withName(s3PhotoFileKey));

        // Search collection for users similar to the largest face in the image.
        SearchUsersByImageRequest request = new SearchUsersByImageRequest()
            .withCollectionId(collectionId)
            .withImage(image)
            .withUserMatchThreshold(70F)
            .withMaxUsers(2);

        SearchUsersByImageResult result =
            rekognitionClient.searchUsersByImage(request);

        System.out.println("Printing search result with matched user and
        similarity score");
        for (UserMatch match: result.getUserMatches()) {
            System.out.println(match.getUser().getUserId() + " with similarity
            score " + match.getSimilarity());
        }
    }
}
```

AWS CLI

AWS CLI Perintah ini mencari pengguna dalam koleksi berdasarkan gambar input, dengan SearchUsersByImage operasi.

```
aws rekognition search-users-by-image --image '{"S3Object":
{"Bucket": "s3BucketName", "Name": "file-name"}}' --collection-id MyCollectionId --
region region-name
```

Python

Contoh berikut mencari pengguna dalam koleksi berdasarkan gambar input, dengan SearchUsersByImage operasi.

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

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

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def load_image(file_name):
    """
    helper function to load the image for indexFaces call from local disk

    :param image_file_name: The image file location that will be used by
indexFaces call.
    :return: The Image in bytes
    """
    print(f'- loading image: {file_name}')
    with open(file_name, 'rb') as file:
        return {'Bytes': file.read()}

def search_users_by_image(collection_id, image_file):
    """
    SearchUsersByImage operation with user ID provided as the search source

    :param collection_id: The ID of the collection where user and faces are
stored.
    :param image_file: The image that contains the reference face to search
for.
```

```
:return: response of SearchUsersByImage API
"""
logger.info(f'Searching for users using an image: {image_file}')
try:
    response = client.search_users_by_image(
        CollectionId=collection_id,
        Image=load_image(image_file)
    )
    print(f'- found {len(response["UserMatches"])} matches')
    print([f'- {x["User"]["UserId"]} - {x["Similarity"]}% ' for x in
response["UserMatches"]])
except ClientError:
    logger.exception(f'Failed to perform SearchUsersByImage with given
image: {image_file}')
    raise
else:
    print(response)
    return response

def main():
    collection_id = "collection-id"
    IMAGE_SEARCH_SOURCE = os.getcwd() + '/image_path'
    search_users_by_image(collection_id, IMAGE_SEARCH_SOURCE)

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

SearchUsersByImage permintaan operasi

Permintaan untuk SearchUsersByImage menyertakan koleksi untuk mencari dan lokasi gambar sumber. Dalam contoh ini, gambar sumber disimpan dalam bucket Amazon S3 (S3Object). Juga ditentukan adalah jumlah maksimum pengguna untuk kembali (MaxUsers) dan keyakinan minimum yang harus dicocokkan agar pengguna dikembalikan (UserMatchThreshold).

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

```
    }
  },
  "MaxUsers": 2,
  "UserMatchThreshold": 99
}
```

SearchUsersByImage respon operasi

Respon untuk SearchUsersByImage mencakup FaceDetail objek untuk SearchedFace, serta daftar UserMatches dengan UserId, Similarity, dan UserStatus untuk masing-masing. Jika gambar input berisi lebih dari satu wajah, daftar UnsearchedFaces akan juga dikembalikan.

```
{
  "SearchedFace": {
    "FaceDetail": {
      "BoundingBox": {
        "Width": 0.23692893981933594,
        "Top": 0.19235000014305115,
        "Left": 0.39177176356315613,
        "Height": 0.5437348484992981
      }
    }
  },
  "UserMatches": [
    {
      "User": {
        "UserId": "demoUser1",
        "UserStatus": "ACTIVE"
      },
      "Similarity": 100.0
    },
    {
      "User": {
        "UserId": "demoUser2",
        "UserStatus": "ACTIVE"
      },
      "Similarity": 99.97946166992188
    }
  ],
  "FaceModelVersion": "6",
  "UnsearchedFaces": [
```

```
{
  "FaceDetails": {
    "BoundingBox": {
      "Width": 0.031677018851041794,
      "Top": 0.5593535900115967,
      "Left": 0.6102562546730042,
      "Height": 0.0682177022099495
    }
  },
  "Reasons": [
    "FACE_NOT_LARGEST"
  ]
},
{
  "FaceDetails": {
    "BoundingBox": {
      "Width": 0.03254449740052223,
      "Top": 0.6080358028411865,
      "Left": 0.516062319278717,
      "Height": 0.06347997486591339
    }
  },
  "Reasons": [
    "FACE_NOT_LARGEST"
  ]
}
]
```

Mencari video yang disimpan untuk wajah

Anda dapat mencari koleksi wajah yang cocok dengan wajah orang yang terdeteksi dalam video yang tersimpan atau video streaming. Bagian ini mencakup pencarian wajah dalam video yang tersimpan. Untuk informasi tentang pencarian wajah dalam video streaming, lihat [Bekerja dengan acara video streaming](#).

Wajah yang Anda cari harus terlebih dahulu diindeks ke dalam koleksi dengan menggunakan [IndexFaces](#). Untuk informasi selengkapnya, lihat [Menambahkan wajah ke koleksi](#).

Pencarian wajah Amazon Rekognition Video mengikuti alur kerja tidak sinkron yang sama seperti operasi Amazon Rekognition Video lainnya yang menganalisis video yang disimpan dalam bucket

Amazon S3. Untuk mulai mencari wajah dalam video yang disimpan, panggil [StartFaceSearch](#) dan berikan ID koleksi yang ingin Anda cari. Amazon Rekognition Video menerbitkan status penyelesaian analisis video ke topik Amazon Simple Notification Service (Amazon SNS). Jika analisis video berhasil, hubungi [GetFaceSearch](#) untuk mendapatkan hasil pencarian. Untuk informasi selengkapnya tentang memulai analisis video dan mendapatkan hasilnya, lihat [Memanggil operasi Amazon Rekognition Video](#).

Prosedur berikut menunjukkan cara mencari koleksi wajah yang sesuai dengan wajah orang yang terdeteksi dalam video. Prosedur ini juga menunjukkan cara mendapatkan data pelacakan untuk orang yang dicocokkan dalam video. Prosedur tersebut melebar ke kode di [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#), yang menggunakan antrean Amazon Simple Queue Service (Amazon SQS) untuk mendapatkan status penyelesaian permintaan analisis video.

Untuk mencari video untuk mencocokkan wajah (SDK)

1. [Buat koleksi](#).
2. [Indeks wajah ke dalam koleksi](#).
3. Lakukan [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#).
4. Tambahkan kode berikut ke kelas VideoDetect yang Anda buat di langkah 3.

Java

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

//Face collection search in video
=====
private static void StartFaceSearchCollection(String bucket, String
video, String collection) throws Exception{

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

    StartFaceSearchRequest req = new StartFaceSearchRequest()
        .withCollectionId(collection)
        .withVideo(new Video()
```

```
        .withS3Object(new S3Object()
            .withBucket(bucket)
            .withName(video))
        .withNotificationChannel(channel);

    StartFaceSearchResult startPersonCollectionSearchResult =
rek.startFaceSearch(req);
    startJobId=startPersonCollectionSearchResult.getJobId();

}

//Face collection search in video
=====
private static void GetFaceSearchCollectionResults() throws Exception{

    GetFaceSearchResult faceSearchResult=null;
    int maxResults=10;
    String paginationToken=null;

    do {

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

        faceSearchResult = rek.getFaceSearch(
            new GetFaceSearchRequest()
                .withJobId(startJobId)
                .withMaxResults(maxResults)
                .withNextToken(paginationToken)
                .withSortBy(FaceSearchSortBy.TIMESTAMP)
            );

        VideoMetadata videoMetaData=faceSearchResult.getVideoMetadata();

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

```
System.out.println();

//Show search results
List<PersonMatch> matches=
    faceSearchResult.getPersons();

for (PersonMatch match: matches) {
    long milliSeconds=match.getTimestamp();
    System.out.print("Timestamp: " + Long.toString(milliSeconds));
    System.out.println(" Person number: " +
match.getPerson().getIndex());
    List <FaceMatch> faceMatches = match.getFaceMatches();
    if (faceMatches != null) {
        System.out.println("Matches in collection...");
        for (FaceMatch faceMatch: faceMatches){
            Face face=faceMatch.getFace();
            System.out.println("Face Id: "+ face.getFaceId());
            System.out.println("Similarity: " +
faceMatch.getSimilarity().toString());
            System.out.println();
        }
    }
    System.out.println();
}

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

}
```

Di fungsi main, ganti baris:

```
StartLabelDetection(bucket, video);

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

dengan:

```
String collection="collection";
StartFaceSearchCollection(bucket, video, collection);

if (GetSQSMessagesSuccess()==true)
    GetFaceSearchCollectionResults();
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import java.util.List;

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

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
            """;
```

```
    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];

    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

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

    startFaceDetection(rekClient, channel, bucket, video);
    getFaceResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

public static void startFaceDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

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

        StartFaceDetectionRequest faceDetectionRequest =
StartFaceDetectionRequest.builder()
            .jobTag("Faces")
```

```
        .faceAttributes(FaceAttributes.ALL)
        .notificationChannel(channel)
        .video(vidObj)
        .build();

        StartFaceDetectionResponse startLabelDetectionResult =
rekClient.startFaceDetection(faceDetectionRequest);
        startJobId = startLabelDetectionResult.jobId();

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

public static void getFaceResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetFaceDetectionResponse faceDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (faceDetectionResponse != null)
                paginationToken = faceDetectionResponse.nextToken();

            GetFaceDetectionRequest recognitionRequest =
GetFaceDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {

                faceDetectionResponse =
rekClient.getFaceDetection(recognitionRequest);
                status = faceDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
```

```
        System.out.println(yy + " status is: " + status);
        Thread.sleep(1000);
    }
    yy++;
}

finished = false;

// Proceed when the job is done - otherwise VideoMetadata is
null.
VideoMetadata videoMetaData =
faceDetectionResponse.videoMetadata();
System.out.println("Format: " + videoMetaData.format());
System.out.println("Codec: " + videoMetaData.codec());
System.out.println("Duration: " +
videoMetaData.durationMillis());
System.out.println("FrameRate: " + videoMetaData.frameRate());
System.out.println("Job");

// Show face information.
List<FaceDetection> faces = faceDetectionResponse.faces();
for (FaceDetection face : faces) {
    String age = face.face().ageRange().toString();
    String smile = face.face().smile().toString();
    System.out.println("The detected face is estimated to be"
        + age + " years old.");
    System.out.println("There is a smile : " + smile);
}

    } while (faceDetectionResponse != null &&
faceDetectionResponse.nextToken() != null);

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

Python

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

# ===== Face Search =====
def StartFaceSearchCollection(self, collection):
    response = self.rek.start_face_search(Video={'S3Object':
{'Bucket':self.bucket, 'Name':self.video}},
    CollectionId=collection,
    NotificationChannel={'RoleArn':self.roleArn,
'SNSTopicArn':self.snsTopicArn})

    self.startJobId=response['JobId']

    print('Start Job Id: ' + self.startJobId)

def GetFaceSearchCollectionResults(self):
    maxResults = 10
    paginationToken = ''

    finished = False

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

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

        for personMatch in response['Persons']:
            print('Person Index: ' + str(personMatch['Person']['Index']))
            print('Timestamp: ' + str(personMatch['Timestamp']))

            if ('FaceMatches' in personMatch):
                for faceMatch in personMatch['FaceMatches']:
                    print('Face ID: ' + faceMatch['Face']['FaceId'])
                    print('Similarity: ' + str(faceMatch['Similarity']))
                print()
            if 'NextToken' in response:
                paginationToken = response['NextToken']
            else:
```



```
        finished = True
    print()
```

Dalam fungsi main, ganti baris:

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

dengan:

```
collection='tests'
analyzer.StartFaceSearchCollection(collection)

if analyzer.GetSQSMessageSuccess()==True:
    analyzer.GetFaceSearchCollectionResults()
```

Jika Anda sudah menjalankan contoh video selain [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#), kode yang akan diganti mungkin berbeda.

5. Ubah nilai `collection` pada nama koleksi yang Anda buat pada langkah 1.
6. Jalankan kode tersebut. Daftar orang dalam video yang wajahnya cocok dengan yang ada di koleksi input akan ditampilkan. Data pelacakan untuk setiap orang yang cocok juga ditampilkan.

GetFaceSearch respon operasi

Berikut ini adalah respons JSON contoh dari GetFaceSearch.

Respons tersebut meliputi array orang (`Persons`) yang terdeteksi dalam video yang wajahnya cocok dengan wajah dalam koleksi input. Elemen array, [PersonMatch](#), ada untuk setiap kali orang tersebut dicocokkan dalam video. Masing-masing `PersonMatch` mencakup serangkaian kecocokan wajah dari koleksi input [FaceMatch](#), informasi tentang orang yang cocok [PersonDetail](#), dan waktu orang tersebut dicocokkan dalam video.

```
{
  "JobStatus": "SUCCEEDED",
  "NextToken": "IJdbzkZfvBRqj8GPV82BPiZKkLOGCqDIIsNZG/gQsEE5faTVK9JH0z/
xxxxxxxxxxxxxxxxxxxx",
  "Persons": [
```

```
{
  "FaceMatches": [
    {
      "Face": {
        "BoundingBox": {
          "Height": 0.527472972869873,
          "Left": 0.33530598878860474,
          "Top": 0.2161169946193695,
          "Width": 0.35503000020980835
        },
        "Confidence": 99.90239715576172,
        "ExternalImageId": "image.PNG",
        "FaceId": "a2f2e224-bfaa-456c-b360-7c00241e5e2d",
        "ImageId": "eb57ed44-8d8d-5ec5-90b8-6d190daff4c3"
      },
      "Similarity": 98.40909576416016
    }
  ],
  "Person": {
    "BoundingBox": {
      "Height": 0.8694444298744202,
      "Left": 0.2473958283662796,
      "Top": 0.10092592239379883,
      "Width": 0.49427083134651184
    },
    "Face": {
      "BoundingBox": {
        "Height": 0.23000000417232513,
        "Left": 0.42500001192092896,
        "Top": 0.16333332657814026,
        "Width": 0.12937499582767487
      },
      "Confidence": 99.97504425048828,
      "Landmarks": [
        {
          "Type": "eyeLeft",
          "X": 0.46415066719055176,
          "Y": 0.2572723925113678
        },
        {
          "Type": "eyeRight",
          "X": 0.5068183541297913,
          "Y": 0.23705792427062988
        }
      ]
    }
  }
}
```

```
        {
            "Type": "nose",
            "X": 0.49765899777412415,
            "Y": 0.28383663296699524
        },
        {
            "Type": "mouthLeft",
            "X": 0.487221896648407,
            "Y": 0.3452930748462677
        },
        {
            "Type": "mouthRight",
            "X": 0.5142884850502014,
            "Y": 0.33167609572410583
        }
    ],
    "Pose": {
        "Pitch": 15.966927528381348,
        "Roll": -15.547388076782227,
        "Yaw": 11.34195613861084
    },
    "Quality": {
        "Brightness": 44.80223083496094,
        "Sharpness": 99.95819854736328
    }
},
"Index": 0
},
"Timestamp": 0
},
{
    "Person": {
        "BoundingBox": {
            "Height": 0.2177777737379074,
            "Left": 0.7593749761581421,
            "Top": 0.13333334028720856,
            "Width": 0.12250000238418579
        },
        "Face": {
            "BoundingBox": {
                "Height": 0.2177777737379074,
                "Left": 0.7593749761581421,
                "Top": 0.13333334028720856,
                "Width": 0.12250000238418579
            }
        }
    }
}
```

```
    },
    "Confidence": 99.63436889648438,
    "Landmarks": [
      {
        "Type": "eyeLeft",
        "X": 0.8005779385566711,
        "Y": 0.20915353298187256
      },
      {
        "Type": "eyeRight",
        "X": 0.8391435146331787,
        "Y": 0.21049551665782928
      },
      {
        "Type": "nose",
        "X": 0.8191410899162292,
        "Y": 0.2523227035999298
      },
      {
        "Type": "mouthLeft",
        "X": 0.8093273043632507,
        "Y": 0.29053622484207153
      },
      {
        "Type": "mouthRight",
        "X": 0.8366993069648743,
        "Y": 0.29101791977882385
      }
    ],
    "Pose": {
      "Pitch": 3.165884017944336,
      "Roll": 1.4182015657424927,
      "Yaw": -11.151537895202637
    },
    "Quality": {
      "Brightness": 28.910892486572266,
      "Sharpness": 97.61507415771484
    }
  },
  "Index": 1
},
"Timestamp": 0
},
{
```

```
"Person": {
  "BoundingBox": {
    "Height": 0.8388888835906982,
    "Left": 0,
    "Top": 0.15833333134651184,
    "Width": 0.2369791716337204
  },
  "Face": {
    "BoundingBox": {
      "Height": 0.20000000298023224,
      "Left": 0.029999999329447746,
      "Top": 0.2199999988079071,
      "Width": 0.11249999701976776
    },
    "Confidence": 99.85971069335938,
    "Landmarks": [
      {
        "Type": "eyeLeft",
        "X": 0.06842322647571564,
        "Y": 0.3010137975215912
      },
      {
        "Type": "eyeRight",
        "X": 0.10543643683195114,
        "Y": 0.29697132110595703
      },
      {
        "Type": "nose",
        "X": 0.09569807350635529,
        "Y": 0.33701086044311523
      },
      {
        "Type": "mouthLeft",
        "X": 0.0732642263174057,
        "Y": 0.3757539987564087
      },
      {
        "Type": "mouthRight",
        "X": 0.10589495301246643,
        "Y": 0.3722417950630188
      }
    ],
    "Pose": {
      "Pitch": -0.5589138865470886,
```

```
        "Roll": -5.1093974113464355,  
        "Yaw": 18.69594955444336  
    },  
    "Quality": {  
        "Brightness": 43.052337646484375,  
        "Sharpness": 99.68138885498047  
    }  
},  
"Index": 2  
,  
"Timestamp": 0  
}.....  
  
],  
"VideoMetadata": {  
    "Codec": "h264",  
    "DurationMillis": 67301,  
    "Format": "QuickTime / MOV",  
    "FrameHeight": 1080,  
    "FrameRate": 29.970029830932617,  
    "FrameWidth": 1920  
}  
}
```

Mencari wajah dalam koleksi dalam streaming video

Anda dapat menggunakan Amazon Rekognition Video untuk mendeteksi dan mengenali wajah dari koleksi dalam video streaming. Dengan Amazon Rekognition Video Anda dapat membuat stream processor ([CreateStreamProcessor](#)) untuk memulai dan mengelola analisis streaming video.

Untuk mendeteksi wajah yang dikenal dalam aliran video (pencarian wajah), Amazon Rekognition Video menggunakan Amazon Kinesis Video Streams untuk menerima dan memproses aliran video. Hasil analisis adalah output dari Amazon Rekognition Video ke aliran data Kinesis dan kemudian dibaca oleh aplikasi klien Anda.

Untuk menggunakan Amazon Rekognition Video dengan video streaming, aplikasi Anda perlu menerapkan berikut:

- Aliran video Kinesis untuk mengirimkan video streaming ke Amazon Rekognition Video. Untuk informasi selengkapnya, lihat [Panduan Developer Amazon Kinesis Video Streams](#).

- Sebuah pemroses aliran Amazon Rekognition Video untuk mengelola analisis video streaming. Untuk informasi selengkapnya, lihat [Ikhtisar operasi prosesor aliran Video Rekognition Amazon](#).
- Pemakai aliran data Kinesis untuk membaca hasil analisis yang dikirimkan Amazon Rekognition Video ke aliran data Kinesis. Untuk informasi selengkapnya, lihat [Pemakai Kinesis Data Streams](#).

Bagian ini berisi informasi tentang menulis aplikasi yang membuat aliran video Kinesis dan sumber daya lain yang diperlukan, mengalirkan video ke Amazon Rekognition Video, dan menerima hasil analisis.

Topik

- [Mempersiapkan Amazon Rekognition Video Amazon dan sumber daya Amazon Kinesis](#)
- [Mencari wajah dalam video streaming](#)
- [Streaming menggunakan plugin GStreamer](#)
- [Mengatasi masalah video streaming](#)

Mempersiapkan Amazon Rekognition Video Amazon dan sumber daya Amazon Kinesis

Prosedur berikut menjelaskan langkah-langkah yang Anda ambil untuk menyediakan aliran video Kinesis dan sumber daya lain yang digunakan untuk mengenali wajah dalam video streaming.

Prasyarat


Untuk menjalankan prosedur ini, Anda harus AWS SDK for Java menginstal. Untuk informasi selengkapnya, lihat [Memulai dengan Amazon Rekognition](#). Yang Akun AWS Anda gunakan harus memiliki izin akses ke Amazon Rekognition API. Untuk informasi selengkapnya, lihat [Tindakan yang Ditetapkan oleh Amazon Rekognition](#) di Panduan Pengguna IAM.

Mengenali wajah dalam video streaming (AWS SDK)

1. Jika belum, buat peran layanan IAM untuk memberikan akses Amazon Rekognition Video ke aliran video Kinesis dan aliran data Kinesis Anda. Perhatikan ARN. Untuk informasi selengkapnya, lihat [Memberikan akses ke aliran menggunakan AmazonRekognitionServiceRole](#).
2. [Buat koleksi](#) dan catat pengenal koleksi yang Anda gunakan.
3. [Indeks wajah](#) yang ingin Anda cari ke dalam koleksi yang Anda buat pada langkah 2.
4. [Buat Kinesis video streams](#) dan catat Amazon Resource Name (ARN) stream tersebut.

5. [Buat aliran data Kinesis](#). Tambahkan nama aliran dengan AmazonRekognition dan catat ARN aliran.

Anda kemudian dapat [membuat prosesor aliran pencarian wajah](#) dan [memulai prosesor streaming](#) menggunakan nama prosesor aliran yang Anda pilih.

 Note

Anda harus memulai pemroses aliran hanya setelah memverifikasi bahwa Anda dapat menyerap media ke dalam aliran video Kinesis.

Melakukan streaming video ke dalam Amazon Rekognition Video

Untuk melakukan streaming video ke Amazon Rekognition Video, Anda menggunakan SDK Amazon Kinesis Video Streams untuk membuat dan menggunakan aliran video Kinesis. Operasi `PutMedia` menulis fragmen data video ke dalam aliran video Kinesis yang dipakai oleh Amazon Rekognition Video. Setiap fragmen data video biasanya berukuran 2-10 detik dan berisi urutan frame video yang sudah terisi. Amazon Rekognition Video mendukung video yang dikodekan H.264, yang dapat memiliki tiga tipe frame (I, B, dan P). Untuk informasi selengkapnya, lihat [Antar Frame](#). Frame pertama dalam fragmen harus berupa I-frame. Sebuah I-frame dapat didekode secara terpisah dari frame lainnya.

Saat data video masuk ke aliran video Kinesis, Kinesis Video Streams memberikan nomor unik pada fragmen tersebut. Sebagai contoh, lihat [PutMedia API Example](#).

- Jika Anda streaming dari sumber yang disandikan Matroska (MKV), gunakan [PutMedia](#) operasi untuk mengalirkan video sumber ke aliran video Kinesis yang Anda buat. Untuk informasi selengkapnya, lihat [Contoh PutMedia API](#).
- Jika Anda melakukan streaming dari kamera perangkat, lihat [Streaming menggunakan plugin GStreamer](#).

Memberikan Amazon Rekognition Video akses ke sumber daya Anda

Anda menggunakan peran layanan AWS Identity and Access Management (IAM) untuk memberikan akses baca Video Rekognition Amazon ke aliran video Kinesis. Jika Anda menggunakan prosesor aliran penelusuran wajah, Anda menggunakan peran layanan IAM untuk memberikan akses tulis Amazon Rekognition Video ke aliran data Kinesis. Jika Anda menggunakan prosesor aliran

pemantauan keamanan, Anda menggunakan peran IAM untuk memberikan akses Video Rekognition Amazon ke bucket Amazon S3 Anda dan ke topik Amazon SNS.

Memberikan akses untuk prosesor aliran pencarian wajah

Anda dapat membuat kebijakan izin yang memungkinkan Amazon Rekognition Video mengakses aliran video Kinesis dan aliran data Kinesis individu.

Untuk memberikan akses Video Rekognition Amazon untuk prosesor aliran pencarian wajah

1. [Buat kebijakan izin baru dengan editor kebijakan IAM JSON](#), dan gunakan kebijakan berikut. Ganti `video-arn` dengan ARN dari aliran video Kinesis yang diinginkan. Jika Anda menggunakan prosesor aliran pencarian wajah, ganti `data-arn` dengan ARN dari aliran data Kinesis yang diinginkan.

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

2. [Buat peran layanan IAM](#), atau perbarui peran layanan IAM yang ada. Gunakan informasi berikut ini untuk membuat peran layanan IAM:
 1. Pilih Rekognition untuk nama layanan.
 2. Pilih Rekognition untuk kasus penggunaan peran layanan.

3. Lampirkan kebijakan izin yang Anda buat pada langkah 1.
3. Catat ARN peran layanan. Anda memerlukannya untuk memulai operasi analisis video.

Memberikan akses ke aliran menggunakan `AmazonRekognitionServiceRole`

Sebagai opsi alternatif untuk mengatur akses ke aliran video Kinesis dan aliran data, Anda dapat menggunakan kebijakan izin. `AmazonRekognitionServiceRole` IAM menyediakan kasus penggunaan peran layanan Rekognition, yang ketika digunakan dengan kebijakan izin `AmazonRekognitionServiceRole`, dapat menulis di beberapa aliran data Kinesis dan membaca dari semua aliran video Kinesis Anda. Untuk memberikan akses tulis Amazon Rekognition Video ke beberapa aliran data Kinesis, Anda dapat menambahkan nama aliran data Kinesis dengan — misalnya, `AmazonRekognitionAmazonRekognitionMyDataStreamName`

Untuk memberikan Amazon Rekognition Video akses ke aliran video Kinesis dan aliran data Kinesis

1. [Buat peran layanan IAM](#). Gunakan informasi berikut ini untuk membuat peran layanan IAM:
 1. Pilih Rekognition untuk nama layanan.
 2. Pilih Rekognition untuk kasus penggunaan peran layanan.
 3. Pilih kebijakan `AmazonRekognitionServiceRole` izin, yang memberikan akses tulis Amazon Rekognition Video ke aliran data Kinesis yang diawali dengan `AmazonRekognition` dan akses baca ke semua aliran video Kinesis Anda.
2. Untuk memastikan Anda Akun AWS aman, batasi ruang lingkup akses Rekognition hanya ke sumber daya yang Anda gunakan. Ini dapat dilakukan dengan melampirkan kebijakan kepercayaan ke peran layanan IAM Anda. Untuk informasi tentang cara melakukannya, lihat [Pencegahan wakil bingung lintas layanan](#).
3. Catat Amazon Resource Name (ARN) dari peran layanan tersebut. Anda memerlukannya untuk memulai operasi analisis video.

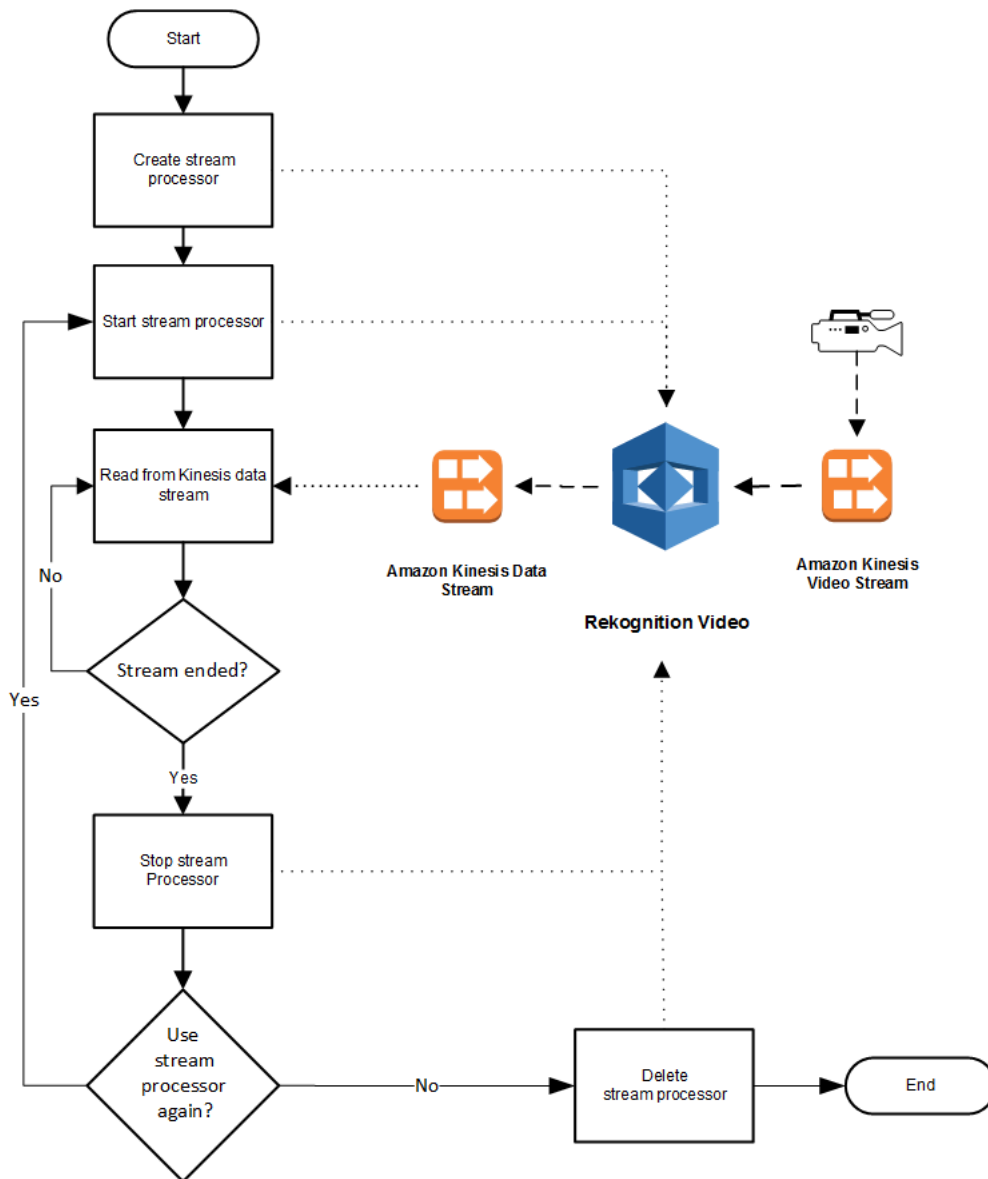
Mencari wajah dalam video streaming

Amazon Rekognition Video dapat mencari wajah dalam koleksi yang cocok dengan wajah yang terdeteksi dalam video streaming. Untuk informasi selengkapnya tentang koleksi, lihat [Mencari wajah dalam koleksi](#).

Topik

- [Membuat prosesor aliran pencarian wajah Amazon Rekognition Video](#)
- [Memulai prosesor aliran pencarian wajah Amazon Rekognition Video](#)
- [Menggunakan prosesor aliran untuk pencarian wajah \(contoh Java V2\)](#)
- [Menggunakan prosesor aliran untuk pencarian wajah \(contoh Java V1\)](#)
- [Membaca hasil analisis video streaming](#)
- [Referensi: Catatan pengenalan wajah Kinesis](#)

Diagram berikut menunjukkan bagaimana Amazon Rekognition Video mendeteksi dan mengenali wajah dalam video streaming.



Membuat prosesor aliran pencarian wajah Amazon Rekognition Video

Sebelum Anda dapat menganalisis video streaming, Anda membuat prosesor streaming Video Rekognition Amazon (). [CreateStreamProcessor](#) Pemroses aliran berisi informasi tentang aliran data Kinesis dan aliran video Kinesis. Ini juga berisi pengenalan untuk koleksi yang berisi wajah yang ingin Anda kenali dalam video streaming input. Anda juga menentukan nama untuk pemroses aliran. Berikut ini adalah contoh JSON untuk permintaan `CreateStreamProcessor`.

```
{
  "Name": "streamProcessorForCam",
  "Input": {
    "KinesisVideoStream": {
      "Arn": "arn:aws:kinesisvideo:us-east-1:nnnnnnnnnnnn:stream/
inputVideo"
    }
  },
  "Output": {
    "KinesisDataStream": {
      "Arn": "arn:aws:kinesis:us-east-1:nnnnnnnnnnnn:stream/outputData"
    }
  },
  "RoleArn": "arn:aws:iam::nnnnnnnnnnnn:role/roleWithKinesisPermission",
  "Settings": {
    "FaceSearch": {
      "CollectionId": "collection-with-100-faces",
      "FaceMatchThreshold": 85.5
    }
  }
}
```

Berikut ini adalah contoh respons dari `CreateStreamProcessor`.

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

Memulai prosesor aliran pencarian wajah Amazon Rekognition Video

Anda mulai menganalisis streaming video dengan menelepon [StartStreamProcessor](#) dengan nama prosesor streaming yang Anda tentukan `CreateStreamProcessor`. Berikut ini adalah contoh JSON untuk permintaan `StartStreamProcessor`.

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

Jika pemroses aliran berhasil dimulai, respons HTTP 200 diberikan, bersama dengan body JSON yang kosong.

Menggunakan prosesor aliran untuk pencarian wajah (contoh Java V2)

Kode contoh berikut menunjukkan cara memanggil berbagai operasi prosesor aliran, seperti [CreateStreamProcessor](#) dan [StartStreamProcessor](#), menggunakan AWS SDK for Java versi 2.

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateStreamProcessorRequest;
import software.amazon.awssdk.services.rekognition.model.CreateStreamProcessorResponse;
import software.amazon.awssdk.services.rekognition.model.FaceSearchSettings;
import software.amazon.awssdk.services.rekognition.model.KinesisDataStream;
import software.amazon.awssdk.services.rekognition.model.KinesisVideoStream;
import software.amazon.awssdk.services.rekognition.model.ListStreamProcessorsRequest;
import software.amazon.awssdk.services.rekognition.model.ListStreamProcessorsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.StreamProcessor;
import software.amazon.awssdk.services.rekognition.model.StreamProcessorInput;
import software.amazon.awssdk.services.rekognition.model.StreamProcessorSettings;
import software.amazon.awssdk.services.rekognition.model.StreamProcessorOutput;
import software.amazon.awssdk.services.rekognition.model.StartStreamProcessorRequest;
import
  software.amazon.awssdk.services.rekognition.model.DescribeStreamProcessorRequest;
import
  software.amazon.awssdk.services.rekognition.model.DescribeStreamProcessorResponse;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateStreamProcessor {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <role> <kinInputStream> <kinOutputStream>
<collectionName> <StreamProcessorName>

            Where:
                role - The ARN of the AWS Identity and Access
Management (IAM) role to use. \s
                kinInputStream - The ARN of the Kinesis video
stream.\s
                kinOutputStream - The ARN of the Kinesis data
stream.\s
                collectionName - The name of the collection to use
that contains content. \s
                StreamProcessorName - The name of the Stream
Processor. \s

            """;

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

        String role = args[0];
        String kinInputStream = args[1];
        String kinOutputStream = args[2];
        String collectionName = args[3];
        String streamProcessorName = args[4];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();
```

```
        processCollection(rekClient, streamProcessorName, kinInputStream,
kinOutputStream, collectionName,
                        role);
        startSpecificStreamProcessor(rekClient, streamProcessorName);
        listStreamProcessors(rekClient);
        describeStreamProcessor(rekClient, streamProcessorName);
        deleteSpecificStreamProcessor(rekClient, streamProcessorName);
    }

    public static void listStreamProcessors(RekognitionClient rekClient) {
        ListStreamProcessorsRequest request =
ListStreamProcessorsRequest.builder()
                            .maxResults(15)
                            .build();

        ListStreamProcessorsResponse listStreamProcessorsResult =
rekClient.listStreamProcessors(request);
        for (StreamProcessor streamProcessor :
listStreamProcessorsResult.streamProcessors()) {
            System.out.println("StreamProcessor name - " +
streamProcessor.name());
            System.out.println("Status - " + streamProcessor.status());
        }
    }

    private static void describeStreamProcessor(RekognitionClient rekClient, String
StreamProcessorName) {
        DescribeStreamProcessorRequest streamProcessorRequest =
DescribeStreamProcessorRequest.builder()
                                .name(StreamProcessorName)
                                .build();

        DescribeStreamProcessorResponse describeStreamProcessorResult =
rekClient
                                .describeStreamProcessor(streamProcessorRequest);
        System.out.println("Arn - " +
describeStreamProcessorResult.streamProcessorArn());
        System.out.println("Input kinesisVideo stream - "
+
describeStreamProcessorResult.input().kinesisVideoStream().arn());
        System.out.println("Output kinesisData stream - "
+
describeStreamProcessorResult.output().kinesisDataStream().arn());
    }
}
```

```
        System.out.println("RoleArn - " +
describeStreamProcessorResult.roleArn());
        System.out.println(
            "CollectionId - "
                +
describeStreamProcessorResult.settings().faceSearch().collectionId());
        System.out.println("Status - " +
describeStreamProcessorResult.status());
        System.out.println("Status message - " +
describeStreamProcessorResult.statusMessage());
        System.out.println("Creation timestamp - " +
describeStreamProcessorResult.creationTimestamp());
        System.out.println("Last update timestamp - " +
describeStreamProcessorResult.lastUpdateTimestamp());
    }

    private static void startSpecificStreamProcessor(RekognitionClient rekClient,
String StreamProcessorName) {
        try {
            StartStreamProcessorRequest streamProcessorRequest =
StartStreamProcessorRequest.builder()
                .name(StreamProcessorName)
                .build();

            rekClient.startStreamProcessor(streamProcessorRequest);
            System.out.println("Stream Processor " + StreamProcessorName +
" started.");

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

    private static void processCollection(RekognitionClient rekClient, String
StreamProcessorName,
        String kinInputStream, String kinOutputStream, String
collectionName, String role) {
        try {
            KinesisVideoStream videoStream = KinesisVideoStream.builder()
                .arn(kinInputStream)
                .build();

            KinesisDataStream dataStream = KinesisDataStream.builder()
```



```
                .arn(kinOutputStream)
                .build();

        StreamProcessorOutput processorOutput =
StreamProcessorOutput.builder()

                .kinesisDataStream(dataStream)
                .build();

        StreamProcessorInput processorInput =
StreamProcessorInput.builder()

                .kinesisVideoStream(videoStream)
                .build();

        FaceSearchSettings searchSettings =
FaceSearchSettings.builder()

                .faceMatchThreshold(75f)
                .collectionId(collectionName)
                .build();

        StreamProcessorSettings processorSettings =
StreamProcessorSettings.builder()

                .faceSearch(searchSettings)
                .build();

        CreateStreamProcessorRequest processorRequest =
CreateStreamProcessorRequest.builder()

                .name(StreamProcessorName)
                .input(processorInput)
                .output(processorOutput)
                .roleArn(role)
                .settings(processorSettings)
                .build();

        CreateStreamProcessorResponse response =
rekClient.createStreamProcessor(processorRequest);
        System.out.println("The ARN for the newly create stream
processor is "

                + response.streamProcessorArn());

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

```
        private static void deleteSpecificStreamProcessor(RekognitionClient rekClient,
String StreamProcessorName) {
            rekClient.stopStreamProcessor(a -> a.name(StreamProcessorName));
            rekClient.deleteStreamProcessor(a -> a.name(StreamProcessorName));
            System.out.println("Stream Processor " + StreamProcessorName + "
deleted.");
        }
    }
```

Menggunakan prosesor aliran untuk pencarian wajah (contoh Java V1)

Kode contoh berikut menunjukkan cara memanggil berbagai operasi prosesor aliran, seperti [CreateStreamProcessor](#) dan [StartStreamProcessor](#), menggunakan Java V1. Contoh ini mencakup kelas manajer prosesor aliran (`StreamManager`) yang menyediakan metode untuk memanggil operasi prosesor aliran. Kelas starter (`Starter`) membuat `StreamManager` objek dan memanggil berbagai operasi.

Untuk mengonfigurasi contoh:

1. Atur nilai bidang anggota kelas `Starter` dengan nilai yang Anda inginkan.
2. Dalam fungsi kelas `Starter` `main`, batalkan komentar panggilan fungsi yang diinginkan.

Kelas Starter

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

// Starter class. Use to create a StreamManager class
// and call stream processor operations.
package com.amazonaws.samples;
import com.amazonaws.samples.*;

public class Starter {

    public static void main(String[] args) {

        String streamProcessorName="Stream Processor Name";
        String kinesisVideoStreamArn="Kinesis Video Stream Arn";
```

```
String kinesisDataStreamArn="Kinesis Data Stream Arn";
String roleArn="Role Arn";
String collectionId="Collection ID";
Float matchThreshold=50F;

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

StreamManager kelas

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

```
    }  
  }  
}
```

Membaca hasil analisis video streaming

Anda dapat menggunakan Pustaka Klien Amazon Kinesis Data Streams untuk memakai hasil analisis yang dikirim ke aliran output Amazon Kinesis Data Streams. Untuk informasi selengkapnya, lihat [Membaca Data dari Aliran Data Kinesis](#). Amazon Rekognition Video menempatkan catatan frame JSON untuk setiap frame yang dianalisis ke dalam aliran output Kinesis. Amazon Rekognition Video tidak menganalisis setiap frame yang diteruskan melalui aliran video Kinesis.

Catatan frame yang dikirim ke aliran data Kinesis berisi informasi tentang di fragmen aliran video Kinesis mana frame berada, lokasi frame berada dalam fragment, dan wajah yang dikenali dalam frame. Hal ini juga mencakup informasi status untuk pemroses aliran. Untuk informasi selengkapnya, lihat [Referensi: Catatan pengenalan wajah Kinesis](#).

Pustaka Parser Amazon Kinesis Video Streams berisi contoh uji yang memakai hasil Amazon Rekognition Video dan mengintegrasikannya dengan aliran video Kinesis asli. Untuk informasi selengkapnya, lihat [Menampilkan hasil Rekognition dengan Kinesis Video Streams secara lokal](#).

Amazon Rekognition Video mengalirkan informasi analisis Amazon Rekognition Video ke aliran data Kinesis. Berikut ini adalah contoh JSON untuk satu catatan.

```
{  
  "InputInformation": {  
    "KinesisVideo": {  
      "StreamArn": "arn:aws:kinesisvideo:us-west-2:nnnnnnnnnnn:stream/stream-name",  
      "FragmentNumber": "91343852333289682796718532614445757584843717598",  
      "ServerTimestamp": 1510552593.455,  
      "ProducerTimestamp": 1510552593.193,  
      "FrameOffsetInSeconds": 2  
    }  
  },  
  "StreamProcessorInformation": {  
    "Status": "RUNNING"  
  },  
  "FaceSearchResponse": [  
    {  
      "DetectedFace": {  
        "BoundingBox": {  
          "Height": 0.075,  

```

```
    "Width": 0.05625,
    "Left": 0.428125,
    "Top": 0.40833333
  },
  "Confidence": 99.975174,
  "Landmarks": [
    {
      "X": 0.4452057,
      "Y": 0.4395594,
      "Type": "eyeLeft"
    },
    {
      "X": 0.46340984,
      "Y": 0.43744427,
      "Type": "eyeRight"
    },
    {
      "X": 0.45960626,
      "Y": 0.4526856,
      "Type": "nose"
    },
    {
      "X": 0.44958648,
      "Y": 0.4696949,
      "Type": "mouthLeft"
    },
    {
      "X": 0.46409217,
      "Y": 0.46704912,
      "Type": "mouthRight"
    }
  ],
  "Pose": {
    "Pitch": 2.9691637,
    "Roll": -6.8904796,
    "Yaw": 23.84388
  },
  "Quality": {
    "Brightness": 40.592964,
    "Sharpness": 96.09616
  }
},
"MatchedFaces": [
  {
```



```
    "Similarity": 88.863960,
    "Face": {
      "BoundingBox": {
        "Height": 0.557692,
        "Width": 0.749838,
        "Left": 0.103426,
        "Top": 0.206731
      },
      "FaceId": "ed1b560f-d6af-5158-989a-ff586c931545",
      "Confidence": 99.999201,
      "ImageId": "70e09693-2114-57e1-807c-50b6d61fa4dc",
      "ExternalImageId": "matchedImage.jpeg"
    }
  ]
}
}
```

Dalam contoh JSON, catat hal berikut:

- **InputInformation**— Informasi tentang aliran video Kinesis yang digunakan untuk mengalirkan video ke Amazon Rekognition Video. Untuk informasi selengkapnya, lihat [InputInformation](#).
- **StreamProcessorInformation**— Informasi status untuk prosesor aliran Video Rekognition Amazon. Satu-satunya nilai yang mungkin untuk bidang Status adalah BERJALAN. Untuk informasi selengkapnya, lihat [StreamProcessorInformation](#).
- **FaceSearchResponse**— Berisi informasi tentang wajah dalam video streaming yang cocok dengan wajah dalam koleksi input. [FaceSearchResponse](#) berisi [DetectedFace](#) objek, yang merupakan wajah yang terdeteksi dalam bingkai video yang dianalisis. Untuk setiap wajah yang terdeteksi, array `MatchedFaces` berisi array objek wajah yang cocok ([MatchedFace](#)) yang ditemukan dalam koleksi masukan, bersama dengan skor kesamaan.

Memetakan aliran video Kinesis ke aliran data Kinesis

Anda mungkin ingin memetakan frame aliran video Kinesis ke frame yang dianalisis yang dikirim ke aliran data Kinesis. Misalnya, selama tampilan video streaming, Anda mungkin ingin menampilkan kotak-kotak di sekitar wajah orang yang dikenal. Koordinat kotak batas dikirim sebagai bagian dari Catatan Pengenal Wajah Kinesis ke aliran data Kinesis. Untuk menampilkan kotak pembatas dengan benar, Anda perlu memetakan informasi waktu yang dikirim dengan Catatan Pengenal Wajah Kinesis dengan frame yang sesuai dalam aliran video Kinesis sumber.

Teknik yang Anda gunakan untuk memetakan aliran video Kinesis ke aliran data Kinesis tergantung pada apakah Anda sedang melakukan streaming media langsung (seperti video streaming langsung), atau jika Anda sedang melakukan streaming media yang diarsipkan (seperti video yang tersimpan).

Pemetaan saat Anda melakukan streaming media langsung

Untuk memetakan frame aliran video Kinesis ke frame aliran data Kinesis

1. Atur parameter `FragmentTimeCodeType` input [PutMedia](#) operasi ke `RELATIVE`.
2. Panggil `PutMedia` untuk mengirimkan media langsung ke dalam aliran video Kinesis.
3. Saat Anda menerima Catatan Pengenalan Wajah Kinesis dari aliran data Kinesis, simpan nilai `ProducerTimestamp` dan `FrameOffsetInSeconds` dari bidang [KinesisVideo](#).
4. Hitung stempel waktu yang sesuai dengan frame aliran video Kinesis dengan menambahkan nilai bidang `ProducerTimestamp` dan `FrameOffsetInSeconds` bersama-sama.

Pemetaan saat Anda melakukan streaming media yang diarsipkan

Untuk memetakan frame aliran video Kinesis ke frame aliran data Kinesis

1. Panggilan [PutMedia](#) untuk mengirimkan media yang diarsipkan ke dalam aliran video Kinesis.
2. Ketika Anda menerima objek `Acknowledgement` dari respons operasi `PutMedia`, simpan nilai bidang `FragmentNumber` dari bidang [Muatan](#). `FragmentNumber` adalah nomor fragmen untuk kluster MKV.
3. Saat Anda menerima Catatan Pengenalan Wajah Kinesis dari aliran data Kinesis, simpan nilai bidang `FrameOffsetInSeconds` dari bidang [KinesisVideo](#).
4. Hitung pemetaan dengan menggunakan nilai-nilai `FrameOffsetInSeconds` dan `FragmentNumber` yang Anda disimpan dalam langkah 2 dan 3. `FrameOffsetInSeconds` adalah offset ke dalam fragmen dengan `FragmentNumber` khusus yang dikirim ke aliran data Amazon Kinesis. Untuk informasi selengkapnya tentang mendapatkan frame video untuk nomor fragmen tertentu, lihat [Media Amazon Kinesis Video Streams yang Diarsipkan](#).

Menampilkan hasil Rekognition dengan Kinesis Video Streams secara lokal

[Anda dapat melihat hasil Video Rekognition Amazon yang ditampilkan di feed Anda dari Amazon Kinesis Video Streams menggunakan contoh pengujian Perpustakaan Pengurai Video Kinesis Amazon Kinesis yang disediakan di - Contoh Rekognition. KinesisVideo](#)

`KinesisVideoRekognitionIntegrationExample` menampilkan kotak batas pada wajah

yang terdeteksi dan membuat video secara lokal melalui JFrame. Sebelum menjalankan proses ini, pastikan Anda telah berhasil menghubungkan input media dari kamera perangkat ke aliran video Kinesis dan memulai Pemroses Aliran Amazon Rekognition. Untuk informasi selengkapnya, lihat [Streaming menggunakan plugin GStreamer](#).

Langkah 1: Memasang Pustaka Parser Kinesis Video Streams

Untuk membuat direktori dan mengunduh repositori Github, jalankan perintah berikut:

```
$ git clone https://github.com/aws/amazon-kinesis-video-streams-parser-library.git
```

Arahkan ke direktori pustaka dan jalankan perintah Maven berikut untuk melakukan instalasi bersih:

```
$ mvn clean install
```

Langkah 2: Mengonfigurasi uji contoh Kinesis Video Streams dan Rekognition

Buka file `KinesisVideoRekognitionIntegrationExampleTest.java`. Hapus `@Ignore` tepat setelah header kelas. Isi kolom data dengan informasi dari sumber daya Amazon Kinesis dan Amazon Rekognition Anda. Untuk informasi selengkapnya, lihat [Mempersiapkan Amazon Rekognition Video Amazon dan sumber daya Amazon Kinesis](#). Jika Anda melakukan streaming video ke aliran video Kinesis Anda, hapus parameter `inputStream`.

Lihat contoh kode berikut ini:

```
RekognitionInput rekognitionInput = RekognitionInput.builder()
    .kinesisVideoStreamArn("arn:aws:kinesisvideo:us-east-1:123456789012:stream/
rekognition-test-video-stream")
    .kinesisDataStreamArn("arn:aws:kinesis:us-east-1:123456789012:stream/
AmazonRekognition-rekognition-test-data-stream")
    .streamingProcessorName("rekognition-test-stream-processor")
    // Refer how to add face collection :
    // https://docs.aws.amazon.com/rekognition/latest/dg/add-faces-to-collection-
procedure.html
    .faceCollectionId("rekognition-test-face-collection")
    .iamRoleArn("rekognition-test-IAM-role")
    .matchThreshold(0.95f)
    .build();

KinesisVideoRekognitionIntegrationExample example =
    KinesisVideoRekognitionIntegrationExample.builder()
```

```
.region(Regions.US_EAST_1)
.kvsStreamName("rekognition-test-video-stream")
.kdsStreamName("AmazonRekognition-rekognition-test-data-stream")
.rekognitionInput(rekognitionInput)
.credentialsProvider(new ProfileCredentialsProvider())
// NOTE: Comment out or delete the inputStream parameter if you are streaming video,
otherwise
// the test will use a sample video.
//.inputStream(TestResourceUtil.getTestInputStream("bezos_vogels.mkv"))
.build();
```

Langkah 3: Menjalankan uji contoh integrasi Kinesis Video Streams dan Rekognition

Pastikan aliran video Kinesis Anda menerima input media jika Anda melakukan streaming dan mulai menganalisis streaming Anda dengan Pemroses Aliran Amazon Rekognition Video yang berjalan. Untuk informasi selengkapnya, lihat [Ikhtisar operasi prosesor aliran Video Rekognition Amazon](#). Jalankan kelas `KinesisVideoRekognitionIntegrationExampleTest` sebagai tes JUnit. Setelah tertunda beberapa saat, jendela baru akan terbuka dengan umpan video dari aliran video Kinesis dengan kotak batas yang ditarik ke wajah yang terdeteksi.

Note


Wajah dalam koleksi yang digunakan dalam contoh ini harus memiliki Id Gambar Eksternal (nama file) yang ditentukan dalam format ini agar label kotak pembatas menampilkan teks yang bermakna: `PersonName 1-Tepercaya`, `PersonName 2-Penyusup`, `3-Netral`, dll. `PersonName` Label juga dapat diberi kode warna dan dapat disesuaikan dalam file.java. `FaceType`

Referensi: Catatan pengenalan wajah Kinesis

Amazon Rekognition Video dapat mengenali wajah dalam video streaming. Untuk setiap frame yang dianalisis, Amazon Rekognition Video menghasilkan catatan frame JSON ke aliran data Kinesis. Amazon Rekognition Video tidak menganalisis setiap frame yang diteruskan melalui aliran video Kinesis.

Catatan frame JSON berisi informasi tentang input dan output stream, status pemroses aliran, dan informasi tentang wajah yang diakui dalam frame yang dianalisis. Bagian ini berisi informasi referensi untuk catatan bingkai JSON.

Berikut ini adalah sintaksis JSON untuk catatan aliran data Kinesis. Untuk informasi selengkapnya, lihat [Bekerja dengan acara video streaming](#).

 Note

API Amazon Rekognition Video bekerja dengan membandingkan wajah di input aliran Anda dengan koleksi wajah, dan memberikan hasil wajah yang paling cocok yang ditemukan, bersama dengan skor kesamaan.

```
{
  "InputInformation": {
    "KinesisVideo": {
      "StreamArn": "string",
      "FragmentNumber": "string",
      "ProducerTimestamp": number,
      "ServerTimestamp": number,
      "FrameOffsetInSeconds": number
    }
  },
  "StreamProcessorInformation": {
    "Status": "RUNNING"
  },
  "FaceSearchResponse": [
    {
      "DetectedFace": {
        "BoundingBox": {
          "Width": number,
          "Top": number,
          "Height": number,
          "Left": number
        },
        "Confidence": number,
        "Landmarks": [
          {
            "Type": "string",
            "X": number,
            "Y": number
          }
        ],
        "Pose": {
          "Pitch": number,
```

```

        "Roll": number,
        "Yaw": number
    },
    "Quality": {
        "Brightness": number,
        "Sharpness": number
    }
},
"MatchedFaces": [
    {
        "Similarity": number,
        "Face": {
            "BoundingBox": {
                "Width": number,
                "Top": number,
                "Height": number,
                "Left": number
            },
            "Confidence": number,
            "ExternalImageId": "string",
            "FaceId": "string",
            "ImageId": "string"
        }
    }
]
}

```

Catatan JSON

Catatan JSON mencakup informasi tentang frame yang diproses oleh Amazon Rekognition Video. Catatan tersebut mencakup informasi tentang video streaming tersebut, status frame yang dianalisis, dan informasi tentang wajah yang dikenali dalam frame.

InputInformation

Informasi tentang aliran video Kinesis yang digunakan untuk melakukan streaming video ke Amazon Rekognition Video.

Tipe: Objek [InputInformation](#)

StreamProcessorInformation

Informasi tentang pemroses aliran Amazon Rekognition Video. Ini termasuk informasi status untuk status pemroses aliran saat ini.

Tipe: Objek [StreamProcessorInformation](#)

FaceSearchResponse

Informasi tentang wajah yang terdeteksi dalam frame video streaming dan wajah paling cocok yang ditemukan dalam koleksi input.

Tipe: Array objek [FaceSearchResponse](#)

InputInformation

Informasi tentang aliran video sumber yang digunakan oleh Amazon Rekognition Video. Untuk informasi selengkapnya, lihat [Bekerja dengan acara video streaming](#).

KinesisVideo

Tipe: Objek [KinesisVideo](#)

KinesisVideo

Informasi tentang aliran video Kinesis yang mengalirkan video sumber ke Amazon Rekognition Video. Untuk informasi selengkapnya, lihat [Bekerja dengan acara video streaming](#).

StreamArn

Amazon Resource Name (ARN) dari aliran video Kinesis.

Jenis: String

FragmentNumber

Fragmen video streaming yang berisi frame yang diwakilkan oleh catatan ini.

Jenis: String

ProducerTimestamp

Stempel waktu Unix sisi produsen fragmen. Untuk informasi lebih lanjut, lihat [PutMedia](#).

Jenis: Angka

ServerTimestamp

Stempel waktu Unix sisi server fragmen. Untuk informasi lebih lanjut, lihat [PutMedia](#).

Jenis: Angka

FrameOffsetInSeconds

Offset frame (dalam detik) di dalam fragmen.

Tipe: Angka

StreamProcessorInformation

Informasi status tentang pemroses aliran.

Status

Status pemroses aliran saat ini. Satu nilai yang mungkin adalah BERJALAN.

Tipe: String

FaceSearchResponse

Informasi tentang wajah yang terdeteksi dalam frame video streaming dan wajah dalam koleksi yang sesuai dengan wajah yang terdeteksi. Anda menentukan koleksi dalam panggilan ke [CreateStreamProcessor](#). Untuk informasi selengkapnya, lihat [Bekerja dengan acara video streaming](#).

DetectedFace

Detail wajah untuk wajah yang terdeteksi dalam frame video yang dianalisis.

Tipe: Objek [DetectedFace](#)

MatchedFaces

Array detail wajah untuk wajah dalam koleksi yang cocok dengan wajah yang terdeteksi di [DetectedFace](#).

Tipe: Array objek [MatchedFace](#)

DetectedFace

Informasi tentang wajah yang terdeteksi dalam frame video streaming. Wajah yang cocok dalam koleksi input tersedia di bidang objek [MatchedFace](#).

BoundingBox

Kotak batas mengoordinasikan wajah yang terdeteksi dalam bingkai video yang dianalisis. BoundingBox Objek memiliki sifat yang sama dengan BoundingBox objek yang digunakan untuk analisis gambar.

Tipe: Objek [BoundingBox](#)

Keyakinan

Tingkat kepercayaan (1-100) yang dimiliki Amazon Rekognition Video bahwa wajah yang terdeteksi adalah benar-benar wajah. 1 adalah kepercayaan terendah, 100 adalah yang tertinggi.

Tipe: Angka

Tengara

Array penanda wajah

Jenis: [Array objek Landmark](#)

Pose

Menunjukkan pose wajah sebagaimana ditentukan oleh pitch, roll, dan yaw.

Jenis: [Pose](#) objek

Kualitas

Mengidentifikasi kecerahan dan ketajaman citra wajah.

Tipe: Objek [ImageQuality](#)

MatchedFace

Informasi tentang wajah yang cocok dengan wajah yang terdeteksi dalam bingkai video yang dianalisis.

Wajah

Informasi kecocokan wajah untuk wajah dalam koleksi input yang cocok dengan wajah di objek [DetectedFace](#).

Jenis: Objek [wajah](#)

Kesamaan

Tingkat kepercayaan (1-100) bahwa tersebut wajah cocok. 1 adalah kepercayaan terendah, 100 adalah yang tertinggi.

Jenis: Angka

Streaming menggunakan plugin GStreamer

Amazon Rekognition Video dapat menganalisis video streaming siaran langsung dari kamera perangkat. Untuk mengakses input media dari sumber perangkat, Anda perlu memasang GStreamer. GStreamer adalah perangkat lunak kerangka kerja multimedia pihak ke tiga yang menghubungkan sumber media dan alat pemrosesan bersama-sama dalam alur kerja. Anda juga harus memasang [Plugin Produsen Amazon Kinesis Video Streams](#) untuk Gstreamer. Sebelum menjalankan proses ini, pastikan Anda telah berhasil mempersiapkan Amazon Rekognition Video dan sumber daya Amazon Kinesis. Untuk informasi selengkapnya, lihat [Mempersiapkan Amazon Rekognition Video Amazon dan sumber daya Amazon Kinesis](#).

Langkah 1: Memasang Gstreamer

Unduh dan pasang Gstreamer, perangkat lunak platform multi-media pihak ke tiga. Anda dapat menggunakan perangkat lunak pengelolaan paket seperti Homebrew ([Gstreamer pada Homebrew](#)) atau mendapatkannya langsung dari [Situs web Freedesktop](#).

Verifikasi bahwa Gstreamer berhasil terpasang dengan meluncurkan umpan video dengan sumber uji dari terminal baris perintah Anda.

```
$ gst-launch-1.0 videotestsrc ! autovideosink
```

Langkah 2: Memasang plugin Produsen Kinesis Video Streams

Pada bagian ini, Anda akan mengunduh [Pustaka Produsen Amazon Kinesis Video Streams](#) dan memasang plugin Gstreamer Kinesis Video Streams.

Buat direktori dan kloning kode sumber dari repositori Github. Pastikan untuk menyertakan parameter `--recursive`.

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

Ikuti [instruksi yang diberikan oleh pustaka](#) untuk mengonfigurasi dan membangun proyek. Pastikan Anda menggunakan perintah khusus platform untuk sistem operasi Anda. Gunakan parameter `-DBUILD_GSTREAMER_PLUGIN=ON` ketika Anda menjalankan `cmake` untuk memasang plugin GStreamer Kinesis Video Streams. Proyek ini memerlukan paket tambahan berikut yang disertakan dalam instalasi: GCC atau Clang, Curl, Openssl dan Log4cplus. Jika bangunan Anda gagal karena ketiadaan paket, verifikasi bahwa paket terpasang dan ada di PATH Anda. Jika Anda mengalami kesalahan “tidak dapat menjalankan program terhimpun C” saat membangun, jalankan perintah membangun lagi. Kadang-kadang, penghimpun C yang benar tidak ditemukan.

Verifikasi pemasangan plugin Kinesis Video Streams dengan menjalankan perintah berikut.

```
$ gst-inspect-1.0 kvssink
```

Informasi berikut, seperti detail pabrik dan plugin, akan muncul:

Factory Details:

Rank	primary + 10 (266)
Long-name	KVS Sink
Klass	Sink/Video/Network
Description	GStreamer AWS KVS plugin
Author	AWS KVS <kinesis-video-support@amazon.com>

Plugin Details:

Name	kvssink
Description	GStreamer AWS KVS plugin
Filename	/Users/YOUR_USER/amazon-kinesis-video-streams-producer-sdk-cpp/build/libgstkvssink.so
Version	1.0
License	Proprietary
Source module	kvssinkpackage
Binary package	GStreamer
Origin URL	http://gstreamer.net/

...

Langkah 3: Menjalankan Gstreamer dengan plugin Kinesis Video Streams

Sebelum memulai streaming dari kamera perangkat ke Kinesis Video Streams, Anda mungkin perlu mengonversi sumber media menjadi codec yang dapat diterima untuk Kinesis Video Streams. Untuk menentukan spesifikasi dan kemampuan format perangkat yang saat ini terhubung ke mesin Anda, jalankan perintah berikut.

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

Untuk memulai streaming, luncurkan Gstreamer dengan perintah contoh berikut dan tambahkan kredensial Anda dan informasi Amazon Kinesis Video Streams. Anda harus menggunakan access key dan wilayah untuk peran layanan IAM yang Anda buat saat [memberikan Amazon Rekognition akses ke aliran Kinesis Anda](#). Untuk informasi selengkapnya tentang access key, lihat [Mengelola Access Key untuk Pengguna IAM](#) dalam Panduan Pengguna IAM. Selain itu, Anda dapat menyesuaikan parameter argumen format video seperti yang disyaratkan oleh penggunaan Anda dan tersedia dari perangkat Anda.

```
$ gst-launch-1.0 autovideosrc device=/dev/video0 ! videoconvert ! video/x-raw,format=I420,width=640,height=480,framerate=30/1 !
    x264enc bframes=0 key-int-max=45 bitrate=500 ! video/x-h264,stream-
format=avc,alignment=au,profile=baseline !
    kvssink stream-name="YOUR_STREAM_NAME" storage-size=512 access-
key="YOUR_ACCESS_KEY" secret-key="YOUR_SECRET_ACCESS_KEY" aws-region="YOUR_AWS_REGION"
```

Untuk perintah peluncuran lainnya, lihat [Contoh Perintah Peluncuran GStreamer](#).

Note

Jika perintah peluncuran Anda berakhir dengan kesalahan non-negosiasi, periksa output dari Monitor Perangkat dan pastikan bahwa nilai parameter `videoconvert` sesuai dengan kemampuan perangkat Anda.

Anda akan melihat umpan video dari kamera perangkat pada aliran video Kinesis setelah beberapa detik. Untuk mulai mendeteksi dan mencocokkan wajah dengan Amazon Rekognition, mulai

pemroses aliran Amazon Rekognition Video Anda. Untuk informasi selengkapnya, lihat [Ikhtisar operasi prosesor aliran Video Rekognition Amazon](#).

Mengatasi masalah video streaming

Topik ini memberikan informasi pemecahan masalah untuk menggunakan Amazon Rekognition Video dengan video streaming.

Topik

- [Saya tidak tahu apakah prosesor streaming saya berhasil dibuat](#)
- [Saya tidak tahu apakah saya telah mengonfigurasi pemroses aliran saya dengan benar](#)
- [Pemroses aliran saya tidak memberikan hasil](#)
- [Status pemroses aliran saya GAGAL](#)
- [Pemroses aliran saya tidak memberikan hasil yang diharapkan](#)

Saya tidak tahu apakah prosesor streaming saya berhasil dibuat

Gunakan AWS CLI perintah berikut untuk mendapatkan daftar prosesor aliran dan statusnya saat ini.

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

Anda bisa mendapatkan detail tambahan dengan menggunakan AWS CLI perintah berikut. Ganti `stream-processor-name` dengan nama pemroses aliran yang diperlukan.

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

Saya tidak tahu apakah saya telah mengonfigurasi pemroses aliran saya dengan benar

Jika kode Anda tidak mengeluarkan hasil analisis dari Amazon Rekognition Video, pemroses aliran Anda mungkin tidak dikonfigurasi dengan benar. Lakukan hal berikut untuk mengonfirmasi bahwa pemroses aliran Anda dikonfigurasi dengan benar dan dapat mengeluarkan hasil.

Untuk menentukan apakah solusi Anda dikonfigurasi dengan benar

1. Jalankan perintah berikut untuk mengonfirmasi bahwa pemroses aliran Anda berada dalam keadaan berjalan. Ubah `stream-processor-name` ke nama pemroses aliran Anda. Pemroses aliran berjalan jika nilai Status adalah RUNNING. Jika statusnya adalah RUNNING dan Anda

tidak mendapatkan hasil, lihat [Pemroses aliran saya tidak memberikan hasil](#). Jika statusnya adalah FAILED, lihat [Status pemroses aliran saya GAGAL](#).

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

2. Jika prosesor stream Anda berjalan, jalankan Bash berikut atau PowerShell perintah untuk membaca data dari output Kinesis data stream.

Bash

```
SHARD_ITERATOR=$(aws kinesis get-shard-iterator --shard-id shardId-000000000000
--shard-iterator-type TRIM_HORIZON --stream-name kinesis-data-stream-name --query
'ShardIterator')
aws kinesis get-records --shard-iterator $SHARD_ITERATOR
```

PowerShell

```
aws kinesis get-records --shard-iterator ((aws kinesis get-shard-iterator --shard-
id shardId-000000000000 --shard-iterator-type TRIM_HORIZON --stream-name kinesis-
data-stream-name).split('')[4])
```

3. Gunakan [Alat dekode](#) pada situs web Base64 Decode untuk menerjemahkan kode output menjadi string yang dapat dibaca manusia. Untuk informasi selengkapnya, lihat [Langkah 3: Dapatkan Catatan](#).
4. Jika perintah berfungsi dan Anda melihat hasil deteksi wajah di aliran data Kinesis, berarti solusi Anda terkonfigurasi dengan benar. Jika perintah gagal, periksa saran pemecahan masalah lainnya dan lihat [Memberikan Amazon Rekognition Video akses ke sumber daya Anda](#).

Atau, Anda dapat menggunakan AWS Lambda cetak biru kinesis-process-record "" untuk mencatat pesan dari aliran data Kinesis untuk visualisasi berkelanjutan. CloudWatch Ini menimbulkan biaya tambahan untuk AWS Lambda dan CloudWatch

Pemroses aliran saya tidak memberikan hasil

Pemroses aliran Anda mungkin tidak memberikan hasil karena beberapa alasan.

Alasan 1: Prosesor streaming Anda tidak dikonfigurasi dengan benar

Pemroses aliran Anda mungkin tidak dikonfigurasi dengan benar. Untuk informasi selengkapnya, lihat [Saya tidak tahu apakah saya telah mengonfigurasi pemroses aliran saya dengan benar](#).

Alasan 2: Pemroses aliran Anda tidak dalam status BERJALAN

Untuk memecahkan masalah status pemroses aliran

1. Periksa status prosesor aliran dengan AWS CLI perintah berikut.

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

2. Jika nilai dari Status adalah STOPPED, mulai pemroses aliran Anda dengan perintah berikut:

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

3. Jika nilai dari Status adalah FAILED, lihat [Status pemroses aliran saya GAGAL](#).
4. Jika nilai dari Status adalah STARTING, tunggu selama 2 menit dan periksa statusnya dengan mengulangi langkah 1. Jika nilai Status masih STARTING, lakukan hal berikut:
 - a. Hapus pemroses aliran dengan perintah berikut.

```
aws rekognition delete-stream-processor --name stream-processor-name
```

- b. Buat pemroses aliran baru dengan konfigurasi yang sama. Untuk informasi selengkapnya, lihat [Bekerja dengan acara video streaming](#).
 - c. Jika Anda masih mengalami masalah, hubungi AWS Support.
5. Jika nilai dari Status adalah RUNNING, lihat [Alasan 3: Tidak ada data aktif di aliran video Kinesis](#).

Alasan 3: Tidak ada data aktif di aliran video Kinesis

Untuk memeriksa apakah ada data aktif dalam aliran video Kinesis

1. [Masuk ke AWS Management Console, dan buka konsol Amazon Kinesis Video Streams di https://console.aws.amazon.com/kinesisvideo/](https://console.aws.amazon.com/kinesisvideo/).
2. Pilih aliran video Kinesis yang merupakan input untuk pemroses aliran Amazon Rekognition.
3. Jika pratinjau menyatakan Tidak ada data pada aliran, maka tidak ada data dalam input stream untuk Amazon Rekognition Video yang bisa diproses.

Untuk informasi tentang memproduksi video dengan Kinesis Video Streams, lihat [Perpustakaan Produsen Kinesis Video Streams](#).

Status pemroses aliran saya GAGAL

Anda dapat memeriksa status prosesor aliran dengan menggunakan AWS CLI perintah berikut.

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

Jika nilai Status GAGAL, periksa informasi pemecahan masalah untuk pesan kesalahan berikut.

Kesalahan: "Akses ditolak ke Peran"

IAM role yang digunakan oleh pemroses aliran tidak ada atau Amazon Rekognition Video tidak memiliki izin untuk mengambil peran.

Memecahkan masalah akses ke IAM role

1. Masuk ke AWS Management Console dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Dari panel navigasi kiri, pilih Perandan Konfirmasikan bahwa peran itu ada.
3. Jika peran ada, periksa apakah peran tersebut memiliki kebijakan AmazonRekognitionServiceRoleizin.
4. Jika peran tidak ada atau tidak memiliki izin yang benar, lihat [Memberikan Amazon Rekognition Video akses ke sumber daya Anda](#).
5. Mulai prosesor aliran dengan AWS CLI perintah berikut.

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

Kesalahan: "Akses ditolak ke Video Kinesis atau Akses ditolak ke Data Kinesis"

Peran tersebut tidak memiliki akses ke operasi API Kinesis Video Streams GetMedia dan GetDataEndpoint. Ini juga mungkin tidak memiliki akses ke operasi API Kinesis Data Streams PutRecord dan PutRecords.

Untuk memecahkan izin API

1. Masuk ke AWS Management Console dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Buka peran dan pastikan kebijakan izin berikut terlampir.


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

3. Jika salah satu izin hilang, perbarui kebijakan. Untuk informasi selengkapnya, lihat [Memberikan Amazon Rekognition Video akses ke sumber daya Anda](#).

Kesalahan: "Stream *input-video-stream-name* tidak ada"

Input aliran video Kinesis ke pemroses aliran tidak ada atau tidak dikonfigurasi dengan benar.

Memecahkan masalah aliran video Kinesis

1. Gunakan perintah berikut untuk mengonfirmasi bahwa aliran ada:

```
aws kinesisvideo list-streams
```

2. Jika aliran ada, periksa berikut ini.

- Amazon Resource Name (ARN) sama dengan ARN dari aliran input untuk pemroses aliran.
- Aliran video Kinesis berada di Wilayah yang sama dengan pemroses aliran.

Jika prosesor stream tidak dikonfigurasi dengan benar, hapus dengan AWS CLI perintah berikut.

```
aws rekognition delete-stream-processor --name stream-processor-name
```

3. Buat pemroses aliran baru dengan aliran video Kinesis yang diinginkan. Untuk informasi selengkapnya, lihat [Membuat prosesor aliran pencarian wajah Amazon Rekognition Video](#).

Kesalahan: “Koleksi tidak ditemukan”

Koleksi Amazon Rekognition yang digunakan oleh pemroses aliran untuk mencocokkan wajah tidak ada, atau koleksi yang salah yang sedang digunakan.

Untuk mengonfirmasi koleksi

1. Gunakan AWS CLI perintah berikut untuk menentukan apakah koleksi yang diperlukan ada. Ubah `region` ke AWS Wilayah tempat Anda menjalankan prosesor streaming.

```
aws rekognition list-collections --region region
```

Jika koleksi yang diperlukan tidak ada, buat koleksi baru dan tambahkan informasi wajah. Untuk informasi selengkapnya, lihat [Mencari wajah dalam koleksi](#).

2. Dalam panggilan Anda ke [CreateStreamProcessor](#), periksa apakah nilai parameter `CollectionId` input sudah benar.
3. Mulai prosesor aliran dengan AWS CLI perintah berikut.

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

Kesalahan: “Streaming ***output-kinesis-data-stream-name*** di bawah ***akun-id akun tidak ditemukan***”

Aliran data Kinesis keluaran yang digunakan oleh prosesor aliran tidak ada di Wilayah Anda Akun AWS atau tidak di AWS Wilayah yang sama dengan prosesor aliran Anda.

Memecahkan masalah aliran data Kinesis

1. Gunakan AWS CLI perintah berikut untuk menentukan apakah aliran data Kinesis ada. Ubah `region` ke AWS Wilayah tempat Anda menggunakan prosesor streaming.

```
aws kinesis list-streams --region region
```

2. Jika aliran data Kinesis ada, periksa apakah nama aliran data Kinesis tersebut sama dengan nama aliran output yang digunakan oleh pemroses aliran.
3. Jika aliran data Kinesis tidak ada, mungkin ada di Wilayah lain AWS . Aliran data Kinesis harus berada di Wilayah yang sama dengan pemroses aliran.
4. Jika perlu, buat aliran data Kinesis.
 - a. Buat aliran data Kinesis dengan nama yang sama dengan nama yang digunakan oleh pemroses aliran. Untuk informasi selengkapnya, lihat [Langkah 1: Buat aliran data](#).
 - b. Mulai prosesor aliran dengan AWS CLI perintah berikut.

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

Pemroses aliran saya tidak memberikan hasil yang diharapkan

Jika pemroses aliran Anda tidak menampilkan kecocokan wajah yang diharapkan, gunakan informasi berikut.

- [Mencari wajah dalam koleksi](#)
- [Rekomendasi untuk pengaturan kamera \(video streaming\)](#)

Lintasan orang

Amazon Rekognition Video dapat membuat jejak lintasan yang diambil orang dalam video dan memberikan informasi seperti:

- Lokasi orang dalam bingkai video pada saat lintasan mereka dilacak.
- Penanda wajah seperti posisi mata kiri, saat terdeteksi.

Lintasan orang Amazon Rekognition Video dalam video yang disimpan adalah operasi tidak sinkron. Untuk memulai lintasan orang dalam panggilan video [StartPersonTracking](#). Amazon Rekognition Video menerbitkan status penyelesaian analisis video ke topik Amazon Simple Notification Service. Jika analisis video berhasil, hubungi [GetPersonTracking](#) untuk mendapatkan hasil analisis video. Untuk informasi selengkapnya tentang memanggil operasi API Amazon Rekognition Video, lihat [Memanggil operasi Amazon Rekognition Video](#).

Prosedur berikut menunjukkan cara melacak lintasan orang melalui video yang disimpan dalam bucket Amazon S3. Contoh meluas pada kode di [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#) yang menggunakan antrian Amazon Simple Queue Service untuk mendapatkan status penyelesaian permintaan analisis video.

Untuk mendeteksi orang dalam video yang disimpan dalam bucket Amazon S3 (SDK)

1. Lakukan [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#).
2. Tambahkan kode berikut ke kelas VideoDetect yang Anda buat di langkah 1.

Java

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

//
Persons=====
    private static void StartPersonDetection(String bucket, String video)
    throws Exception{
```

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

StartPersonTrackingRequest req = new StartPersonTrackingRequest()
    .withVideo(new Video()
        .withS3Object(new S3Object()
            .withBucket(bucket)
            .withName(video)))
    .withNotificationChannel(channel);

StartPersonTrackingResult startPersonDetectionResult =
rek.startPersonTracking(req);
startJobId=startPersonDetectionResult.getJobId();

}

private static void GetPersonDetectionResults() throws Exception{
    int maxResults=10;
    String paginationToken=null;
    GetPersonTrackingResult personTrackingResult=null;

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

        personTrackingResult = rek.getPersonTracking(new
GetPersonTrackingRequest()
            .withJobId(startJobId)
            .withNextToken(paginationToken)
            .withSortBy(PersonTrackingSortBy.TIMESTAMP)
            .withMaxResults(maxResults));

        VideoMetadata
videoMetaData=personTrackingResult.getVideoMetadata();

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

```
        System.out.println("FrameRate: " +
videoMetaData.getFrameRate());

        //Show persons, confidence and detection times
        List<PersonDetection> detectedPersons=
personTrackingResult.getPersons();

        for (PersonDetection detectedPerson: detectedPersons) {

            long seconds=detectedPerson.getTimestamp()/1000;
            System.out.print("Sec: " + Long.toString(seconds) + " ");
            System.out.println("Person Identifier: " +
detectedPerson.getPerson().getIndex());
                System.out.println();
            }
        } while (personTrackingResult !=null &&
personTrackingResult.getNextToken() != null);

    }
```

Dalam fungsi main, ganti baris:

```
StartLabelDetection(bucket, video);

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

dengan:

```
StartPersonDetection(bucket, video);

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

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import
    software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import
    software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

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

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
            """;
```

```
    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

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

    startPersonLabels(rekClient, channel, bucket, video);
    getPersonDetectionResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

public static void startPersonLabels(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

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

        StartPersonTrackingRequest personTrackingRequest =
        StartPersonTrackingRequest.builder()
            .jobTag("DetectingLabels")
            .video(vidObj)
```



```
        .notificationChannel(channel)
        .build();

        StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
        startJobId = labelDetectionResponse.jobId();

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

public static void getPersonDetectionResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetPersonTrackingResponse personTrackingResult = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (personTrackingResult != null)
                paginationToken = personTrackingResult.nextToken();

            GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {

                personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                status = personTrackingResult.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
            }
        }
    }
}
```

```

        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is
null.
    VideoMetadata videoMetaData =
personTrackingResult.videoMetadata();

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

    List<PersonDetection> detectedPersons =
personTrackingResult.persons();
    for (PersonDetection detectedPerson : detectedPersons) {
        long seconds = detectedPerson.timestamp() / 1000;
        System.out.print("Sec: " + seconds + " ");
        System.out.println("Person Identifier: " +
detectedPerson.person().index());
        System.out.println();
    }

    } while (personTrackingResult != null &&
personTrackingResult.nextToken() != null);

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

```

Python

```

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

```

```
# ===== People pathing =====
def StartPersonPathing(self):
    response=self.rek.start_person_tracking(Video={'S3Object': {'Bucket':
self.bucket, 'Name': self.video}},
        NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
self.snsTopicArn})

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

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

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

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

        for personDetection in response['Persons']:
            print('Index: ' + str(personDetection['Person']['Index']))
            print('Timestamp: ' + str(personDetection['Timestamp']))
            print()

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

Dalam fungsi main, ganti baris:

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

dengan:

```
analyzer.StartPersonPathing()
if analyzer.GetSQSMessageSuccess()==True:
    analyzer.GetPersonPathingResults()
```

CLI

Jalankan AWS CLI perintah berikut untuk memulai jalur orang dalam video.

```
aws rekognition start-person-tracking --video '{"S3Object":{"Bucket":"bucket-
name","Name":"video-name"}}' \
--notification-channel '{"SNSTopicArn":"topic-ARN","RoleArn":"role-ARN"}' \
--region region-name --profile profile-name
```

Perbarui nilai berikut:

- Ubah `bucket-name` dan `video-name` ke nama bucket Amazon S3 dan nama file yang Anda tentukan pada langkah 2.
- Ubah `region-name` ke wilayah AWS yang Anda gunakan.
- Ganti nilai `profile-name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.
- Ubah `topic-ARN` ke ARN dari topik Amazon SNS yang Anda buat pada langkah 3 [Mengonfigurasi Amazon Rekognition Video](#).
- Perubah `role-ARN` ke ARN dari peran layanan IAM yang Anda buat di langkah 7 [Mengonfigurasi Amazon Rekognition Video](#).

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda alih-alih tanda kutip tunggal dan hindari tanda kutip ganda bagian dalam dengan garis miring terbalik (yaitu `\`) untuk mengatasi kesalahan parser yang mungkin Anda temui. Sebagai contoh, lihat di bawah:

```
aws rekognition start-person-tracking --video "{\\"S3Object\\":{\\"Bucket\\":
\\"bucket-name\\",\\"Name\\":\\"video-name\\"}}"
--notification-channel "{\\"SNSTopicArn\\":\\"topic-ARN\\",\\"RoleArn\\":\\"role-ARN
\\"}" \
--region region-name --profile profile-name
```

Setelah menjalankan contoh kode proses, salin yang dikembalikan `jobID` dan berikan ke `GetPersonTracking` perintah berikut di bawah ini untuk mendapatkan hasil Anda, ganti `job-id-number` dengan yang `jobID` Anda terima sebelumnya:

```
aws rekognition get-person-tracking --job-id job-id-number
```

Note

Jika Anda sudah menjalankan contoh video selain [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#), kode yang akan diganti mungkin berbeda.

3. Jalankan kode tersebut. Pengidentifikasi unik untuk orang-orang yang dilacak ditampilkan bersamaan dengan waktu, dalam hitungan detik, lintasan orang tersebut dilacak.

GetPersonTracking respon operasi

`GetPersonTracking` mengembalikan sebuah array, `Persons`, objek dari [PersonDetection](#) yang berisi detail tentang orang yang terdeteksi dalam video dan kapan lintasannya dilacak.

Anda dapat mengurutkan `Persons` dengan menggunakan parameter input `SortBy`. Tentukan `TIMESTAMP` untuk mengurutkan elemen pada saat lintasan orang dilacak dalam video. Tentukan `INDEX` untuk mengurutkan berdasarkan orang yang dilacak dalam video. Dalam setiap rangkaian hasil untuk seseorang, elemen diurutkan berdasarkan kepercayaan dalam keakuratan pelacakan lintasan. Secara default, `Persons` dikembalikan dan diurutkan berdasarkan `TIMESTAMP`. Berikut ini adalah contoh respons JSON dari `GetPersonDetection`. Hasilnya diurutkan berdasarkan waktu, dalam milidetik sejak awal video, lintasan orang yang dilacak dalam video. Dalam respons, perhatikan hal berikut:

- Informasi orang - Elemen array `PersonDetection` berisi informasi tentang orang yang terdeteksi. Misalnya, waktu orang tersebut terdeteksi (`Timestamp`), posisi orang tersebut dalam bingkai video

pada saat mereka terdeteksi (BoundingBox), dan seberapa yakin Amazon Rekognition Video bahwa orang tersebut telah terdeteksi dengan benar (Confidence).

Fitur wajah tidak dikembalikan pada setiap timestamp untuk lintasan orang yang dilacak. Selain itu, di beberapa keadaan, tubuh seseorang yang dilacak mungkin tidak terlihat, dalam hal ini hanya lokasi wajah mereka yang dikembalikan.

- Informasi halaman - Contoh tersebut menunjukkan satu halaman informasi pendeteksian orang. Anda dapat menentukan berapa banyak elemen orang yang akan dikembalikan di parameter input `MaxResults` untuk `GetPersonTracking`. Jika ada hasil yang lebih banyak dari `MaxResults`, `GetPersonTracking` mengembalikan sebuah token (`NextToken`) yang digunakan untuk mendapatkan halaman hasil berikutnya. Untuk informasi selengkapnya, lihat [Mendapatkan hasil analisis Amazon Rekognition Video](#).
- Indeks - Pengidentifikasi unik untuk mengidentifikasi orang tersebut di seluruh video.
- Informasi video - Tanggapan mencakup informasi tentang format video (`VideoMetadata`) di setiap halaman informasi yang dikembalikan oleh `GetPersonDetection`.

```
{
  "JobStatus": "SUCCEEDED",
  "NextToken": "AcDymG0fSSoaI6+BBYpka5wVlqttysSPP8VvWcuJMDluj1QpFo/vf
+mRMoqBGk8eUEiF11lR6g==",
  "Persons": [
    {
      "Person": {
        "BoundingBox": {
          "Height": 0.8787037134170532,
          "Left": 0.00572916679084301,
          "Top": 0.12129629403352737,
          "Width": 0.21666666865348816
        },
        "Face": {
          "BoundingBox": {
            "Height": 0.20000000298023224,
            "Left": 0.029999999329447746,
            "Top": 0.2199999988079071,
            "Width": 0.11249999701976776
          },
          "Confidence": 99.85971069335938,
          "Landmarks": [
            {
```

```
        "Type": "eyeLeft",
        "X": 0.06842322647571564,
        "Y": 0.3010137975215912
    },
    {
        "Type": "eyeRight",
        "X": 0.10543643683195114,
        "Y": 0.29697132110595703
    },
    {
        "Type": "nose",
        "X": 0.09569807350635529,
        "Y": 0.33701086044311523
    },
    {
        "Type": "mouthLeft",
        "X": 0.0732642263174057,
        "Y": 0.3757539987564087
    },
    {
        "Type": "mouthRight",
        "X": 0.10589495301246643,
        "Y": 0.3722417950630188
    }
],
"Pose": {
    "Pitch": -0.5589138865470886,
    "Roll": -5.1093974113464355,
    "Yaw": 18.69594955444336
},
"Quality": {
    "Brightness": 43.052337646484375,
    "Sharpness": 99.68138885498047
}
},
"Index": 0
},
"Timestamp": 0
},
{
    "Person": {
        "BoundingBox": {
            "Height": 0.9074074029922485,
            "Left": 0.24791666865348816,
```

```
    "Top": 0.09259258955717087,
    "Width": 0.375
  },
  "Face": {
    "BoundingBox": {
      "Height": 0.23000000417232513,
      "Left": 0.42500001192092896,
      "Top": 0.16333332657814026,
      "Width": 0.12937499582767487
    },
    "Confidence": 99.97504425048828,
    "Landmarks": [
      {
        "Type": "eyeLeft",
        "X": 0.46415066719055176,
        "Y": 0.2572723925113678
      },
      {
        "Type": "eyeRight",
        "X": 0.5068183541297913,
        "Y": 0.23705792427062988
      },
      {
        "Type": "nose",
        "X": 0.49765899777412415,
        "Y": 0.28383663296699524
      },
      {
        "Type": "mouthLeft",
        "X": 0.487221896648407,
        "Y": 0.3452930748462677
      },
      {
        "Type": "mouthRight",
        "X": 0.5142884850502014,
        "Y": 0.33167609572410583
      }
    ],
    "Pose": {
      "Pitch": 15.966927528381348,
      "Roll": -15.547388076782227,
      "Yaw": 11.34195613861084
    },
    "Quality": {
```



```
        "Brightness": 44.80223083496094,  
        "Sharpness": 99.95819854736328  
      }  
    },  
    "Index": 1  
  },  
  "Timestamp": 0  
}.....  
  
],  
"VideoMetadata": {  
  "Codec": "h264",  
  "DurationMillis": 67301,  
  "FileExtension": "mp4",  
  "Format": "QuickTime / MOV",  
  "FrameHeight": 1080,  
  "FrameRate": 29.970029830932617,  
  "FrameWidth": 1920  
}  
}
```

Mendeteksi alat pelindung diri

Amazon Rekognition dapat mendeteksi Alat Pelindung Diri (APD) yang dikenakan oleh orang-orang dalam citra. Anda dapat menggunakan informasi ini untuk meningkatkan praktik keselamatan di tempat kerja. Misalnya, Anda dapat menggunakan pendeteksian APD untuk membantu menentukan apakah pekerja di lokasi konstruksi mengenakan penutup kepala, atau jika petugas medis mengenakan penutup wajah dan penutup tangan. Citra berikut menunjukkan beberapa tipe APD yang dapat dideteksi.



Untuk mendeteksi APD dalam gambar, Anda memanggil [DetectProtectiveEquipment](#) API dan meneruskan gambar input. Respons adalah struktur JSON yang meliputi hal berikut ini.

- Orang-orang yang terdeteksi dalam citra.
- Bagian tubuh tempat APD dipakai (wajah, kepala, tangan kiri, dan tangan kanan).

- tipe APD yang terdeteksi pada bagian tubuh (penutup wajah, penutup tangan, dan penutup kepala).
- Untuk item APD yang terdeteksi, indikator apakah APD menutupi bagian tubuh yang sesuai atau tidak.

Kotak batas dikembalikan untuk lokasi orang dan item APD yang terdeteksi pada citra.

Atau, Anda dapat meminta ringkasan item APD dan orang yang terdeteksi dalam citra. Untuk informasi selengkapnya, lihat [Meringkas APD yang terdeteksi dalam sebuah citra](#).

Note

Deteksi APD Amazon Rekognition tidak melakukan pengenalan wajah atau perbandingan wajah, dan tidak dapat mengidentifikasi orang yang terdeteksi.

Tipe APD

[DetectProtectiveEquipment](#) mendeteksi jenis APD berikut. Jika Anda ingin mendeteksi tipe APD lainnya dalam citra, pertimbangkan untuk menggunakan label kustom Amazon Rekognition untuk melatih model kustom. Untuk informasi selengkapnya, lihat [Label Kustom Amazon Rekognition](#).

Penutup wajah

`DetectProtectiveEquipment` dapat mendeteksi penutup wajah umum seperti masker bedah, N95, dan masker yang terbuat dari kain.

Penutup tangan

`DetectProtectiveEquipment` dapat mendeteksi penutup tangan seperti sarung tangan bedah dan sarung tangan pengaman.

Penutup kepala

`DetectProtectiveEquipment` dapat mendeteksi topi dan helm keras.

API menunjukkan bahwa penutup kepala, tangan, atau wajah terdeteksi dalam citra. API tidak mengembalikan informasi tentang tipe penutup tertentu. Misalnya, 'sarung tangan bedah' untuk tipe penutup tangan.

kepercayaan deteksi APD

Amazon Rekognition membuat prediksi tentang kehadiran APD, orang, dan bagian tubuh dalam citra. API memberikan skor (50-100) yang menunjukkan tingginya kepercayaan Amazon Rekognition dalam akurasi prediksi.

Note

Jika Anda berencana untuk menggunakan operasi `DetectProtectiveEquipment` untuk membuat keputusan yang memengaruhi hak, privasi, atau akses seseorang ke layanan, maka kami sarankan supaya Anda memberikan hasilnya kepada manusia untuk ditinjau dan divalidasi sebelum mengambil tindakan.

Meringkas APD yang terdeteksi dalam sebuah citra

Anda dapat meminta ringkasan item APD dan orang yang terdeteksi dalam citra. Anda dapat menentukan daftar alat pelindung yang diperlukan (penutup wajah, penutup tangan, atau penutup kepala) dan ambang batas kepercayaan minimum (misalnya, 80%). Respons tersebut mencakup ringkasan pengenalan per citra (ID) terkonsolidasi dari orang-orang dengan APD yang diperlukan, orang tanpa APD yang diperlukan, dan orang-orang yang membuat ketentuan tidak dapat ditetapkan.

Ringkasan ini memungkinkan Anda untuk dengan cepat menjawab pertanyaan seperti Berapa banyak orang yang tidak memakai penutup wajah? atau Apakah semua orang memakai APD? Setiap orang yang terdeteksi dalam ringkasan memiliki ID yang unik. Anda dapat menggunakan informasi temukan ID seperti lokasi kotak batas seseorang yang tidak memakai APD.

Note

ID dibuat secara acak berdasarkan analisis per citra dan tidak konsisten di seluruh citra atau beberapa analisis dari citra yang sama.

Anda dapat meringkas penutup wajah, penutup kepala, penutup tangan, atau kombinasi pilihan Anda. Untuk menentukan tipe APD yang diperlukan, lihat [Menentukan persyaratan peringkasan](#). Anda juga dapat menentukan tingkat kepercayaan minimum (50-100) yang harus dipenuhi dalam setiap deteksi untuk dimasukkan dalam ringkasan.

Untuk informasi selengkapnya tentang respons ringkasan dari `DetectProtectiveEquipment`, lihat [Memahami `DetectProtectiveEquipment` tanggapannya](#).

Tutorial: Membuat AWS Lambda fungsi yang mendeteksi gambar dengan APD

Anda dapat membuat AWS Lambda fungsi yang mendeteksi alat pelindung diri (APD) dalam gambar yang terletak di ember Amazon S3. Lihat [contoh GitHub repositori AWS Documentation SDK](#) untuk tutorial Java V2 ini.

Memahami API deteksi alat pelindung diri

Informasi berikut menjelaskan [DetectProtectiveEquipment](#) API. Untuk kode sampel, lihat [Mendeteksi alat pelindung diri dalam citra](#).

Menyuplai citra

Anda dapat memberikan citra input (JPG atau format PNG), baik sebagai bit citra atau pun referensi citra yang disimpan dalam bucket Amazon S3.

Sebaiknya gunakan citra yang menunjukkan wajah orang tersebut menghadap ke kamera.

Jika citra input Anda tidak diputar ke orientasi 0 derajat, sebaiknya putar ke orientasi 0 derajat sebelum mengirimkannya ke `DetectProtectiveEquipment`. Citra dalam format JPG mungkin berisi informasi orientasi dalam metadata format (Exif) file citra yang dapat ditukar. Anda dapat menggunakan informasi ini untuk menulis kode yang memutar citra Anda. Untuk informasi selengkapnya, lihat [Exif versi 2.32](#). Citra format PNG tidak berisi informasi orientasi citra.

Untuk meneruskan gambar dari bucket Amazon S3, gunakan pengguna dengan setidaknya priviliges `ReadOnlyAccess` `AmazonS3`. Gunakan pengguna dengan `AmazonRekognitionFullAccess` priviliges untuk menelepon `DetectProtectiveEquipment`.

Pada contoh JSON input berikut, citra dilewatkan dalam bucket Amazon S3. Untuk informasi selengkapnya, lihat [Bekerja dengan citra](#). Contoh meminta ringkasan semua tipe APD (penutup kepala, penutup tangan, dan penutup wajah) dengan kepercayaan pendeteksian minimal (`MinConfidence`) dari 80%. Anda harus menentukan nilai `MinConfidence` yang berada antara 50-100% sebagai prediksi kembali `DetectProtectiveEquipment` hanya jika tingkat kepercayaan

pendeteksian berada antara 50% - 100%. Jika Anda menentukan nilai yang kurang dari 50%, hasilnya sama dengan menentukan nilai 50%. Untuk informasi selengkapnya, lihat [Menentukan persyaratan peringkasan](#).

```
{
  "Image": {
    "S3Object": {
      "Bucket": "bucket",
      "Name": "worker.jpg"
    }
  },
  "SummarizationAttributes": {
    "MinConfidence": 80,
    "RequiredEquipmentTypes": [
      "FACE_COVER",
      "HAND_COVER",
      "HEAD_COVER"
    ]
  }
}
```

Jika koleksi citra yang akan diproses cukup besar, pertimbangkan untuk menggunakan [AWS Batch](#) untuk memproses panggilan ke DetectProtectiveEquipment dalam batch di latar belakang.

Menentukan persyaratan peringkasan

Anda dapat menggunakan parameter input SummarizationAttributes ([ProtectiveEquipmentSummarizationAttributes](#)) secara opsional untuk meminta informasi ringkasan untuk jenis APD yang terdeteksi dalam gambar.

Untuk menentukan tipe APD untuk meringkas, gunakan bidang array RequiredEquipmentTypes. Dalam array, masukkan satu atau beberapa FACE_COVER, HAND_COVER atau HEAD_COVER.

Gunakan MinConfidence untuk menentukan kepercayaan pendeteksian minimum (50-100). Ringkasan tidak termasuk Orang, bagian tubuh, cakupan bagian tubuh, dan item APD, yang terdeteksi dengan kepercayaan kurang dari MinConfidence.

Untuk informasi tentang respons ringkasan dari DetectProtectiveEquipment, lihat [Memahami DetectProtectiveEquipment tanggapannya](#).

Memahami DetectProtectiveEquipment tanggapannya

DetectProtectiveEquipment mengembalikan array orang yang terdeteksi dalam citra input. Untuk setiap orang, informasi tentang bagian tubuh yang terdeteksi dan item APD yang terdeteksi dikembalikan. JSON untuk citra seorang pekerja yang mengenakan penutup kepala, penutup tangan, dan penutup wajah adalah sebagai berikut.



Dalam JSON, perhatikan hal berikut.

- Orang yang terdeteksi — `Persons` adalah array orang yang terdeteksi pada citra (termasuk orang yang tidak memakai APD). DetectProtectiveEquipment dapat mendeteksi APD pada hingga 15 orang yang terdeteksi dalam sebuah citra. Setiap [ProtectiveEquipmentPerson](#) objek dalam array berisi ID orang, kotak pembatas untuk orang tersebut, bagian tubuh yang terdeteksi, dan item APD yang terdeteksi. Nilai dari `Confidence` dalam `ProtectiveEquipmentPerson` menunjukkan persentase kepercayaan yang dimiliki Amazon Rekognition jika kotak pembatas tersebut berisi seseorang.

- **Bagian Tubuh** — `BodyParts` adalah susunan bagian tubuh ([ProtectiveEquipmentBodyPart](#)) yang terdeteksi pada seseorang (termasuk bagian tubuh yang tidak tercakup oleh APD). Setiap `ProtectiveEquipmentBodyPart` termasuk nama (`Name`) dari bagian tubuh yang terdeteksi. `DetectProtectEquipment` dapat mendeteksi bagian tubuh wajah, kepala, tangan kiri, dan tangan kanan. Bidang `Confidence` dalam `ProtectiveEquipmentBodyPart` menunjukkan persentase keyakinan yang dimiliki Amazon Rekognition dalam akurasi deteksi bagian tubuh.
- **Item PPE** — `EquipmentDetections` array dalam objek `ProtectiveEquipmentBodyPart` berisi array item APD yang terdeteksi. Setiap [EquipmentDetection](#) objek berisi bidang-bidang berikut.
 - `Type` — tipe APD yang terdeteksi.
 - `BoundingBox` — kotak pembatas di sekitar APD yang terdeteksi.
 - `Confidence` — kepercayaan yang dimiliki Amazon Rekognition bahwa kotak pembatas berisi APD yang terdeteksi.
 - `CoversBodyPart` — Menunjukkan jika APD yang terdeteksi berada di bagian tubuh yang sesuai.

[CoversBodyPart](#) Bidang `Value` adalah nilai boolean yang menunjukkan apakah APD yang terdeteksi ada di bagian tubuh yang sesuai. Bidang `Confidence` menunjukkan kepercayaan dalam prediksi. Anda dapat menggunakan `CoversBodyPart` untuk memfilter kasus yang APDnya terdeteksi di dalam citra, namun tidak benar-benar pada orang tersebut.

Note

`CoversBodyPart` tidak menunjukkan, atau menyiratkan, bahwa orang tersebut dilindungi secara memadai oleh peralatan pelindung atau bahwa peralatan pelindung itu sendiri sudah dipakai dengan benar.

- **Informasi ringkasan** — `Summary` berisi informasi ringkasan yang ditentukan dalam parameter input `SummarizationAttributes`. Untuk informasi selengkapnya, lihat [Menentukan persyaratan peringkasan](#).

`Summary` adalah objek tipe [ProtectiveEquipmentSummary](#) yang berisi informasi berikut.

- `PersonsWithRequiredEquipment` — Array ID orang yang setiap orangnya memenuhi kriteria berikut.
 - Orang tersebut mengenakan semua APD yang ditentukan dalam parameter input `SummarizationAttributes`.

- Parameter Confidence untuk orang (ProtectiveEquipmentPerson), bagian tubuh (ProtectiveEquipmentBodyPart), peralatan pelindung (EquipmentDetection) sama dengan atau lebih dari ambang batas kepercayaan minimum yang ditentukan (MinConfidence).
- Nilai CoversBodyPart untuk semua item APD adalah betul.
- PersonsWithoutRequiredEquipment — Array ID orang yang memenuhi salah satu kriteria berikut.
 - Nilai Confidence untuk orang (ProtectiveEquipmentPerson), bagian tubuh (ProtectiveEquipmentBodyPart), dan cakupan bagian tubuh (CoversBodyPart) lebih dari ambang batas minimum yang ditentukan (MinConfidence), namun orang tersebut tidak memakai satu atau beberapa APD yang ditentukan (SummarizationAttributes).
 - Nilai dari CoversBodyPart adalah salah untuk APD tertentu (SummarizationAttributes) yang memiliki nilai Confidence lebih dari ambang batas kepercayaan minimum yang ditentukan (MinConfidence). Orang tersebut juga memiliki semua APD yang ditentukan (SummarizationAttributes) dan nilai Confidence untuk orang (ProtectiveEquipmentPerson), bagian tubuh (ProtectiveEquipmentBodyPart), dan peralatan pelindung (EquipmentDetection) lebih dari atau sama dengan ambang batas kepercayaan minimum (MinConfidence).
- PersonsIndeterminate — Array ID orang terdeteksi yang nilai Confidence untuk orang (ProtectiveEquipmentPerson), bagian tubuh (ProtectiveEquipmentBodyPart), alat pelindung (EquipmentDetection), atau boolean CoversBodyPart kurang dari ambang batas minimum yang ditentukan (MinConfidence).

Gunakan ukuran array untuk mendapatkan hitungan untuk ringkasan tertentu. Misalnya, ukuran PersonsWithRequiredEquipment memberi tahu Anda jumlah orang yang terdeteksi memakai tipe APD tertentu.

Anda dapat menggunakan ID orang untuk mengetahui informasi lebih lanjut tentang seseorang, seperti lokasi kotak pembatas orang tersebut. ID orang memetakan ke bidang ID ProtectiveEquipmentPerson) objek yang dikembalikan dalam Persons (array ProtectiveEquipmentPerson). Anda kemudian bisa mendapatkan kotak pembatas dan informasi lainnya dari objek ProtectiveEquipmentPerson yang sesuai.

```
{
```



```
{
  "Name": "RIGHT_HAND",
  "Confidence": 99.82939147949219,
  "EquipmentDetections": [
    {
      "BoundingBox": {
        "Width": 0.20971858501434326,
        "Height": 0.20528452098369598,
        "Left": 0.2711356580257416,
        "Top": 0.6750612258911133
      },
      "Confidence": 95.70789337158203,
      "Type": "HAND_COVER",
      "CoversBodyPart": {
        "Confidence": 99.85433197021484,
        "Value": true
      }
    }
  ],
},
{
  "Name": "HEAD",
  "Confidence": 99.9999008178711,
  "EquipmentDetections": [
    {
      "BoundingBox": {
        "Width": 0.24350935220718384,
        "Height": 0.34623199701309204,
        "Left": 0.43011072278022766,
        "Top": 0.01103297434747219
      },
      "Confidence": 83.88762664794922,
      "Type": "HEAD_COVER",
      "CoversBodyPart": {
        "Confidence": 99.96485900878906,
        "Value": true
      }
    }
  ]
},
{
  "BoundingBox": {
    "Width": 0.7403100728988647,
    "Height": 0.9412225484848022,
```

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

Mendeteksi alat pelindung diri dalam citra

Untuk mendeteksi Alat Pelindung Diri (APD) pada orang dalam gambar, gunakan operasi API [DetectProtectiveEquipment](#) non-penyimpanan.

Anda dapat memberikan citra input sebagai array bit citra (bit citra yang dikodekan base64) atau sebagai objek Amazon S3, dengan menggunakan AWS SDK atau AWS Command Line Interface (AWS CLI). Contoh-contoh ini menggunakan citra yang tersimpan di bucket Amazon S3. Untuk informasi selengkapnya, lihat [Bekerja dengan citra](#).

Untuk mendeteksi APD pada orang dalam citra

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan `AmazonRekognitionFullAccess` dan `AmazonS3ReadOnlyAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Unggah citra (yang berisi satu atau beberapa orang yang memakai APD) ke bucket S3 Anda.

Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

- Gunakan contoh berikut untuk memanggil operasi DetectProtectiveEquipment. Untuk informasi tentang menampilkan kotak pembatas di citra, lihat [Menampilkan kotak pembatas](#).

Java

Contoh ini menampilkan informasi tentang item APD yang terdeteksi pada orang yang terdeteksi dalam citra.

Ubah nilai bucket menjadi nama bucket Amazon S3 yang berisi citra Anda. Ubah nilai photo ke nama file citra Anda.

```
//Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package com.amazonaws.samples;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.ProtectiveEquipmentBodyPart;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.ProtectiveEquipmentPerson;
import
    com.amazonaws.services.rekognition.model.ProtectiveEquipmentSummarizationAttributes;

import java.util.List;
import com.amazonaws.services.rekognition.model.BoundingBox;
import
    com.amazonaws.services.rekognition.model.DetectProtectiveEquipmentRequest;
import com.amazonaws.services.rekognition.model.DetectProtectiveEquipmentResult;
import com.amazonaws.services.rekognition.model.EquipmentDetection;

public class DetectPPE {

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

        String photo = "photo";
        String bucket = "bucket";
```

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

ProtectiveEquipmentSummarizationAttributes summaryAttributes = new
ProtectiveEquipmentSummarizationAttributes()
    .withMinConfidence(80F)
    .withRequiredEquipmentTypes("FACE_COVER", "HAND_COVER",
"HEAD_COVER");

DetectProtectiveEquipmentRequest request = new
DetectProtectiveEquipmentRequest()
    .withImage(new Image()
        .withS3Object(new S3Object()
            .withName(photo).withBucket(bucket)))
    .withSummarizationAttributes(summaryAttributes);

try {
    System.out.println("Detected PPE for people in image " + photo);
    System.out.println("Detected people\n-----");
    DetectProtectiveEquipmentResult result =
rekognitionClient.detectProtectiveEquipment(request);

    List <ProtectiveEquipmentPerson> persons = result.getPersons();

    for (ProtectiveEquipmentPerson person: persons) {
        System.out.println("ID: " + person.getId());
        List<ProtectiveEquipmentBodyPart>
bodyParts=person.getBodyParts();
        if (bodyParts.isEmpty()){
            System.out.println("\tNo body parts detected");
        } else
            for (ProtectiveEquipmentBodyPart bodyPart: bodyParts) {
                System.out.println("\t" + bodyPart.getName() + ".
Confidence: " + bodyPart.getConfidence().toString());

                List<EquipmentDetection>
equipmentDetections=bodyPart.getEquipmentDetections();

                if (equipmentDetections.isEmpty()){
```

```

        System.out.println("\t\tNo PPE Detected on " +
bodyPart.getName());
    }
    else {
        for (EquipmentDetection item: equipmentDetections) {
            System.out.println("\t\tItem: " + item.getType()
+ ". Confidence: " + item.getConfidence().toString());
            System.out.println("\t\tCovers body part: "
+
item.getCoversBodyPart().getValue().toString() + ". Confidence: " +
item.getCoversBodyPart().getConfidence().toString());

            System.out.println("\t\tBounding Box");
            BoundingBox box =item.getBoundingBox();

            System.out.println("\t\tLeft: "
+box.getLeft().toString());
            System.out.println("\t\tTop: " +
box.getTop().toString());
            System.out.println("\t\tWidth: " +
box.getWidth().toString());
            System.out.println("\t\tHeight: " +
box.getHeight().toString());
            System.out.println("\t\tConfidence: " +
item.getConfidence().toString());
            System.out.println();
        }
    }
}

System.out.println("Person ID Summary\n-----");

//List<Integer> list=;
DisplaySummary("With required equipment",
result.getSummary().getPersonsWithRequiredEquipment());
DisplaySummary("Without required equipment",
result.getSummary().getPersonsWithoutRequiredEquipment());
DisplaySummary("Indeterminate",
result.getSummary().getPersonsIndeterminate());

} catch(AmazonRekognitionException e) {

```

```
        e.printStackTrace();
    }
}
static void DisplaySummary(String summaryType,List<Integer> idList)
{
    System.out.print(summaryType + "\n\tIDs  ");
    if (idList.size()==0) {
        System.out.println("None");
    }
    else {
        int count=0;
        for (Integer id: idList ) {
            if (count++ == idList.size()-1) {
                System.out.println(id.toString());
            }
            else {
                System.out.print(id.toString() + ", ");
            }
        }
    }

    System.out.println();

}
}
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
//snippet-start:[rekognition.java2.detect_ppe.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import
    software.amazon.awssdk.services.rekognition.model.DetectProtectiveEquipmentRequest;
import
    software.amazon.awssdk.services.rekognition.model.DetectProtectiveEquipmentResponse;
```



```
import software.amazon.awssdk.services.rekognition.model.EquipmentDetection;
import
    software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentBodyPart;
import
    software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentSummarizationAttri
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
    software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentPerson;
import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.List;
//snippet-end:[rekognition.java2.detect_ppe.import]

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

    public static void main(String[] args) {

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

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

```
String sourceImage = args[0];
String bucketName = args[1];
Region region = Region.US_WEST_2;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
    .build();

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

displayGear(s3, rekClient, sourceImage, bucketName) ;
s3.close();
rekClient.close();
System.out.println("This example is done!");
}

// snippet-start:[rekognition.java2.detect_ppe.main]
public static void displayGear(S3Client s3,
                               RekognitionClient rekClient,
                               String sourceImage,
                               String bucketName) {

    byte[] data = getObjectBytes (s3, bucketName, sourceImage);
    InputStream is = new ByteArrayInputStream(data);

    try {
        ProtectiveEquipmentSummarizationAttributes summarizationAttributes =
        ProtectiveEquipmentSummarizationAttributes.builder()
            .minConfidence(80F)
            .requiredEquipmentTypesWithStrings("FACE_COVER", "HAND_COVER",
            "HEAD_COVER")
            .build();

        SdkBytes sourceBytes = SdkBytes.fromInputStream(is);
        software.amazon.awssdk.services.rekognition.model.Image souImage =
        Image.builder()
            .bytes(sourceBytes)
            .build();
    }
}
```

```
        DetectProtectiveEquipmentRequest request =
DetectProtectiveEquipmentRequest.builder()
        .image(souImage)
        .summarizationAttributes(summarizationAttributes)
        .build();

        DetectProtectiveEquipmentResponse result =
rekClient.detectProtectiveEquipment(request);
        List<ProtectiveEquipmentPerson> persons = result.persons();
        for (ProtectiveEquipmentPerson person: persons) {
            System.out.println("ID: " + person.id());
            List<ProtectiveEquipmentBodyPart> bodyParts=person.bodyParts();
            if (bodyParts.isEmpty()){
                System.out.println("\tNo body parts detected");
            } else
                for (ProtectiveEquipmentBodyPart bodyPart: bodyParts) {
                    System.out.println("\t" + bodyPart.name() + ". Confidence:
" + bodyPart.confidence().toString());
                    List<EquipmentDetection>
equipmentDetections=bodyPart.equipmentDetections();

                    if (equipmentDetections.isEmpty()){
                        System.out.println("\t\tNo PPE Detected on " +
bodyPart.name());
                    } else {
                        for (EquipmentDetection item: equipmentDetections) {
                            System.out.println("\t\tItem: " + item.type() + ".
Confidence: " + item.confidence().toString());
                            System.out.println("\t\tCovers body part: "
                                + item.coversBodyPart().value().toString()
+ ". Confidence: " + item.coversBodyPart().confidence().toString());

                                System.out.println("\t\tBounding Box");
                                BoundingBox box =item.boundingBox();
                                System.out.println("\t\tLeft: "
+box.left().toString());
                                System.out.println("\t\tTop: " +
box.top().toString());
                                System.out.println("\t\tWidth: " +
box.width().toString());
                                System.out.println("\t\tHeight: " +
box.height().toString());
                                System.out.println("\t\tConfidence: " +
item.confidence().toString());
```

```
                System.out.println();
            }
        }
    }
    System.out.println("Person ID Summary\n-----");

    displaySummary("With required equipment",
result.summary().personsWithRequiredEquipment());
    displaySummary("Without required equipment",
result.summary().personsWithoutRequiredEquipment());
    displaySummary("Indeterminate",
result.summary().personsIndeterminate());

    } catch (RekognitionException e) {
        e.printStackTrace();
        System.exit(1);
    }
}

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

    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

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

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

static void displaySummary(String summaryType, List<Integer> idList) {
    System.out.print(summaryType + "\n\tIDs ");
    if (idList.size()==0) {
```

```
        System.out.println("None");
    } else {
        int count=0;
        for (Integer id: idList ) {
            if (count++ == idList.size()-1) {
                System.out.println(id.toString());
            } else {
                System.out.print(id.toString() + ", ");
            }
        }
        System.out.println();
    }
}
// snippet-end:[rekognition.java2.detect_ppe.main]
}
```

AWS CLI

AWS CLI Perintah ini meminta ringkasan APD dan menampilkan output JSON untuk operasi CLI `detect-protective-equipment`.

Ubah `bucketname` dengan nama bucket Amazon S3 yang berisi citra. Ubah `input.jpg` menjadi nama citra yang ingin Anda gunakan.

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

AWS CLI Perintah ini menampilkan output JSON untuk operasi `detect-protective-equipment` CLI.

Ubah `bucketname` dengan nama bucket Amazon S3 yang berisi citra. Ubah `input.jpg` menjadi nama citra yang ingin Anda gunakan.

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

Python

Contoh ini menampilkan informasi tentang item APD yang terdeteksi pada orang yang terdeteksi dalam citra.

Ubah nilai bucket menjadi nama bucket Amazon S3 yang berisi citra Anda. Ubah nilai photo ke nama file citra Anda. Ganti nilai profile_name di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
# Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def detect_ppe(photo, bucket):

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

    response = client.detect_protective_equipment(Image={'S3Object': {'Bucket':
bucket, 'Name': photo}},

SummarizationAttributes={'MinConfidence': 80,

'RequiredEquipmentTypes': ['FACE_COVER',

                            'HAND_COVER',

                            'HEAD_COVER']})

    print('Detected PPE for people in image ' + photo)
    print('\nDetected people\n-----')
    for person in response['Persons']:

        print('Person ID: ' + str(person['Id']))
        print('Body Parts\n-----')
        body_parts = person['BodyParts']
        if len(body_parts) == 0:
            print('No body parts found')
        else:
            for body_part in body_parts:
```

```

        print('\t' + body_part['Name'] + '\n\t\tConfidence: ' +
str(body_part['Confidence']))
        print('\n\t\tDetected PPE\n\t\t-----')
        ppe_items = body_part['EquipmentDetections']
        if len(ppe_items) == 0:
            print('\t\tNo PPE detected on ' + body_part['Name'])
        else:
            for ppe_item in ppe_items:
                print('\t\t' + ppe_item['Type'] + '\n\t\t\tConfidence: '
+ str(ppe_item['Confidence']))
                print('\t\tCovers body part: ' + str(
                    ppe_item['CoversBodyPart']['Value']) + '\n\t\t\t
\tConfidence: ' + str(
                    ppe_item['CoversBodyPart']['Confidence']))
                print('\t\tBounding Box:')
                print('\t\t\tTop: ' + str(ppe_item['BoundingBox']
['Top']))
                print('\t\t\tLeft: ' + str(ppe_item['BoundingBox']
['Left']))
                print('\t\t\tWidth: ' + str(ppe_item['BoundingBox']
['Width']))
                print('\t\t\tHeight: ' + str(ppe_item['BoundingBox']
['Height']))
                print('\t\t\tConfidence: ' +
str(ppe_item['Confidence']))
            print()
            print()

        print('Person ID Summary\n-----')
        display_summary('With required equipment', response['Summary']
['PersonsWithRequiredEquipment'])
        display_summary('Without required equipment', response['Summary']
['PersonsWithoutRequiredEquipment'])
        display_summary('Indeterminate', response['Summary']
['PersonsIndeterminate'])

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

# Display summary information for supplied summary.
def display_summary(summary_type, summary):
    print(summary_type + '\n\tIDs: ', end='')
    if (len(summary) == 0):
        print('None')

```

```
else:
    for num, id in enumerate(summary, start=0):
        if num == len(summary) - 1:
            print(id)
        else:
            print(str(id) + ', ', end='')

def main():
    photo = 'photo-name'
    bucket = 'bucket-name'
    person_count = detect_ppe(photo, bucket)
    print("Persons detected: " + str(person_count))

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

Contoh: Menggambar kotak pembatas di sekitar penutup wajah

Contoh berikut menunjukkan cara menggambar kotak pembatas di sekitar penutup wajah yang terdeteksi pada orang. Untuk contoh yang menggunakan AWS Lambda dan Amazon DynamoDB, lihat AWS repositori contoh SDK [Documentation](#). GitHub

Untuk mendeteksi penutup wajah, Anda menggunakan operasi API [DetectProtectiveEquipment](#) non-penyimpanan. Citra dimuat dari sistem file lokal. Anda memberikan citra input ke `DetectProtectiveEquipment` sebagai array bit citra (bit citra yang dikodekan base64). Untuk informasi selengkapnya, lihat [Bekerja dengan citra](#).

Contoh ini menampilkan kotak pembatas di sekitar penutup wajah yang terdeteksi. Kotak pembatas berwarna hijau jika penutup wajah menutupi bagian tubuh sepenuhnya. Jika tidak, kotak pembatas merah ditampilkan. Sebagai peringatan, kotak pembatas kuning ditampilkan dalam kotak pembatas penutup wajah, jika kepercayaan pendeteksian kurang dari nilai kepercayaan yang ditentukan. Jika penutup wajah tidak terdeteksi, maka kotak pembatas merah dicitra di sekitar orang tersebut.

Output citra tersebut mirip dengan hal berikut.



Untuk menampilkan kotak pembatas pada penutup wajah yang terdeteksi

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan AmazonRekognitionFullAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut untuk memanggil operasi DetectProtectiveEquipment. Untuk informasi tentang menampilkan kotak pembatas di citra, lihat [Menampilkan kotak pembatas](#).

Java

Dalam fungsi main, ubah hal berikut:

- Nilai photo ke jalur dan nama file citra lokal (PNG atau JPEG).

- Nilai confidence sampai dengan tingkat kepercayaan yang diinginkan (50-100).

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

import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import com.amazonaws.util.IOUtils;

import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.BoundingBox;
import
    com.amazonaws.services.rekognition.model.DetectProtectiveEquipmentRequest;
import com.amazonaws.services.rekognition.model.DetectProtectiveEquipmentResult;
import com.amazonaws.services.rekognition.model.EquipmentDetection;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.ProtectiveEquipmentBodyPart;
import com.amazonaws.services.rekognition.model.ProtectiveEquipmentPerson;

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

    private static final long serialVersionUID = 1L;

    BufferedImage image;
    static int scale;
    DetectProtectiveEquipmentResult result;
```

```
float confidence=80;

public PPEBoundingBox(DetectProtectiveEquipmentResult ppeResult,
BufferedImage bufImage, float requiredConfidence) throws Exception {
    super();
    scale = 2; // increase to shrink image size.

    result = ppeResult;
    image = bufImage;

    confidence=requiredConfidence;
}
// Draws the bounding box around the detected faces.
public void paintComponent(Graphics g) {
    float left = 0;
    float top = 0;
    int height = image.getHeight(this);
    int width = image.getWidth(this);
    int offset=20;

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

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

    // Iterate through detected persons and display bounding boxes.
    List<ProtectiveEquipmentPerson> persons = result.getPersons();

    for (ProtectiveEquipmentPerson person: persons) {
        BoundingBox boxPerson = person.getBoundingBox();
        left = width * boxPerson.getLeft();
        top = height * boxPerson.getTop();
        Boolean foundMask=false;

        List<ProtectiveEquipmentBodyPart> bodyParts=person.getBodyParts();

        if (bodyParts.isEmpty()==false)
        {
            //body parts detected

            for (ProtectiveEquipmentBodyPart bodyPart: bodyParts) {
```

```

        List<EquipmentDetection>
equipmentDetections=bodyPart.getEquipmentDetections();

        for (EquipmentDetection item: equipmentDetections) {

            if (item.getType().contentEquals("FACE_COVER"))
            {
                // Draw green or red bounding box depending on
mask coverage.

                foundMask=true;
                BoundingBox box =item.getBoundingBox();
                left = width * box.getLeft();
                top = height * box.getTop();
                Color maskColor=new Color( 0, 212, 0);

                if (item.getCoversBodyPart().getValue()==false)
            {

                    // red bounding box
                    maskColor=new Color( 255, 0, 0);
                }
                g2d.setColor(maskColor);
                g2d.drawRect(Math.round(left / scale),
Math.round(top / scale),
                                Math.round((width * box.getWidth()) /
scale), Math.round((height * box.getHeight())) / scale);

                // Check confidence is > supplied confidence.
                if (item.getCoversBodyPart().getConfidence(<
confidence)

                {
                    // Draw a yellow bounding box inside face
mask bounding box

                    maskColor=new Color( 255, 255, 0);
                    g2d.setColor(maskColor);
                    g2d.drawRect(Math.round((left + offset) /
scale),
                                Math.round((top + offset) / scale),
                                Math.round((width *
box.getWidth())- (offset * 2 ))/ scale,
                                Math.round((height *
box.getHeight()) -( offset* 2)) / scale);
                }
            }
        }
    }

```

```
        }

    }

}

// Didn't find a mask, so draw person bounding box red
if (foundMask==false) {

    left = width * boxPerson.getLeft();
    top = height * boxPerson.getTop();
    g2d.setColor(new Color(255, 0, 0));
    g2d.drawRect(Math.round(left / scale), Math.round(top / scale),
        Math.round(((width) * boxPerson.getWidth()) / scale),
Math.round((height * boxPerson.getHeight())) / scale);
    }
}

}

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

    String photo = "photo";

    float confidence =80;

    int height = 0;
    int width = 0;

    BufferedImage image = null;
    ByteBuffer imageBytes;

    // Get image bytes for call to DetectProtectiveEquipment
    try (InputStream inputStream = new FileInputStream(new File(photo))) {
        imageBytes = ByteBuffer.wrap(IUtils.toByteArray(inputStream));
    }

    //Get image for display
    InputStream imageBytesStream;
    imageBytesStream = new ByteArrayInputStream(imageBytes.array());

    ByteArrayOutputStream baos = new ByteArrayOutputStream();
```

```
        image=ImageIO.read(imageBytesStream);
        ImageIO.write(image, "jpg", baos);
        width = image.getWidth();
        height = image.getHeight();

        //Get Rekognition client
        AmazonRekognition rekognitionClient =
AmazonRekognitionClientBuilder.defaultClient();

        // Call DetectProtectiveEquipment
        DetectProtectiveEquipmentRequest request = new
DetectProtectiveEquipmentRequest()
                .withImage(new Image()
                        .withBytes(imageBytes));

        DetectProtectiveEquipmentResult result =
rekognitionClient.detectProtectiveEquipment(request);

        // Create frame and panel.
        JFrame frame = new JFrame("Detect PPE");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        PPEBoundingBox panel = new PPEBoundingBox(result, image, confidence);
        panel.setPreferredSize(new Dimension(image.getWidth() / scale,
image.getHeight() / scale));
        frame.setContentPane(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.*;
import java.util.List;
```

```
import javax.imageio.ImageIO;
import javax.swing.*;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import
    software.amazon.awssdk.services.rekognition.model.DetectProtectiveEquipmentRequest;
import software.amazon.awssdk.services.rekognition.model.EquipmentDetection;
import
    software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentBodyPart;
import
    software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentPerson;
import
    software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentSummarizationAttri
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
    software.amazon.awssdk.services.rekognition.model.DetectProtectiveEquipmentResponse;
//snippet-end:[rekognition.java2.display_mask.import]

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

    DetectProtectiveEquipmentResponse result;
    static BufferedImage image;
    static int scale;
    float confidence;

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

```
final String usage = "\n" +
    "Usage: " +
    "  <sourceImage> <bucketName>\n\n" +
    "Where:\n" +
    "  sourceImage - The name of the image in an Amazon S3 bucket that
shows a person wearing a mask (for example, masks.png). \n\n" +
    "  bucketName - The name of the Amazon S3 bucket (for example,
myBucket). \n\n";

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

String sourceImage = args[0];
String bucketName = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
    .build();

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

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

// snippet-start:[rekognition.java2.display_mask.main]
public static void displayGear(S3Client s3,
                              RekognitionClient rekClient,
                              String sourceImage,
                              String bucketName) {

    float confidence = 80;
    byte[] data = getObjectBytes(s3, bucketName, sourceImage);
    InputStream is = new ByteArrayInputStream(data);

    try {
```



```
        ProtectiveEquipmentSummarizationAttributes summarizationAttributes =
ProtectiveEquipmentSummarizationAttributes.builder()
        .minConfidence(70F)
        .requiredEquipmentTypesWithStrings("FACE_COVER")
        .build();

        SdkBytes sourceBytes = SdkBytes.fromInputStream(is);
        image = ImageIO.read(sourceBytes.asInputStream());

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

        DetectProtectiveEquipmentRequest request =
DetectProtectiveEquipmentRequest.builder()
        .image(souImage)
        .summarizationAttributes(summarizationAttributes)
        .build();

        DetectProtectiveEquipmentResponse result =
rekClient.detectProtectiveEquipment(request);
        JFrame frame = new JFrame("Detect PPE");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        PPEBoundingBoxFrame panel = new PPEBoundingBoxFrame(result, image,
confidence);
        panel.setPreferredSize(new Dimension(image.getWidth() / scale,
image.getHeight() / scale));
        frame.setContentPane(panel);
        frame.pack();
        frame.setVisible(true);

    } catch (RekognitionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

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

```
try {
    GetObjectRequest objectRequest = GetObjectRequest
        .builder()
        .key(keyName)
        .bucket(bucketName)
        .build();

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

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

public PPEBoundingBoxFrame(DetectProtectiveEquipmentResponse ppeResult,
BufferedImage bufImage, float requiredConfidence) {
    super();
    scale = 1; // increase to shrink image size.
    result = ppeResult;
    image = bufImage;
    confidence=requiredConfidence;
}

// Draws the bounding box around the detected masks.
public void paintComponent(Graphics g) {
    float left = 0;
    float top = 0;
    int height = image.getHeight(this);
    int width = image.getWidth(this);
    int offset=20;

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

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

    // Iterate through detected persons and display bounding boxes.
    List<ProtectiveEquipmentPerson> persons = result.persons();
    for (ProtectiveEquipmentPerson person: persons) {
```

```

List<ProtectiveEquipmentBodyPart> bodyParts=person.bodyParts();
if (!bodyParts.isEmpty()){
    for (ProtectiveEquipmentBodyPart bodyPart: bodyParts) {
        List<EquipmentDetection>
equipmentDetections=bodyPart.equipmentDetections();
        for (EquipmentDetection item: equipmentDetections) {

            String myType = item.type().toString();
            if (myType.compareTo("FACE_COVER") ==0) {

                // Draw green bounding box depending on mask coverage.
                BoundingBox box =item.boundingBox();
                left = width * box.left();
                top = height * box.top();
                Color maskColor=new Color( 0, 212, 0);

                if (item.coversBodyPart().equals(false)) {
                    // red bounding box.
                    maskColor=new Color( 255, 0, 0);
                }
                g2d.setColor(maskColor);
                g2d.drawRect(Math.round(left / scale), Math.round(top /
scale),
                    Math.round((width * box.width()) / scale),
Math.round((height * box.height())) / scale);

                // Check confidence is > supplied confidence.
                if (item.coversBodyPart().confidence() < confidence) {
                    // Draw a yellow bounding box inside face mask
                    bounding box.

                    maskColor=new Color( 255, 255, 0);
                    g2d.setColor(maskColor);
                    g2d.drawRect(Math.round((left + offset) / scale),
                        Math.round((top + offset) / scale),
                        Math.round((width * box.width())- (offset *
2 ))/ scale,
                            Math.round((height * box.height()) -
( offset* 2)) / scale);
                }
            }
        }
    }
}

```

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

Python

Dalam fungsi main, ubah hal berikut:

- Nilai photo ke jalur dan nama file citra lokal (PNG atau JPEG).
- Nilai confidence sampai dengan tingkat kepercayaan yang diinginkan (50-100).
- Ganti nilai profile_name di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
#Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/  
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
import boto3  
import io  
from PIL import Image, ImageDraw, ExifTags, ImageColor  
  
def detect_ppe(photo, confidence):  
  
    fill_green='#00d400'  
    fill_red='#ff0000'  
    fill_yellow='#ffff00'  
    line_width=3  
  
    #open image and get image data from stream.  
    image = Image.open(open(photo, 'rb'))  
    stream = io.BytesIO()  
    image.save(stream, format=image.format)  
    image_binary = stream.getvalue()  
    imgWidth, imgHeight = image.size  
    draw = ImageDraw.Draw(image)  
  
    client=boto3.client('rekognition')  
  
    response = client.detect_protective_equipment(Image={'Bytes': image_binary})
```

```

for person in response['Persons']:

    found_mask=False

    for body_part in person['BodyParts']:
        ppe_items = body_part['EquipmentDetections']

        for ppe_item in ppe_items:
            #found a mask
            if ppe_item['Type'] == 'FACE_COVER':
                fill_color=fill_green
                found_mask=True
                # check if mask covers face
                if ppe_item['CoversBodyPart']['Value'] == False:
                    fill_color=fill='#ff0000'
                # draw bounding box around mask
                box = ppe_item['BoundingBox']
                left = imgWidth * box['Left']
                top = imgHeight * box['Top']
                width = imgWidth * box['Width']
                height = imgHeight * box['Height']
                points = (
                    (left,top),
                    (left + width, top),
                    (left + width, top + height),
                    (left , top + height),
                    (left, top)
                )
                draw.line(points, fill=fill_color, width=line_width)

                # Check if confidence is lower than supplied value
                if ppe_item['CoversBodyPart']['Confidence'] < confidence:
                    #draw warning yellow bounding box within face mask
                    bounding box

                    offset=line_width+ line_width
                    points = (
                        (left+offset,top + offset),
                        (left + width-offset, top+offset),
                        ((left) + (width-offset), (top-offset) +
                    (height)),
                        (left+ offset , (top) + (height -offset)),
                        (left + offset, top + offset)
                    )
                    draw.line(points, fill=fill_yellow, width=line_width)

```

```
    if found_mask==False:
        # no face mask found so draw red bounding box around body
        box = person['BoundingBox']
        left = imgWidth * box['Left']
        top = imgHeight * box['Top']
        width = imgWidth * box['Width']
        height = imgHeight * box['Height']
        points = (
            (left,top),
            (left + width, top),
            (left + width, top + height),
            (left , top + height),
            (left, top)
        )
        draw.line(points, fill=fill_red, width=line_width)

    image.show()

def main():
    photo='photo'
    confidence=80
    detect_ppe(photo, confidence)

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

CLI

Dalam contoh CLI berikut mengubah nilai argumen yang tercantum di bawah ini:

- Nilai `photo` ke jalur dan nama file citra lokal (PNG atau JPEG).
- Nilai `confidence` sampai dengan tingkat kepercayaan yang diinginkan (50-100).
- Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
aws rekognition detect-protective-equipment
--image '{"S3Object":{"Bucket":"bucket-name","Name":"image-name"}}' --profile
profile-name \
--summarization-attributes
'{"MinConfidence":MinConfidenceNumber,"RequiredEquipmentTypes":["FACE_COVER"]}'
```

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda alih-alih tanda kutip tunggal dan hindari tanda kutip ganda bagian dalam dengan garis miring terbalik (yaitu \) untuk mengatasi kesalahan parser yang mungkin Anda temui. Sebagai contoh, lihat berikut ini:

```
aws rekognition detect-protective-equipment --
image '{"S3Object\":{"Bucket\":"bucket-name\","Name\":"image-name\}}' \
--profile profile-name --summarization-
attributes '{"MinConfidence\":MinConfidenceNumber,\"RequiredEquipmentTypes\":
[\"FACE_COVER\"]}'
```

Mengenali selebriti

Amazon Rekognition memudahkan pelanggan untuk secara otomatis mengenali puluhan ribu tokoh terkenal dalam gambar dan video menggunakan pembelajaran mesin. Metadata yang disediakan oleh API pengenalan selebriti secara signifikan mengurangi upaya manual berulang yang diperlukan untuk menandai konten dan membuatnya mudah dicari.

Proliferasi konten gambar dan video yang cepat berarti bahwa perusahaan media sering berjuang untuk mengatur, mencari, dan memanfaatkan katalog media mereka dalam skala besar. Saluran berita dan penyiar olahraga sering kali perlu menemukan gambar dan video dengan cepat, untuk menanggapi peristiwa terkini dan membuat program yang relevan. Metadata yang tidak mencukupi membuat tugas-tugas ini sulit, tetapi dengan Amazon Rekognition Anda dapat secara otomatis menandai volume besar konten baru atau arsip untuk membuatnya mudah dicari untuk set komprehensif selebriti internasional yang dikenal luas seperti aktor, olahragawan, dan pembuat konten online.

Pengakuan selebriti Amazon Rekognition dirancang untuk digunakan secara eksklusif dalam kasus di mana Anda berharap mungkin ada selebriti yang dikenal dalam gambar atau video. Untuk informasi tentang pengenalan wajah yang bukan selebriti, lihat [Mencari wajah dalam koleksi](#).

Note

Jika Anda seorang selebriti dan tidak ingin disertakan dalam fitur ini, hubungi [AWS Support](#) atau email <rekognition-celebrity-opt-out@amazon.com>.

Topik

- [Pengenalan selebriti dibandingkan dengan pencarian wajah](#)
- [Mengenali selebriti dalam sebuah citra](#)
- [Mengenali selebriti dalam video yang tersimpan](#)
- [Mendapatkan informasi tentang selebriti](#)

Pengenalan selebriti dibandingkan dengan pencarian wajah

Amazon Rekognition menawarkan pengenalan selebriti dan fungsionalitas pengenalan wajah. Fungsionalitas ini memiliki beberapa kunci perbedaan dalam kasus penggunaan dan praktik terbaik.

Pengenalan selebriti telah dilatih dengan kemampuan untuk mengenali ratusan ribu orang populer di berbagai macam bidang seperti olahraga, media, politik, dan bisnis. Fungsionalitas ini dirancang untuk membantu Anda mencari citra atau video dalam volume besar untuk mengidentifikasi satu set kecil yang mungkin terdapat selebriti. Ini tidak dimaksudkan untuk digunakan untuk mencocokkan wajah antara orang yang berbeda yang bukan selebriti. Di situasi ketika keakuratan pencocokan selebriti bersifat penting, kami juga merekomendasikan penggunaan operator manusia untuk melihat sekilas sedikit konten yang ditandai untuk memastikan tingkat akurasi yang tinggi dan penerapan yang tepat dari penilaian manual. Pengenalan selebriti tidak boleh digunakan dengan cara yang dapat mengakibatkan dampak negatif pada kebebasan sipil.

Sebaliknya, pengenalan wajah adalah fungsi yang lebih umum yang memungkinkan Anda membuat koleksi wajah Anda sendiri dengan vektor wajah Anda sendiri untuk memverifikasi identitas atau mencari siapa pun, bukan hanya selebriti. Pengenalan wajah dapat digunakan untuk aplikasi seperti mengautentikasikan akses menuju bangunan, keselamatan publik, dan media sosial. Dalam semua kasus ini, direkomendasikan agar Anda menggunakan praktik terbaik, batasan kepercayaan yang sesuai (termasuk 99% untuk kasus penggunaan keselamatan publik), dan peninjauan manusia dalam situasi ketika keakuratan dari pencocokan itu penting.

Untuk informasi selengkapnya, lihat [Mencari wajah dalam koleksi](#).

Mengenali selebriti dalam sebuah citra

Untuk mengenali selebriti dalam citra dan mendapatkan informasi tambahan tentang selebriti yang dikenali, gunakan operasi API non-penyimpanan [RecognizeCelebrities](#). Misalnya, di media sosial atau industri berita dan hiburan ketika pengumpulan informasi bisa menjadi prioritas yang tinggi, Anda dapat menggunakan operasi `RecognizeCelebrities` untuk mengidentifikasi sebanyak 64 selebriti dalam sebuah citra, dan mengembalikan tautan ke halaman web selebriti, jika tersedia. Amazon Rekognition tidak mengingat citra mana yang mendeteksi selebriti. Aplikasi Anda harus menyimpan informasi ini.

Jika Anda belum menyimpan informasi tambahan untuk selebriti yang dikirimkan oleh `RecognizeCelebrities` dan Anda ingin menghindari penganalisaan ulang pada sebuah citra untuk mendapatkannya, gunakan [GetCelebrityInfo](#). Untuk memanggil `GetCelebrityInfo`, Anda perlu pengenal unik yang ditetapkan Amazon Rekognition untuk setiap selebriti. Pengenal dikirimkan sebagai bagian dari respons `RecognizeCelebrities` untuk setiap selebriti yang telah dikenali dalam sebuah citra.

Jika Anda memiliki koleksi citra yang besar yang perlu diproses untuk pengenalan selebriti, pertimbangkan penggunaan [AWS Batch](#) untuk memproses panggilan ke `RecognizeCelebrities` pada batch di latar belakang. Saat menambahkan citra baru ke koleksi, Anda dapat menggunakan fungsi AWS Lambda untuk mengenali selebriti dengan memanggil `RecognizeCelebrities` saat citra diunggah ke dalam bucket S3.

Memanggil `RecognizeCelebrities`

Anda dapat memberikan citra input sebagai array bit citra (bit citra yang dikodekan base64) atau sebagai objek Amazon S3, dengan menggunakan AWS Command Line Interface (AWS CLI) atau AWS SDK. Di prosedur AWS CLI, Anda mengunggah sebuah citra dalam format .jpg atau .png ke Bucket S3. Di Prosedur SDK AWS, Anda menggunakan citra yang dimuat dari sistem file lokal Anda. Untuk informasi tentang rekomendasi input citra, lihat [Bekerja dengan citra](#).

Untuk menjalankan prosedur ini, Anda memerlukan file citra yang berisi satu wajah selebriti atau lebih.

Untuk mengenali selebriti dalam sebuah citra

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan `AmazonRekognitionFullAccess` dan `AmazonS3ReadOnlyAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi SDK AWS CLI dan AWS. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut ini untuk memanggil operasi `RecognizeCelebrities`.

Java

Contoh ini menampilkan informasi tentang selebriti yang terdeteksi dalam sebuah citra.

Ubah nilai `photo` dengan nama jalur dan file dari sebuah file citra yang berisi satu wajah selebriti atau lebih.

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

```
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.Celebrity;
import com.amazonaws.services.rekognition.model.RecognizeCelebritiesRequest;
import com.amazonaws.services.rekognition.model.RecognizeCelebritiesResult;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import com.amazonaws.util.IOUtils;
import java.util.List;

public class RecognizeCelebrities {

    public static void main(String[] args) {
        String photo = "moviestars.jpg";

        AmazonRekognition rekognitionClient =
        AmazonRekognitionClientBuilder.defaultClient();

        ByteBuffer imageBytes=null;
        try (InputStream inputStream = new FileInputStream(new File(photo))) {
            imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
        }
        catch(Exception e)
        {
            System.out.println("Failed to load file " + photo);
            System.exit(1);
        }

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

        System.out.println("Looking for celebrities in image " + photo + "\n");

        RecognizeCelebritiesResult
        result=rekognitionClient.recognizeCelebrities(request);

        //Display recognized celebrity information
```

```

List<Celebrity> celebs=result.getCelebrityFaces();
System.out.println(celebs.size() + " celebrity(s) were recognized.\n");

for (Celebrity celebrity: celebs) {
    System.out.println("Celebrity recognized: " + celebrity.getName());
    System.out.println("Celebrity ID: " + celebrity.getId());
    BoundingBox boundingBox=celebrity.getFace().getBoundingBox();
    System.out.println("position: " +
        boundingBox.getLeft().toString() + " " +
        boundingBox.getTop().toString());
    System.out.println("Further information (if available):");
    for (String url: celebrity.getUrls()){
        System.out.println(url);
    }
    System.out.println();
}
System.out.println(result.getUnrecognizedFaces().size() + " face(s) were
unrecognized.");
}
}

```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```

//snippet-start:[rekognition.java2.recognize_celebs.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesRequest;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Celebrity;
//snippet-end:[rekognition.java2.recognize_celebs.import]

```

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

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "  <sourceImage>\n\n" +
            "Where:\n" +
            "  sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png). \n\n";

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

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
            .build();

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

    // snippet-start:[rekognition.java2.recognize_celebs.main]
    public static void recognizeAllCelebrities(RekognitionClient rekClient, String
sourceImage) {

        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
```

```
    SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
    Image souImage = Image.builder()
        .bytes(sourceBytes)
        .build();

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

    RecognizeCelebritiesResponse result =
rekClient.recognizeCelebrities(request) ;
    List<Celebrity> celebs=result.celebrityFaces();
    System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
    for (Celebrity celebrity: celebs) {
        System.out.println("Celebrity recognized: " + celebrity.name());
        System.out.println("Celebrity ID: " + celebrity.id());

        System.out.println("Further information (if available):");
        for (String url: celebrity.urls()){
            System.out.println(url);
        }
        System.out.println();
    }
    System.out.println(result.unrecognizedFaces().size() + " face(s) were
unrecognized.");

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

AWS CLI

Perintah AWS CLI ini menampilkan output JSON untuk operasi CLI `recognize-celebrities`.

Ubah `bucketname` dengan nama bucket Amazon S3 yang berisi citra. Ubah `input.jpg` ke nama file dari sebuah citra yang berisi satu wajah selebriti atau lebih.

Ganti nilai `profile_name` dengan nama profil pengembang Anda.

```
aws rekognition recognize-celebrities \
  --image "S3object={Bucket=bucketname,Name=input.jpg}"
```

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda alih-alih tanda kutip tunggal dan hindari tanda kutip ganda bagian dalam dengan garis miring terbalik (yaitu `\`) untuk mengatasi kesalahan parser yang mungkin Anda temui. Sebagai contoh, lihat berikut ini:

```
aws rekognition recognize-celebrities --
image \
  "{\"S3object\":{\"Bucket\":\"bucket-name\",
  \"Name\":\"image-name\"}}\" --profile profile-name
```

Python

Contoh ini menampilkan informasi tentang selebriti yang terdeteksi dalam sebuah citra.

Ubah nilai `photo` dengan nama jalur dan file dari sebuah file citra yang berisi satu wajah selebriti atau lebih.

Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import boto3

def recognize_celebrities(photo):

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

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

```
print('Detected faces for ' + photo)
for celebrity in response['CelebrityFaces']:
    print('Name: ' + celebrity['Name'])
    print('Id: ' + celebrity['Id'])
    print('KnownGender: ' + celebrity['KnownGender']['Type'])
    print('Smile: ' + str(celebrity['Face']['Smile']['Value']))
    print('Position:')
    print('  Left: ' + '{:.2f}'.format(celebrity['Face']['BoundingBox']
['Height']))
    print('  Top: ' + '{:.2f}'.format(celebrity['Face']['BoundingBox']
['Top']))
    print('Info')
    for url in celebrity['Urls']:
        print('  ' + url)
    print()
return len(response['CelebrityFaces'])

def main():
    photo = 'photo-name'
    celeb_count = recognize_celebrities(photo)
    print("Celebrities detected: " + str(celeb_count))

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

Node.Js

Contoh ini menampilkan informasi tentang selebriti yang terdeteksi dalam sebuah citra.

Ubah nilai photo dengan nama jalur dan file dari sebuah file citra yang berisi satu wajah selebriti atau lebih. Ubah nilai bucket ke nama bucket S3 yang berisi file gambar yang disediakan. Ubah nilai REGION ke nama wilayah yang terkait dengan pengguna Anda. Ganti nilai profile_name di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

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



```
const profileName = "profile-name";

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

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

// Set params
const params = {
  Image: {
    S3Object: {
      Bucket: bucket,
      Name: photo
    },
  },
}

const recognize_celebrity = async() => {
  try {
    const response = await rekogClient.send(new
    RecognizeCelebritiesCommand(params));
    console.log(response.Labels)
    response.CelebrityFaces.forEach(celebrity =>{
      console.log(`Name: ${celebrity.Name}`)
      console.log(`ID: ${celebrity.Id}`)
      console.log(`KnownGender: ${celebrity.KnownGender.Type}`)
      console.log(`Smile: ${celebrity.Smile}`)
      console.log('Position: ')
      console.log(`  Left: ${celebrity.Face.BoundingBox.Height}`)
      console.log(`  Top : ${celebrity.Face.BoundingBox.Top}`)

    })
    return response.length; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
}

recognize_celebrity()
```

.NET

Contoh ini menampilkan informasi tentang selebriti yang terdeteksi dalam sebuah citra.

Ubah nilai photo menjadi nama jalur dan file dari file citra yang berisi satu wajah selebriti atau lebih (.jpg atau .png format).

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

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

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

        AmazonRekognitionClient rekognitionClient = new
AmazonRekognitionClient();

        RecognizeCelebritiesRequest recognizeCelebritiesRequest = new
RecognizeCelebritiesRequest();

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

```
        return;
    }

    img.Bytes = new MemoryStream(data);
    recognizeCelebritiesRequest.Image = img;

    Console.WriteLine("Looking for celebrities in image " + photo + "\n");

    RecognizeCelebritiesResponse recognizeCelebritiesResponse =
    rekognitionClient.RecognizeCelebrities(recognizeCelebritiesRequest);

    Console.WriteLine(recognizeCelebritiesResponse.CelebrityFaces.Count + "
celebrity(s) were recognized.\n");
    foreach (Celebrity celebrity in
    recognizeCelebritiesResponse.CelebrityFaces)
    {
        Console.WriteLine("Celebrity recognized: " + celebrity.Name);
        Console.WriteLine("Celebrity ID: " + celebrity.Id);
        BoundingBox boundingBox = celebrity.Face.BoundingBox;
        Console.WriteLine("position: " +
            boundingBox.Left + " " + boundingBox.Top);
        Console.WriteLine("Further information (if available):");
        foreach (String url in celebrity.Urls)
            Console.WriteLine(url);
    }
    Console.WriteLine(recognizeCelebritiesResponse.UnrecognizedFaces.Count +
" face(s) were unrecognized.");
    }
}
```

3. Catatan nilai dari salah satu ID selebriti yang ditampilkan. Anda akan membutuhkannya di [Mendapatkan informasi tentang selebriti](#).

RecognizeCelebrities permintaan operasi

Input ke `RecognizeCelebrities` adalah citra. Dalam contoh ini, citra dilewatkan sebagai bit citra. Untuk informasi selengkapnya, lihat [Bekerja dengan citra](#).

```
{
  "Image": {
    "Bytes": "/AoSiyvFpm....."
  }
}
```

```
}
```

RecognizeCelebrities respon operasi

Berikut ini adalah contoh input dan output JSON untuk `RecognizeCelebrities`.

`RecognizeCelebrities` mengirimkan array selebriti yang dikenal dan array wajah yang tidak dikenal. Pada contoh beriku ini, perhatikan beberapa hal ini:

- Selebriti yang dikenal — `Celebrities` adalah array selebriti yang dikenal. Setiap objek [Selebriti](#) dalam larik berisi nama selebriti dan daftar URL yang menunjuk ke konten terkait—misalnya, tautan IMDB atau Wikidata selebriti. Amazon Rekognition mengirimkan sebuah objek [ComparedFace](#) yang dapat digunakan aplikasi untuk memastikan letak wajah selebriti pada citra dan pengenalan yang unik untuk selebriti tersebut. Gunakan pengenalan unik untuk mengambil informasi terbaru selebriti dengan Operasi API [GetCelebrityInfo](#).
- Wajah tak dikenal — `UnrecognizedFaces` adalah array wajah yang tidak cocok dengan selebriti yang dikenal. Setiap objek [ComparedFace](#) dalam array yang berisi kotak batas (serta informasi lainnya) yang dapat Anda gunakan untuk menemukan wajah dalam citra.

```
{
  "CelebrityFaces": [{
    "Face": {
      "BoundingBox": {
        "Height": 0.617123007774353,
        "Left": 0.15641026198863983,
        "Top": 0.10864841192960739,
        "Width": 0.3641025722026825
      },
      "Confidence": 99.99589538574219,
      "Emotions": [{
        "Confidence": 96.3981749057023,
        "Type": "Happy"
      }
    ],
    "Landmarks": [{
      "Type": "eyeLeft",
      "X": 0.2837241291999817,
      "Y": 0.3637104034423828
    }], {
```

```
        "Type": "eyeRight",
        "X": 0.4091649055480957,
        "Y": 0.37378931045532227
    }, {
        "Type": "nose",
        "X": 0.35267341136932373,
        "Y": 0.49657556414604187
    }, {
        "Type": "mouthLeft",
        "X": 0.2786353826522827,
        "Y": 0.5455248355865479
    }, {
        "Type": "mouthRight",
        "X": 0.39566439390182495,
        "Y": 0.5597742199897766
    }
  ]],
  "Pose": {
    "Pitch": -7.749263763427734,
    "Roll": 2.004552125930786,
    "Yaw": 9.012002944946289
  },
  "Quality": {
    "Brightness": 32.69192123413086,
    "Sharpness": 99.9305191040039
  },
  "Smile": {
    "Confidence": 95.45394855702342,
    "Value": True
  }
},
"Id": "3Ir0du6",
"KnownGender": {
  "Type": "Male"
},
"MatchConfidence": 98.0,
"Name": "Jeff Bezos",
"Urls": ["www.imdb.com/name/nm1757263"]
}],
"OrientationCorrection": "NULL",
"UnrecognizedFaces": [{
  "BoundingBox": {
    "Height": 0.5345501899719238,
    "Left": 0.48461538553237915,
    "Top": 0.16949152946472168,
```

```
    "Width": 0.3153846263885498
  },
  "Confidence": 99.92860412597656,
  "Landmarks": [{
    "Type": "eyeLeft",
    "X": 0.5863404870033264,
    "Y": 0.36940744519233704
  }, {
    "Type": "eyeRight",
    "X": 0.6999204754829407,
    "Y": 0.3769848346710205
  }, {
    "Type": "nose",
    "X": 0.6349524259567261,
    "Y": 0.4804527163505554
  }, {
    "Type": "mouthLeft",
    "X": 0.5872702598571777,
    "Y": 0.5535582304000854
  }, {
    "Type": "mouthRight",
    "X": 0.6952020525932312,
    "Y": 0.5600858926773071
  }
  ],
  "Pose": {
    "Pitch": -7.386096477508545,
    "Roll": 2.304218292236328,
    "Yaw": -6.175624370574951
  },
  "Quality": {
    "Brightness": 37.16635513305664,
    "Sharpness": 99.9305191040039
  },
  "Smile": {
    "Confidence": 95.45394855702342,
    "Value": True
  }
}
}]
}
```

Mengenali selebriti dalam video yang tersimpan

Pengenalan selebriti Amazon Rekognition Video dalam video yang disimpan merupakan operasi yang tidak sinkron. Untuk mengenali selebriti dalam video yang tersimpan, gunakan [StartCelebrityRecognition](#) untuk memulai analisis pada video. Amazon Rekognition Video menerbitkan status penyelesaian analisis video ke topik Amazon Simple Notification Service. Jika analisis video berhasil, panggil [GetCelebrityRecognition](#) untuk mendapatkan hasil analisis. Untuk informasi selengkapnya tentang memulai analisis video dan mendapatkan hasilnya, lihat [Memanggil operasi Amazon Rekognition Video](#).

Prosedur ini melebar pada kode di [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#), yang menggunakan antrean Amazon SQS untuk mendapatkan status selesai dari permintaan analisis video. Untuk menjalankan prosedur ini, Anda memerlukan file video yang berisi satu wajah selebriti atau lebih.

Untuk mendeteksi selebriti dalam video yang disimpan dalam bucket Amazon S3 (SDK)

1. Lakukan [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#).
2. Tambahkan kode berikut ke kelas VideoDetect yang Anda buat di langkah 1.

Java

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

//
Celebrities=====
private static void StartCelebrityDetection(String bucket, String video)
throws Exception{

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

    StartCelebrityRecognitionRequest req = new
StartCelebrityRecognitionRequest()
        .withVideo(new Video()
            .withS3Object(new S3Object()
```

```
        .withBucket(bucket)
        .withName(video)))
    .withNotificationChannel(channel);

    StartCelebrityRecognitionResult startCelebrityRecognitionResult =
rek.startCelebrityRecognition(req);
    startJobId=startCelebrityRecognitionResult.getJobId();

}

private static void GetCelebrityDetectionResults() throws Exception{

    int maxResults=10;
    String paginationToken=null;
    GetCelebrityRecognitionResult celebrityRecognitionResult=null;

    do{
        if (celebrityRecognitionResult !=null){
            paginationToken = celebrityRecognitionResult.getNextToken();
        }
        celebrityRecognitionResult = rek.getCelebrityRecognition(new
GetCelebrityRecognitionRequest()
            .withJobId(startJobId)
            .withNextToken(paginationToken)
            .withSortBy(CelebrityRecognitionSortBy.TIMESTAMP)
            .withMaxResults(maxResults));

        System.out.println("File info for page");
        VideoMetadata
videoMetaData=celebrityRecognitionResult.getVideoMetadata();

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

        System.out.println("Job");

        System.out.println("Job status: " +
celebrityRecognitionResult.getJobStatus());
```



```
        //Show celebrities
        List<CelebrityRecognition> celebs=
celebrityRecognitionResult.getCelebrities();

        for (CelebrityRecognition celeb: celebs) {
            long seconds=celeb.getTimestamp()/1000;
            System.out.print("Sec: " + Long.toString(seconds) + " ");
            CelebrityDetail details=celeb.getCelebrity();
            System.out.println("Name: " + details.getName());
            System.out.println("Id: " + details.getId());
            System.out.println();
        }
    } while (celebrityRecognitionResult !=null &&
celebrityRecognitionResult.getNextToken() != null);

}
```

Dalam fungsi main, ganti baris:

```
StartLabelDetection(bucket, video);

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

dengan:

```
StartCelebrityDetection(bucket, video);

if (GetSQSMessagesSuccess()==true)
    GetCelebrityDetectionResults();
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
//snippet-start:[rekognition.java2.recognize_video_celebrity.import]
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;
//snippet-end:[rekognition.java2.recognize_video_celebrity.import]

/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class RecognizeCelebritiesVideo {

    private static String startJobId = "";

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

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

if (args.length != 4) {
    System.out.println(usage);
    System.exit(1);
}

String bucket = args[0];
String video = args[1];
String topicArn = args[2];
String roleArn = args[3];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
    .build();

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

StartCelebrityDetection(rekClient, channel, bucket, video);
GetCelebrityDetectionResults(rekClient);
System.out.println("This example is done!");
rekClient.close();
}

// snippet-start:[rekognition.java2.recognize_video_celebrity.main]
public static void StartCelebrityDetection(RekognitionClient rekClient,
                                           NotificationChannel channel,
                                           String bucket,
                                           String video){

    try {
```

```
S3Object s3obj = S3Object.builder()
    .bucket(bucket)
    .name(video)
    .build();

Video vid0b = Video.builder()
    .s3object(s3obj)
    .build();

StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
    .jobTag("Celebrities")
    .notificationChannel(channel)
    .video(vid0b)
    .build();

StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient.startCelebrityRecognition(recognitionRequest);
startJobId = startCelebrityRecognitionResult.jobId();

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

public static void GetCelebrityDetectionResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetCelebrityRecognitionResponse recognitionResponse = null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (recognitionResponse !=null)
                paginationToken = recognitionResponse.nextToken();

            GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
```

```
        .maxResults(10)
        .build();

    // Wait until the job succeeds
    while (!finished) {
        recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
        status = recognitionResponse.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    VideoMetadata videoMetaData=recognitionResponse.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " + videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<CelebrityRecognition> celebs= recognitionResponse.celebrities();
    for (CelebrityRecognition celeb: celebs) {
        long seconds=celeb.timestamp()/1000;
        System.out.print("Sec: " + seconds + " ");
        CelebrityDetail details=celeb.celebrity();
        System.out.println("Name: " + details.name());
        System.out.println("Id: " + details.id());
        System.out.println();
    }

    } while (recognitionResponse.nextToken() != null);

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

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

Python

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/  
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
# ===== Celebrities =====  
def StartCelebrityDetection(self):  
    response=self.rek.start_celebrity_recognition(Video={'S3Object':  
{'Bucket': self.bucket, 'Name': self.video}},  
        NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':  
self.snsTopicArn})  
  
    self.startJobId=response['JobId']  
    print('Start Job Id: ' + self.startJobId)  
  
def GetCelebrityDetectionResults(self):  
    maxResults = 10  
    paginationToken = ''  
    finished = False  
  
    while finished == False:  
        response = self.rek.get_celebrity_recognition(JobId=self.startJobId,  
                                                    MaxResults=maxResults,  
                                                    NextToken=paginationToken)  
  
        print(response['VideoMetadata']['Codec'])  
        print(str(response['VideoMetadata']['DurationMillis']))  
        print(response['VideoMetadata']['Format'])  
        print(response['VideoMetadata']['FrameRate'])  
  
        for celebrityRecognition in response['Celebrities']:  
            print('Celebrity: ' +  
                  str(celebrityRecognition['Celebrity']['Name']))  
            print('Timestamp: ' + str(celebrityRecognition['Timestamp']))  
            print()  
  
        if 'NextToken' in response:  
            paginationToken = response['NextToken']
```

```
else:
    finished = True
```

Dalam fungsi main, ganti baris:

```
analyzer.StartLabelDetection()
if analyzer.GetSQSMessagesSuccess()==True:
    analyzer.GetLabelDetectionResults()
```

dengan:

```
analyzer.StartCelebrityDetection()
if analyzer.GetSQSMessagesSuccess()==True:
    analyzer.GetCelebrityDetectionResults()
```

Node.JS

Dalam contoh kode Node.Js berikut, ganti nilai bucket dengan nama bucket S3 yang berisi video Anda dan nilai videoName dengan nama file video. Anda juga harus mengganti nilai dengan Arn yang terkait roleArn dengan peran layanan IAM Anda. Akhirnya, ganti nilai region dengan nama wilayah operasi yang terkait dengan akun Anda. Ganti nilai profile_name di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

// Import required AWS SDK clients and commands for Node.js
import { CreateQueueCommand, GetQueueAttributesCommand, GetQueueUrlCommand,
    SetQueueAttributesCommand, DeleteQueueCommand, ReceiveMessageCommand,
    DeleteMessageCommand } from "@aws-sdk/client-sqs";
import { CreateTopicCommand, SubscribeCommand, DeleteTopicCommand } from "@aws-
sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";
import { SNSClient } from "@aws-sdk/client-sns";
import { RekognitionClient, StartLabelDetectionCommand,
    GetLabelDetectionCommand,
    StartCelebrityRecognitionCommand, GetCelebrityRecognitionCommand} from "@aws-
sdk/client-rekognition";
import { stdout } from "process";
```

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

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

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

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

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

const createTopicandQueue = async () => {
  try {
    // Create SNS topic
    const topicResponse = await snsClient.send(new
CreateTopicCommand(snsTopicParams));
    const topicArn = topicResponse.TopicArn
    console.log("Success", topicResponse);
    // Create SQS Queue
```



```
    const sqsResponse = await sqsClient.send(new
CreateQueueCommand(sqsParams));
    console.log("Success", sqsResponse);
    const sqsQueueCommand = await sqsClient.send(new
GetQueueUrlCommand({QueueName: sqsQueueName}))
    const sqsQueueUrl = sqsQueueCommand.QueueUrl
    const attrsResponse = await sqsClient.send(new
GetQueueAttributesCommand({QueueUrl: sqsQueueUrl, AttributeNames:
['QueueArn']}))
    const attrs = attrsResponse.Attributes
    console.log(attrs)
    const queueArn = attrs.QueueArn
    // subscribe SQS queue to SNS topic
    const subscribed = await snsClient.send(new SubscribeCommand({TopicArn:
topicArn, Protocol:'sqs', Endpoint: queueArn}))
    const policy = {
      Version: "2012-10-17",
      Statement: [
        {
          Sid: "MyPolicy",
          Effect: "Allow",
          Principal: {AWS: "*"},
          Action: "SQS:SendMessage",
          Resource: queueArn,
          Condition: {
            ArnEquals: {
              'aws:SourceArn': topicArn
            }
          }
        }
      ]
    }
  ];

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

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

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

const getCelebrityRecognitionResults = async(startJobId) =>{
  try {
    //Initiate label detection and update value of startJobId with returned
    Job ID
    var maxResults = 10
    var paginationToken = ''
    var finished = false

    while (finished == false){
      var response = await rekClient.send(new
      GetCelebrityRecognitionCommand({JobId: startJobId, MaxResults: maxResults,
      NextToken: paginationToken}))
      console.log(response.VideoMetadata.Codec)
      console.log(response.VideoMetadata.DurationMillis)
      console.log(response.VideoMetadata.Format)
      console.log(response.VideoMetadata.FrameRate)
      response.Celebrities.forEach(celebrityRecognition => {
        console.log(`Celebrity: ${celebrityRecognition.Celebrity.Name}`)
        console.log(`Timestamp: ${celebrityRecognition.Timestamp}`)
        console.log()
      })
      // Search for pagination token, if found, set variable to next token
      if (String(response).includes("NextToken")){
        paginationToken = response.NextToken
      }else{
        finished = true
      }
    }
  }
}
```

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

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

    // Once job found, log Job ID and return true if status is succeeded
    for (var message of sqsReceivedResponse.Messages){
      console.log("Retrieved messages:")
      var notification = JSON.parse(message.Body)
      var rekMessage = JSON.parse(notification.Message)
      var messageJobId = rekMessage.JobId
      if (String(rekMessage.JobId).includes(String(startJobId))){
        console.log('Matching job found:')
      }
    }
  }
}
```

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

// Start label detection job, sent status notification, check for success
status
// Retrieve results if status is "SUCCEEDED", delete notification queue and
topic
const runCelebRecognitionAndGetResults = async () => {
    try {
        const sqsAndTopic = await createTopicandQueue();
        //const startLabelDetectionRes = await startLabelDetection(roleArn,
sqsAndTopic[1]);
        //const getSqsMessageStatus = await getSqsMessageSuccess(sqsAndTopic[0],
startLabelDetectionRes)
        const startCelebrityDetectionRes = await startCelebrityDetection(roleArn,
sqsAndTopic[1]);
        const getSqsMessageStatus = await getSqsMessageSuccess(sqsAndTopic[0],
startCelebrityDetectionRes)
        console.log(getSqsMessageSuccess)
        if (getSqsMessageSuccess){
            console.log("Retrieving results:")
            const results = await
getCelebrityRecognitionResults(startCelebrityDetectionRes)
```

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

runCelebRecognitionAndGetResults()
```

CLI

Jalankan AWS CLI perintah berikut untuk mulai mendeteksi selebriti dalam video.

```
aws rekognition start-celebrity-recognition --video '{"S3Object":
{"Bucket":"bucket-name","Name":"video-name"}}' \
--notification-channel '{"SNSTopicArn":"topic-arn","RoleArn":"role-arn"}' \
--region region-name --profile profile-name
```

Perbarui nilai berikut:

- Ubah `bucket-name` dan `video-name` ke nama bucket Amazon S3 dan nama file yang Anda tentukan pada langkah 2.
- Ubah `region-name` ke wilayah AWS yang Anda gunakan.
- Ganti nilai `profile-name` dengan nama profil pengembang Anda.
- Ubah `topic-ARN` ke ARN dari topik Amazon SNS yang Anda buat pada langkah 3 [Mengonfigurasi Amazon Rekognition Video](#).
- Perubah `role-ARN` ke ARN dari peran layanan IAM yang Anda buat di langkah 7 [Mengonfigurasi Amazon Rekognition Video](#).

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda alih-alih tanda kutip tunggal dan hindari tanda kutip ganda bagian dalam dengan garis miring terbalik (yaitu \) untuk mengatasi kesalahan parser yang mungkin Anda temui. Sebagai contoh, lihat di bawah:

```
aws rekognition start-celebrity-recognition --video "{\"S3Object\":{\"Bucket\":  
\"bucket-name\"},\"Name\":{\"video-name\"}}\" \  
--notification-channel "{\"SNSTopicArn\":{\"topic-arn\"},\"RoleArn\":{\"role-arn  
\"}\"}\" \  
--region region-name --profile profile-name
```

Setelah menjalankan contoh kode proses, salin yang dikembalikan jobID dan berikan ke GetCelebrityRecognition perintah berikut di bawah ini untuk mendapatkan hasil Anda, ganti job-id-number dengan yang jobID Anda terima sebelumnya:

```
aws rekognition get-celebrity-recognition --job-id job-id-number --profile  
profile-name
```

Note

Jika Anda sudah menjalankan contoh video selain [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#), kode yang akan diganti mungkin berbeda.

3. Jalankan kode tersebut. Informasi tentang selebriti yang dikenal dalam video sedang ditampilkan.

GetCelebrityRecognition respon operasi

Berikut ini adalah contoh respons JSON. Respons mencakup hal berikut ini:

- Selebriti yang dikenal — `Celebrities` adalah array selebriti dan waktu ketika selebriti tersebut dikenali dalam video. Objek [CelebrityRecognition](#) muncul setiap kali selebriti dikenal dalam video. Setiap `CelebrityRecognition` berisi informasi tentang selebriti yang dikenal ([CelebrityDetail](#)) dan waktu (`Timestamp`) selebriti itu dikenal dalam video. `Timestamp` diukur dalam satuan milidetik dari permulaan video.
- `CelebrityDetail`— Berisi informasi tentang selebriti yang diakui. Ini termasuk nama selebriti (`Name`), identifier (`ID`), jenis kelamin selebriti yang diketahui (`KnownGender`), dan daftar URL yang menunjuk ke konten terkait (`Urls`). `Urls` Ini juga mencakup tingkat kepercayaan yang dimiliki

Amazon Rekognition Video dalam keakuratan pengakuan, dan detail tentang wajah selebriti,. [FaceDetail](#) Jika Anda perlu mendapatkan konten terkait nanti, Anda dapat menggunakan ID dengan [getCelebrityInfo](#).

- VideoMetadata— Informasi tentang video yang dianalisis.

```
{
  "Celebrities": [
    {
      "Celebrity": {
        "Confidence": 0.699999988079071,
        "Face": {
          "BoundingBox": {
            "Height": 0.20555555820465088,
            "Left": 0.029374999925494194,
            "Top": 0.223333332896232605,
            "Width": 0.11562500149011612
          },
          "Confidence": 99.89837646484375,
          "Landmarks": [
            {
              "Type": "eyeLeft",
              "X": 0.06857934594154358,
              "Y": 0.30842265486717224
            },
            {
              "Type": "eyeRight",
              "X": 0.10396526008844376,
              "Y": 0.300625205039978
            },
            {
              "Type": "nose",
              "X": 0.0966852456331253,
              "Y": 0.34081998467445374
            },
            {
              "Type": "mouthLeft",
              "X": 0.075217105448246,
              "Y": 0.3811396062374115
            },
            {
              "Type": "mouthRight",
              "X": 0.10744428634643555,
```

```

        "Y": 0.37407416105270386
      }
    ],
    "Pose": {
      "Pitch": -0.9784082174301147,
      "Roll": -8.808176040649414,
      "Yaw": 20.28228759765625
    },
    "Quality": {
      "Brightness": 43.312068939208984,
      "Sharpness": 99.9305191040039
    }
  },
  "Id": "XXXXXX",
  "KnownGender": {
    "Type": "Female"
  },
  "Name": "Celeb A",
  "Urls": []
},
"Timestamp": 367
},.....
],
"JobStatus": "SUCCEEDED",
"NextToken": "XfXnZKiyMOGDhzBzYUhS5puM+g1IgezqFeYpv/H/+5noP/LmM57FitUAwSQ5D6G4AB/PNwolrw==",
"VideoMetadata": {
  "Codec": "h264",
  "DurationMillis": 67301,
  "FileExtension": "mp4",
  "Format": "QuickTime / MOV",
  "FrameHeight": 1080,
  "FrameRate": 29.970029830932617,
  "FrameWidth": 1920
}
}

```

Mendapatkan informasi tentang selebriti

Dalam prosedur ini, Anda mendapatkan informasi selebriti dengan menggunakan Operasi API [getCelebrityInfo](#). Selebriti diidentifikasi dengan menggunakan ID selebriti yang dikirimkan dari panggilan sebelumnya ke [RecognizeCelebrities](#).

Memanggil GetCelebrityInfo

Prosedur ini memerlukan ID selebriti untuk selebriti yang diketahui oleh Amazon Rekognition. Gunakan ID selebriti yang Anda catat di [Mengenali selebriti dalam sebuah citra](#).

Untuk mendapatkan informasi selebriti (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan `AmazonRekognitionFullAccess` dan `AmazonS3ReadOnlyAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi SDK AWS CLI dan AWS. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan contoh berikut ini untuk memanggil operasi `GetCelebrityInfo`.

Java

Contoh ini menampilkan nama dan informasi tentang selebriti.

Ganti id dengan salah satu ID selebriti yang ditampilkan di [Mengenali selebriti dalam sebuah citra](#).

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

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.GetCelebrityInfoRequest;
import com.amazonaws.services.rekognition.model.GetCelebrityInfoResult;

public class CelebrityInfo {

    public static void main(String[] args) {
        String id = "nnnnnnnn";

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

```
GetCelebrityInfoRequest request = new GetCelebrityInfoRequest()
    .withId(id);

System.out.println("Getting information for celebrity: " + id);

GetCelebrityInfoResult
result=rekognitionClient.getCelebrityInfo(request);

//Display celebrity information
System.out.println("celebrity name: " + result.getName());
System.out.println("Further information (if available):");
for (String url: result.getUrls()){
    System.out.println(url);
}
}
}
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityInfoRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityInfoResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

```
final String usage = ""

    Usage:    <id>

    Where:
        id - The id value of the celebrity. You can use the
RecognizeCelebrities example to get the ID value.\s
    """;

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

String id = args[0];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

getCelebrityInfo(rekClient, id);
rekClient.close();
}

public static void getCelebrityInfo(RekognitionClient rekClient, String id)
{
    try {
        GetCelebrityInfoRequest info = GetCelebrityInfoRequest.builder()
            .id(id)
            .build();

        GetCelebrityInfoResponse response =
rekClient.getCelebrityInfo(info);
        System.out.println("celebrity name: " + response.name());
        System.out.println("Further information (if available):");
        for (String url : response.urls()) {
            System.out.println(url);
        }

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

```
}
```

AWS CLI

Perintah AWS CLI ini menampilkan output JSON untuk operasi CLI `get-celebrity-info`. Ganti ID dengan salah satu ID selebriti yang ditampilkan di [Mengenali selebriti dalam sebuah citra](#). Ganti nilai `profile-name` dengan nama profil pengembang Anda.

```
aws rekognition get-celebrity-info --id celebrity-id --profile profile-name
```

Python

Contoh ini menampilkan nama dan informasi tentang selebriti.

Ganti `id` dengan salah satu ID selebriti yang ditampilkan di [Mengenali selebriti dalam sebuah citra](#). Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import boto3

def get_celebrity_info(id):

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

    # Display celebrity info
    print('Getting celebrity info for celebrity: ' + id)

    response = client.get_celebrity_info(Id=id)

    print(response['Name'])
    print('Further information (if available):')
    for url in response['Urls']:
        print(url)

def main():
    id = "celebrity-id"
    celebrity_info = get_celebrity_info(id)
```

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

.NET

Contoh ini menampilkan nama dan informasi tentang selebriti.

Ganti id dengan salah satu ID selebriti yang ditampilkan di [Mengenali selebriti dalam sebuah citra](#).

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/  
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
using System;  
using Amazon.Rekognition;  
using Amazon.Rekognition.Model;  
  
public class CelebrityInfo  
{  
    public static void Example()  
    {  
        String id = "nnnnnnnn";  
  
        AmazonRekognitionClient rekognitionClient = new  
AmazonRekognitionClient();  
  
        GetCelebrityInfoRequest celebrityInfoRequest = new  
GetCelebrityInfoRequest()  
        {  
            Id = id  
        };  
  
        Console.WriteLine("Getting information for celebrity: " + id);  
  
        GetCelebrityInfoResponse celebrityInfoResponse =  
rekognitionClient.GetCelebrityInfo(celebrityInfoRequest);  
  
        //Display celebrity information  
        Console.WriteLine("celebrity name: " + celebrityInfoResponse.Name);  
        Console.WriteLine("Further information (if available):");
```

```
        foreach (String url in celebrityInfoResponse.Urls)
            Console.WriteLine(url);
    }
}
```

GetCelebrityInfo permintaan operasi

Berikut ini adalah contoh input dan output JSON untuk GetCelebrityInfo.

Input ke GetCelebrityInfo merupakan ID untuk selebriti yang diperlukan.

```
{
  "Id": "nnnnnnnn"
}
```

GetCelebrityInfo respon operasi

GetCelebrityInfo mengirimkan array (Urls) tautan pada informasi tentang selebriti yang diminta.

```
{
  "Name": "Celebrity Name",
  "Urls": [
    "www.imdb.com/name/nmnnnnnnnn"
  ]
}
```

Memoderasi konten

Anda dapat menggunakan Amazon Rekognition untuk mendeteksi konten yang tidak pantas, tidak diinginkan, atau menyinggung. Anda dapat menggunakan API moderasi Rekognition di situasi media sosial, media siaran, iklan, dan perdagangan elektronik untuk membuat pengalaman pengguna yang lebih aman, memberikan jaminan keamanan merek kepada pengiklan, dan mematuhi peraturan lokal dan global.

Saat ini, banyak perusahaan sepenuhnya bergantung pada moderator manusia untuk meninjau konten pihak ke tiga atau buatan pengguna, sedangkan yang lain hanya bereaksi terhadap aduan pengguna untuk menghapus gambar, iklan, atau video yang menyinggung atau tidak pantas. Namun, moderator manusia saja tidak dapat menskalakan untuk memenuhi kebutuhan ini dengan kualitas atau kecepatan yang memadai, yang mengarah ke pengalaman pengguna yang buruk, biaya tinggi untuk mencapai skala, atau bahkan hilangnya reputasi merek. Dengan menggunakan Rekognition untuk moderasi citra dan video, moderator manusia dapat meninjau set konten yang jauh lebih kecil, biasanya 1-5% dari total volume, yang telah ditandai oleh machine learning. Hal ini memungkinkan mereka untuk fokus pada kegiatan yang lebih berharga dan masih mencapai cakupan moderasi yang komprehensif dengan sebagian kecil dari biaya yang ada. Untuk menyiapkan tenaga kerja manusia dan melakukan tugas tinjauan manusia, Anda dapat menggunakan Amazon Augmented AI, yang sudah terintegrasi dengan Rekognition.

Anda dapat meningkatkan akurasi model pembelajaran mendalam moderasi dengan fitur Custom Moderation. Dengan Custom Moderation, Anda melatih adaptor moderasi khusus dengan mengunggah gambar Anda dan membuat anotasi gambar-gambar ini. Adaptor terlatih kemudian dapat diberikan ke operasi [DetectModerationLabel](#) untuk meningkatkan kinerjanya pada gambar Anda. Untuk informasi selengkapnya, lihat [Meningkatkan akurasi dengan Custom Moderation](#).

Label didukung oleh operasi moderasi konten Rekognition

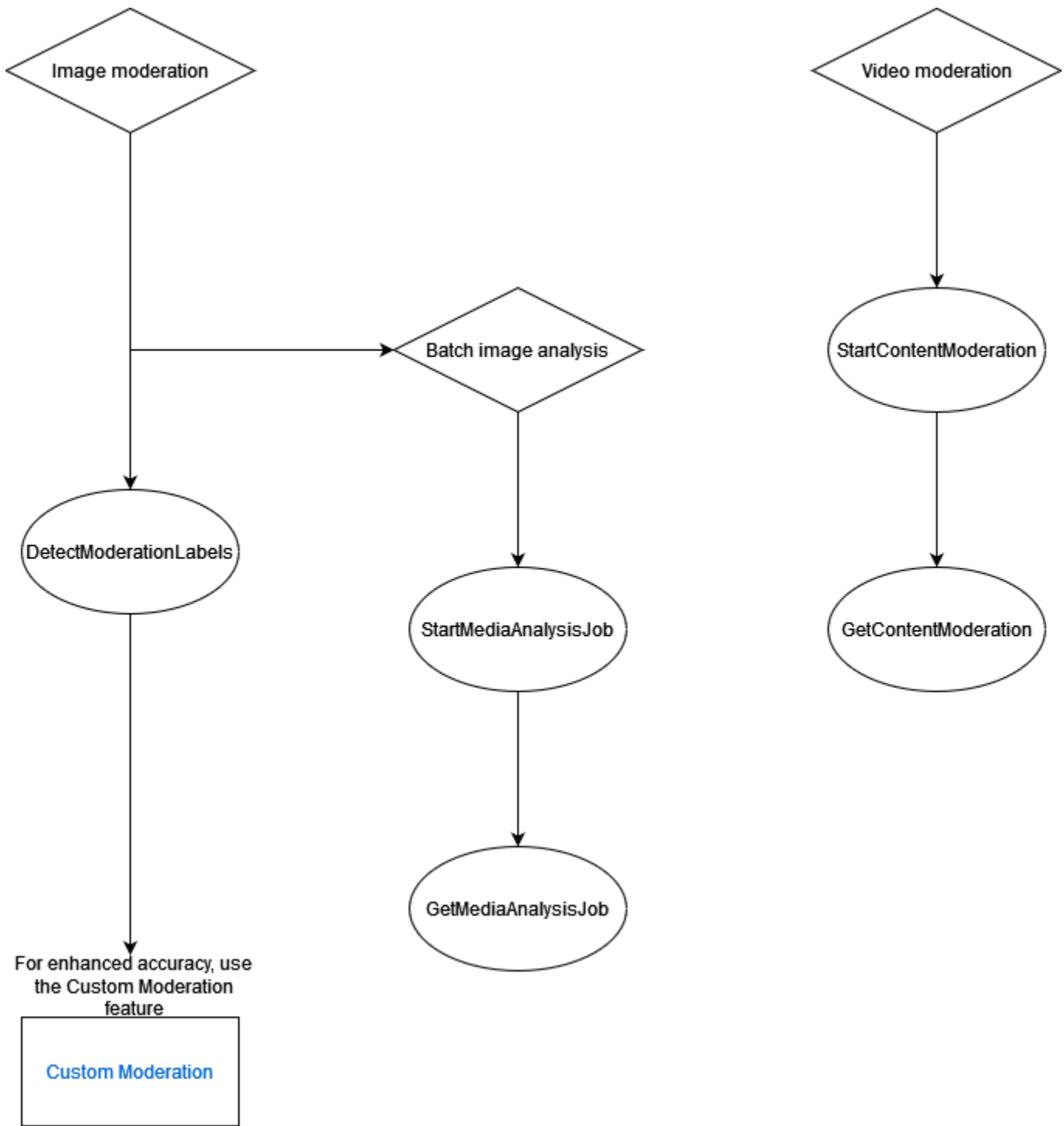
- Untuk mengunduh daftar label moderasi, klik [di sini](#).

Topik

- [Menggunakan API moderasi citra dan video](#)
- [Menguji Moderasi Konten versi 7 dan mengubah respons API](#)
- [Mendeteksi citra yang tidak pantas](#)
- [Mendeteksi video tersimpan yang tidak pantas](#)

- [Meningkatkan akurasi dengan Custom Moderation](#)
- [Meninjau konten yang tidak sesuai dengan Amazon Augmented AI](#)

Diagram berikut menunjukkan urutan operasi panggilan, tergantung pada tujuan Anda untuk menggunakan komponen gambar atau video dari Moderasi Konten:



Menggunakan API moderasi citra dan video

Di Amazon Rekognition Image API, Anda dapat mendeteksi konten yang tidak pantas, tidak diinginkan, atau menyinggung secara sinkron [DetectModeration](#) menggunakan Label dan penggunaan serta pengoperasian secara asinkron. [StartMediaAnalysisJob](#) [GetMediaAnalysisJob](#) [Anda dapat menggunakan Amazon Rekognition Video API untuk mendeteksi konten tersebut secara asinkron dengan menggunakan operasi Moderasi dan Moderasi. StartContent](#) [GetContent](#)

Kategori Label

Amazon Rekognition menggunakan taksonomi hierarkis tiga tingkat untuk memberi label kategori konten yang tidak pantas, tidak diinginkan, atau menyinggung. Setiap label dengan Taksonomi Level 1 (L1) memiliki sejumlah label Taksonomi Level 2 (L2), dan beberapa label Taksonomi Level 2 mungkin memiliki label Taksonomi Level 3 (L3). Ini memungkinkan klasifikasi hierarkis konten.

Untuk setiap label moderasi yang terdeteksi, API juga mengembalikan `TaxonomyLevel`, yang berisi level (1, 2, atau 3) yang dimiliki label tersebut. Misalnya, gambar dapat diberi label sesuai dengan kategorisasi berikut:

L1: Ketelanjangan Non-Eksplisit dari bagian Intim dan Ciuman, L2: Ketelanjangan Non-Eksplisit, L3: Ketelanjangan Tersirat.

Note

Sebaiknya gunakan kategori L1 atau L2 untuk memoderasi konten Anda dan menggunakan kategori L3 hanya untuk menghapus konsep spesifik yang tidak ingin Anda moderasi (yaitu untuk mendeteksi konten yang mungkin tidak ingin Anda kategorikan sebagai konten yang tidak pantas, tidak diinginkan, atau menyinggung berdasarkan kebijakan moderasi Anda).

Tabel berikut menunjukkan hubungan antara tingkat kategori dan label yang mungkin untuk setiap tingkat. Untuk mengunduh daftar label moderasi, klik [di sini](#).

Kategori Tingkat Atas (L1)	Kategori Tingkat Kedua (L2)	Kategori Tingkat Ketiga (L3)	Ketentuan
Eksplisit	Ketelanjangan Eksplisit	Alat Kelamin Pria Terpapar	Alat kelamin pria manusia, termasuk

	<p>penis (apakah ereksi atau lembek), skrotum, dan rambut kemaluan yang terlihat. Istilah ini berlaku dalam konteks yang melibatkan aktivitas seksual atau konten visual apa pun di mana alat kelamin pria ditampilkan baik seluruhnya atau sebagian.</p>
<p>Alat Kelamin Wanita Terpapar</p>	<p>Bagian luar dari sistem reproduksi wanita, meliputi vulva, vagina, dan rambut kemaluan yang dapat diamati. Istilah ini berlaku dalam skenario yang melibatkan aktivitas seksual atau konten visual apa pun di mana aspek-aspek anatomi wanita ini ditampilkan baik seluruhnya atau sebagian.</p>

Bokong atau Anus
Terpapar

Bokong atau anus manusia, termasuk kasus di mana bokong telanjang atau ketika mereka terlihat melalui pakaian tipis. Definisi ini secara khusus berlaku untuk situasi di mana bokong atau anus secara langsung dan sepenuhnya terlihat, tidak termasuk skenario di mana segala bentuk pakaian dalam atau pakaian memberikan cakupan lengkap atau sebagian.

Puting Wanita
Terpapar

Puting wanita manusia, termasuk aerola yang terlihat sepenuhnya dan sebagian terlihat (area di sekitar puting susu) dan puting susu.

Aktivitas Seksual Eksplisit	N/A	Penggambaran tindakan seksual aktual atau simulasi yang mencakup hubungan seksual manusia, seks oral, serta stimulasi genital pria dan stimulasi genital wanita oleh bagian dan objek tubuh lainnya. Istilah ini juga mencakup ejakulasi atau cairan vagina pada bagian tubuh dan praktik erotis atau permainan peran yang melibatkan n perbudakan, disiplin, dominasi dan ketundukan, dan sadomasokisme.
Mainan Seks	N/A	Objek atau perangkat yang digunakan untuk rangsangan atau kesenangan seksual, misalnya, Dildo, vibrator, butt plug, beats, dll.

Ketelanjangan Non-Eksplisit dari bagian intim dan ciuman	Ketelanjangan Non-Eksplisit	Telanjang Kembali	Bagian posterior manusia di mana sebagian besar kulit terlihat dari leher ke ujung tulang belakang. Istilah ini tidak berlaku ketika punggung individu sebagian atau seluruhnya tersumbat.
		Puting Pria Terpapar	Puting pria manusia, termasuk puting yang terlihat sebagian.
		Bokong Terkena Sebagian	Bokong manusia yang terpapar sebagian. Istilah ini mencakup daerah bokong atau pipi bokong yang terlihat sebagian karena pakaian pendek, atau bagian atas celah anus yang terlihat sebagian. Istilah ini tidak berlaku untuk kasus-kasus di mana bokong sepenuhnya telanjang

	Payudara Wanita Terpapar Sebagian	Payudara wanita manusia yang terpapar sebagian di mana satu bagian dari dada wanita terlihat atau terbuka sementara tidak mengungkapkan seluruh payudara ya. Istilah ini berlaku ketika daerah lipatan dada bagian dalam terlihat atau ketika lipatan dada bagian bawah terlihat dengan puting sepenuhnya tertutup atau tersumbat.
	Ketelanjangan Tersirat	Seseorang yang telanjang, baik topless atau tanpa dasar, tetapi dengan bagian intim seperti bokong, puting, atau alat kelamin tertutup, tersumbat, atau tidak sepenuhnya terlihat.
Bagian Intim yang Terhambat	Puting Wanita Terhambat	Penggambaran visual dari situasi di mana puting perempuan ditutupi oleh pakaian atau penutup buram, tetapi bentuknya terlihat jelas.

		Alat Kelamin Pria Terhambat	Penggambaran visual dari situasi di mana alat kelamin atau penis pria ditutupi oleh pakaian atau penutup buram, tetapi bentuknya terlihat jelas. Istilah ini berlaku ketika alat kelamin yang terhambat pada gambar berada dalam jarak dekat.
	Berciuman di Bibir	N/A	Penggambaran bibir seseorang melakukan kontak dengan bibir orang lain.
Pakaian Renang atau Pakaian Dalam	Pakaian Renang atau Pakaian Dalam Wanita	N/A	Pakaian manusia untuk pakaian renang wanita (misalnya , pakaian renang one-piece, bikini, tankinis, dll.) Dan pakaian dalam wanita (misalnya, bra, celana dalam, celana dalam, pakaian dalam, sandal jepit, dll.)

	Pakaian Renang atau Pakaian Dalam Pria	N/A	Pakaian manusia untuk pakaian renang pria (misalnya, celana renang, celana pendek papan, celana renang, dll.) Dan pakaian dalam pria (misalnya, celana dalam, petinju, dll.)
Kekerasan	Senjata	N/A	Instrumen atau perangkat yang digunakan untuk menyebabkan kerusakan atau kerusakan pada makhluk hidup, struktur, atau sistem. Ini termasuk senjata api (misalnya, senjata, senapan mesin, dll.), senjata tajam (misalnya, pedang, pisau, dll.), bahan peledak dan amunisi (misalnya, rudal, bom, peluru, dll.).
	Kekerasan Grafis	Kekerasan Senjata	Penggunaan senjata untuk menyebabkan kerusakan, kerusakan, cedera, atau kematian bagi diri sendiri, individu lain, atau properti.

Kekerasan Fisik	Tindakan yang menyebabkan kerusakan pada individu atau properti lain (misalnya, memukul, berkelahi, mencabut rambut, dll.) atau tindakan kekerasan lainnya yang melibatkan kerumunan atau banyak individu.
Menyakiti Diri Sendiri	Tindakan menyebabkan kerusakan pada diri sendiri, seringkali dengan memotong bagian tubuh seperti lengan atau kaki, di mana luka biasanya terlihat.
Darah & Gore	Representasi visual kekerasan pada seseorang, sekelompok individu, atau hewan, yang melibatkan luka terbuka, pertumpahan darah, dan bagian tubuh yang dimutilasi.

		Ledakan dan Ledakan	Penggambaran ledakan api yang hebat dan merusak dengan asap tebal atau debu dan asap meletus dari tanah.
Secara Visual Mengganggu	Kematian dan Kurus Kurus	Badan yang Kurus	Tubuh manusia yang sangat kurus dan kurang gizi dengan pemborosan fisik yang parah dan penipisan otot dan jaringan lemak.
		Mayat	Mayat manusia dalam bentuk tubuh yang dimutilasi, mayat gantung, atau kerangka.
	Kecelakaan	Kecelakaan Udara	Insiden kendaraan udara, seperti pesawat terbang, helikopter, atau kendaraan terbang lainnya, yang mengakibatkan kerusakan, cedera, atau kematian. Istilah ini berlaku ketika bagian-bagian dari kendaraan udara terlihat.

Obat-obatan & Tembakau	Produk	Pil	Meja atau kapsul kecil, padat, sering berbentuk bulat atau oval. Istilah ini berlaku untuk pil yang disajikan sebagai standalon es, dalam botol, atau paket transparan dan tidak berlaku untuk penggambaran visual seseorang yang minum pil.
	Obat-obatan & Perlengkapan Tembakau & Penggunaan	Merokok	Tindakan menghirup, menghembuskan napas, dan menyalakan zat yang terbakar termasuk rokok, cerutu, e-rokok, hookah, atau sendi.
Alkohol	Penggunaan Alkohol	Minum	Tindakan minum minuman beralkohol dari botol atau gelas alkohol atau minuman keras.

	Minuman Beralkohol	N/A	Tutup satu atau beberapa botol alkohol atau minuman keras, gelas atau mug dengan alkohol atau minuman keras, dan gelas atau mug dengan alkohol atau minuman keras yang dipegang oleh seseorang. Istilah ini tidak berlaku untuk individu yang minum dari botol atau gelas alkohol atau minuman keras.
Tindakan Kasar	Jari Tengah	N/A	Penggambaran visual gerakan tangan dengan jari tengah diperpanjang ke atas sementara jari-jari lainnya dilipat ke bawah.
Judi	N/A	N/A	Tindakan berpartisipasi dalam permainan kesempatan untuk kesempatan memenangkan hadiah di kasino, misalnya, bermain kartu, blackjacks, roulette, mesin slot di kasino, dll.

Simbol Kebencian	Partai Nazi	N/A	Penggambaran visual simbol, bendera, atau gerakan yang terkait dengan Partai Nazi.
	Supremasi Putih	N/A	Penggambaran visual simbol atau pakaian yang terkait dengan Ku Klux Klan (KKK) dan gambar dengan bendera konfederasi.
	Ekstremis	N/A	Gambar berisi bendera kelompok ekstremis dan teroris.

Tidak setiap label dalam kategori L2 memiliki label yang didukung dalam kategori L3. Selain itu, label L3 di bawah label L2 “Produk” dan “Perlengkapan dan Penggunaan Obat dan Tembakau” tidak lengkap. Label L2 ini mencakup konsep di luar label L3 yang disebutkan dan dalam kasus seperti itu, hanya label L2 yang dikembalikan dalam respons API.

Anda menentukan kesesuaian konten untuk aplikasi Anda. Misalnya, citra yang bersifat sugestif mungkin dapat diterima, namun citra yang mengandung ketelanjangan mungkin tidak. Untuk memfilter gambar, gunakan larik [ModerationLabel](#) yang dikembalikan oleh `DetectModerationLabels` (gambar) dan oleh `GetContentModeration` (video).

Jenis konten

API juga dapat mengidentifikasi jenis konten animasi atau bergambar, dan jenis konten dikembalikan sebagai bagian dari respons:

- Konten animasi termasuk video game dan animasi (misalnya, kartun, komik, manga, anime).
- Konten bergambar termasuk menggambar, melukis, dan sketsa.

Kepercayaan

Anda dapat mengatur ambang batas kepercayaan yang digunakan Amazon Rekognition untuk mendeteksi konten yang tidak pantas dengan menentukan parameter input `MinConfidence`. Label tidak dikembalikan untuk konten yang tidak pantas yang terdeteksi dengan kepercayaan yang lebih rendah dari `MinConfidence`.

Menentukan nilai untuk `MinConfidence` itu kurang dari 50% kemungkinan akan mengembalikan sejumlah besar hasil positif palsu (yaitu penarikan yang lebih tinggi, presisi lebih rendah). Di sisi lain, menentukan `MinConfidence` di atas 50% kemungkinan akan mengembalikan jumlah hasil positif palsu yang lebih rendah (yaitu penarikan yang lebih rendah, presisi lebih tinggi). Jika Anda tidak menentukan nilai untuk `MinConfidence`, Amazon Rekognition mengembalikan label untuk konten yang tidak pantas yang terdeteksi dengan kepercayaan setidaknya 50%.

Label berisi array `ModerationLabel` dalam kategori sebelumnya, dan perkiraan kepercayaan dalam keakuratan konten yang dikenali. Label tingkat atas dikembalikan bersama dengan label tingkat kedua yang diidentifikasi. Misalnya, Amazon Rekognition mungkin mengembalikan "Ketelanjangan Eksplisit" dengan skor kepercayaan yang tinggi sebagai label tingkat atas. Itu mungkin cukup untuk kebutuhan pemfilteran Anda. Namun, jika perlu, Anda dapat menggunakan skor kepercayaan label tingkat kedua (seperti "Ketelanjangan Laki-Laki Grafis") untuk mendapatkan pemfilteran yang lebih terperinci. Sebagai contoh, lihat [Mendeteksi citra yang tidak pantas](#).

Penentuan Versi

Baik Amazon Rekognition Image maupun Amazon Rekognition Video mengembalikan versi model deteksi moderasi yang digunakan untuk mendeteksi konten yang tidak pantas (`ModerationModelVersion`).

Menyortir dan Mengagregasi

Saat mengambil hasil dengan `GetContentModeration`, Anda dapat mengurutkan dan mengumpulkan hasil Anda.

Urutkan urutan - Array label yang dikembalikan diurutkan berdasarkan waktu. Untuk mengurutkan berdasarkan label, tentukan `NAME` dalam parameter input `SortBy` untuk `GetContentModeration`. Jika label muncul beberapa kali dalam video, akan ada beberapa contoh elemen `ModerationLabel`.

Informasi label - Elemen `ModerationLabels` array berisi `ModerationLabel` objek, yang pada gilirannya berisi nama label dan kepercayaan Amazon Rekognition dalam keakuratan label yang

terdeteksi. Stempel waktu adalah waktu `ModerationLabel` terdeteksi, didefinisikan sebagai jumlah milidetik yang berlalu sejak awal video. Untuk hasil yang dikumpulkan berdasarkan video `SEGMENTSStartTimeMillis`, `DurationMillis` struktur `EndTimeMillis`, dan dikembalikan, yang menentukan waktu mulai, waktu akhir, dan durasi segmen masing-masing.

Agregasi - Menentukan bagaimana hasil dikumpulkan ketika dikembalikan. Defaultnya adalah agregat dengan `TIMESTAMPS`. Anda juga dapat memilih untuk agregat menurut `SEGMENTS`, yang mengumpulkan hasil melalui jendela waktu. Hanya label yang terdeteksi selama segmen yang dikembalikan.

Status adaptor Moderasi Kustom

Adaptor Moderasi Kustom dapat berada di salah satu status berikut: `TRAINING_IN_PROGRESS`, `TRAINING_COMPLETED`, `TRAINING_FAILED`, `DELETING`, `DEPRECATED`, atau `EXPIRED`. Untuk penjelasan lengkap tentang status adaptor ini, lihat [Mengelola adaptor](#).

Note

Amazon Rekognition bukanlah otoritas, dan sama sekali tidak mengklaim sebagai filter lengkap dari konten yang tidak pantas atau menyinggung. Selain itu, API moderasi gambar dan video tidak mendeteksi apakah suatu gambar menyertakan konten ilegal, seperti CSAM.

Menguji Moderasi Konten versi 7 dan mengubah respons API

Rekognition memperbarui model pembelajaran mesin untuk komponen video gambar fitur deteksi label Moderasi Konten dari versi 6.1 hingga 7. Pembaruan ini meningkatkan akurasi keseluruhan, dan memperkenalkan beberapa kategori baru bersama dengan memodifikasi yang lain.

Jika Anda adalah pengguna video versi 6.1 saat ini, kami sarankan Anda untuk mengambil tindakan berikut untuk transisi mulus ke versi 7:

1. Unduh dan gunakan AWS private SDK (lihat [the section called "AWS SDK dan Panduan Penggunaan untuk Moderasi Konten versi 7"](#)) untuk memanggil `StartContentModeration` API.
2. Tinjau daftar label dan skor kepercayaan yang diperbarui yang ditampilkan dalam respons API atau konsol. Sesuaikan logika pasca-pemrosesan aplikasi Anda jika perlu.
3. Akun Anda akan tetap di versi 6.1 hingga 13 Mei 2024. Jika Anda ingin menggunakan versi 6.1 setelah 13 Mei 2024, hubungi [tim AWS Support paling lambat](#) 30 April 2024 untuk meminta

perpanjangan. Kami dapat memperpanjang akun Anda untuk tetap pada versi 6.1 hingga 10 Juni 2024. Jika kami tidak mendengar kabar dari Anda pada 30 April 2024, akun Anda akan secara otomatis dimigrasikan ke versi 7.0 mulai 13 Mei 2024.

AWS SDK dan Panduan Penggunaan untuk Moderasi Konten versi 7

Unduh SDK yang sesuai dengan bahasa pengembangan pilihan Anda, dan lihat panduan pengguna yang sesuai.

Tautan ke SDK

[Jawa-1.X](#)

[Jawa-2.X](#)

[JavaScript v2](#)

[JavaScript v3](#)

[Python](#)

[Ruby](#)

[go_v1](#)

[go_v2](#)

[DotNet](#)

[php](#)

Instalasi/Panduan Pengguna

[Panduan - Java 1.pdf](#)

[Panduan - Java 2.pdf](#)

[Panduan - JavaScript v2.pdf](#)

[Panduan - JavaScript v3.pdf](#)

[Panduan - Python & AWS CLI.pdf](#)

[Panduan - RubyV3.pdf](#)

[Panduan - GO V1.pdf](#)

[Panduan - GO V2.pdf](#)

[Panduan - .net.pdf](#)

[Panduan - PHP.pdf](#)

Pemetaan label untuk Versi 6.1 hingga 7

Moderasi konten versi 7 menambahkan kategori label baru dan memodifikasi nama label yang sudah ada sebelumnya. Referensikan tabel taksonomi yang ditemukan pada [the section called “ Kategori Label ”](#) saat memutuskan cara memetakan 6.1 label ke 7 label.

Beberapa contoh pemetaan label ditemukan di bagian berikut. Kami menyarankan Anda meninjau pemetaan ini dan definisi label sebelum membuat pembaruan yang diperlukan berdasarkan logika pasca-pemrosesan aplikasi Anda.

Skema Pemetaan L1

Jika Anda menggunakan logika pasca-pemrosesan yang hanya memfilter pada kategori tingkat atas (L1) (seperti `Explicit Nudity`, `Suggestive`, `Violence` dll), lihat tabel di bawah ini untuk memperbarui kode Anda.

V6.1 L1	V7 L1
Ketelanjangan Eksplisit	Eksplisit
Sugestif	Ketelanjangan Non-Eksplisit dari bagian intim dan ciuman
	Pakaian Renang atau Pakaian Dalam
Kekerasan	Kekerasan
Secara Visual Mengganggu	Secara Visual Mengganggu
Tindakan Kasar	Tindakan Kasar
Narkoba	Obat-obatan & Tembakau
Tembakau	Obat-obatan & Tembakau
Alkohol	Alkohol
Judi	Judi
Simbol Kebencian	Simbol Kebencian

Skema Pemetaan L2

Jika Anda menggunakan logika pasca-pemrosesan yang memfilter pada kategori L1 dan L2 (seperti `Explicit Nudity / Nudity`, `Suggestive / Female Swimwear Or Underwear`, `Violence / Weapon Violence` dll.), Lihat tabel di bawah ini untuk memperbarui kode Anda.

V6.1 L1	V6.1 L2	V7 L1	V7 L2	V7 L3	V7 ContentTypes
Ketelanjangan Eksplisit	Ketelanjangan	Eksplisit	Ketelanjangan Eksplisit	Puting Wanita Terpapar Bokong atau Anus Terpapar	
	Grafik Ketelanjangan Laki-Laki	Eksplisit	Ketelanjangan Eksplisit	Alat Kelamin Pria Terpapar	
	Grafik Ketelanjangan Perempuan	Eksplisit	Ketelanjangan Eksplisit	Alat Kelamin Wanita Terpapar	
	Aktivitas Seksual	Eksplisit	Aktivitas Seksual Eksplisit		
	Gambaran Ketelanjangan Eksplisit	Eksplisit	Ketelanjangan Eksplisit		Peta ke "Animasi" dan "Ilustrasi"
	Gambaran Ketelanjangan Eksplisit	Eksplisit	Aktivitas Seksual Eksplisit		Peta ke "Animasi" dan "Ilustrasi"
	Mainan Dewasa	Eksplisit	Mainan Seks		
Sugestif	Pakaian Renang Atau Pakaian	Pakaian Renang atau Pakaian Dalam	Pakaian Renang atau Pakaian		

Dalam Perempuan		Dalam Wanita	
Pakaian Renang Atau Pakaian Dalam Laki-Laki	Pakaian Renang atau Pakaian Dalam	Pakaian Renang atau Pakaian Dalam Pria	
Ketelanjangan Sebagian	Ketelanjangan Non-Eksplisit dari bagian intim dan ciuman	Ketelanjangan Non-Eksplisit	Ketelanjangan Tersirat
Laki-Laki Bertelanjangan Dada	Ketelanjangan Non-Eksplisit dari bagian intim dan ciuman	Ketelanjangan Non-Eksplisit	Puting Pria Terpapar
Mengungkap Pakaian	Ketelanjangan Non-Eksplisit dari bagian intim dan ciuman	Ketelanjangan Non-Eksplisit	
	Ketelanjangan Non-Eksplisit dari bagian intim dan ciuman	Bagian Intim yang Terhambat	
Situasi Seksual	Ketelanjangan Non-Eksplisit dari bagian intim dan ciuman	Berciuman di Bibir	

Kekerasan	Kekerasan Grafis Atau Gore	Kekerasan	Kekerasan Grafis	Darah & Gore
	Kekerasan Fisik	Kekerasan	Kekerasan Grafis	Kekerasan Fisik
	Kekerasan Senjata	Kekerasan	Kekerasan Grafis	Kekerasan Senjata
	Senjata	Kekerasan	Senjata	
	Melukai Diri Sendiri	Kekerasan	Kekerasan Grafis	Menyakiti Diri Sendiri
Secara Visual Mengganggu	Badan yang Kurus	Secara Visual Mengganggu	Kematian dan Kurus Kurus	Badan yang Kurus
	Mayat	Secara Visual Mengganggu	Kematian dan Kurus Kurus	Mayat
	Menggantung	Secara Visual Mengganggu	Kematian dan Kurus Kurus	Mayat
	Kecelakaan Udara	Secara Visual Mengganggu	Kecelakaan	Kecelakaan Udara
	Ledakan Dan Letusan	Kekerasan	Kekerasan Grafis	Ledakan dan Ledakan
Tindakan Kasar	Jari Tengah	Tindakan Kasar	Jari Tengah	
Narkoba	Produk Narkoba	Obat-obatan & Tembakau	Produk	

	Penggunaan Narkoba	Obat-obatan & Tembakau	Obat-obatan & Perlengkapan Tembakau & Penggunaan	
	Pil	Obat-obatan & Tembakau	Produk	Pil
	Obat Paraphernalia	Obat-obatan & Tembakau	Obat-obatan & Perlengkapan Tembakau & Penggunaan	
Tembakau	Produk Tembakau	Obat-obatan & Tembakau	Produk	
	Merokok	Obat-obatan & Tembakau	Obat-obatan & Perlengkapan Tembakau & Penggunaan	Merokok
Alkohol	Minum	Alkohol	Penggunaan Alkohol	Minum
	Minuman Beralkohol	Alkohol	Minuman Beralkohol	
Judi	Judi	Judi		
Simbol Kebencian	Partai Nazi	Simbol Kebencian	Partai Nazi	
	Supremasi Putih	Simbol Kebencian	Supremasi Putih	

Ekstremis Simbol Ekstremis
 Kebencian

Mendeteksi citra yang tidak pantas

Anda dapat menggunakan operasi [DetectModerationLabel](#) untuk menentukan apakah gambar berisi konten yang tidak pantas atau menyinggung. Untuk daftar label moderasi di Amazon Rekognition, lihat [Menggunakan API moderasi citra dan video](#).

Mendeteksi konten yang tidak pantas dalam gambar

Citra harus dalam format .jpg atau .png. Anda dapat menyediakan citra input sebagai array bit citra (bit citra yang dikodekan base64), atau menentukan objek Amazon S3. Dalam prosedur ini, Anda mengunggah citra (.jpg atau .png) ke bucket S3 Anda.

Untuk menjalankan prosedur ini, Anda harus menginstal AWS CLI atau AWS SDK yang sesuai. Untuk informasi selengkapnya, lihat [Memulai dengan Amazon Rekognition](#). Akun AWS yang Anda gunakan harus memiliki izin akses ke API Amazon Rekognition. Untuk informasi selengkapnya, lihat [Tindakan yang Ditetapkan oleh Amazon Rekognition](#).

Untuk mendeteksi label moderasi dalam citra (SDK)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan AmazonRekognitionFullAccess dan AmazonS3ReadOnlyAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Unggah citra ke bucket S3.

Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

3. Gunakan contoh berikut untuk memanggil operasi DetectModerationLabels.

Java

Contoh output ini mendeteksi nama label konten yang tidak pantas, tingkat kepercayaan, dan label induk untuk label moderasi yang terdeteksi.

Ganti nilai-nilai bucket dan photo dengan nama bucket S3 dan nama file citra yang Anda gunakan pada langkah 2.

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

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.DetectModerationLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectModerationLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.ModerationLabel;
import com.amazonaws.services.rekognition.model.S3Object;

import java.util.List;

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

        AmazonRekognition rekognitionClient =
        AmazonRekognitionClientBuilder.defaultClient();

        DetectModerationLabelsRequest request = new
        DetectModerationLabelsRequest()
            .withImage(new Image().withS3Object(new
        S3Object().withName(photo).withBucket(bucket)))
            .withMinConfidence(60F);
        try
        {
```

```
        DetectModerationLabelsResult result =
    rekognitionClient.detectModerationLabels(request);
    List<ModerationLabel> labels = result.getModerationLabels();
    System.out.println("Detected labels for " + photo);
    for (ModerationLabel label : labels)
    {
        System.out.println("Label: " + label.getName()
            + "\n Confidence: " + label.getConfidence().toString() + "%"
            + "\n Parent:" + label.getParentName());
    }
}
catch (AmazonRekognitionException e)
{
    e.printStackTrace();
}
}
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
//snippet-start:[rekognition.java2.recognize_video_text.import]
//snippet-start:[rekognition.java2.detect_mod_labels.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.ModerationLabel;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
//snippet-end:[rekognition.java2.detect_mod_labels.import]
```



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

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "  <sourceImage>\n\n" +
            "Where:\n" +
            "  sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png). \n\n";

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

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

        detectModLabels(rekClient, sourceImage);
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.detect_mod_labels.main]
    public static void detectModLabels(RekognitionClient rekClient, String
sourceImage) {

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

```

        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
            .image(souImage)
            .minConfidence(60F)
            .build();

        DetectModerationLabelsResponse moderationLabelsResponse =
rekClient.detectModerationLabels(moderationLabelsRequest);
        List<ModerationLabel> labels =
moderationLabelsResponse.moderationLabels();
        System.out.println("Detected labels for image");

        for (ModerationLabel label : labels) {
            System.out.println("Label: " + label.name()
                + "\n Confidence: " + label.confidence().toString() + "%"
                + "\n Parent:" + label.parentName());
        }

    } catch (RekognitionException | FileNotFoundException e) {
        e.printStackTrace();
        System.exit(1);
    }
}
// snippet-end:[rekognition.java2.detect_mod_labels.main]

```

AWS CLI

AWS CLI Perintah ini menampilkan output JSON untuk operasi `detect-moderation-labels` CLI.

Ganti `bucket` dan `input.jpg` dengan nama bucket S3 dan nama file citra yang Anda gunakan pada langkah 2. Ganti nilai `profile_name` dengan nama profil pengembang Anda. Untuk menggunakan adaptor, berikan ARN versi proyek ke parameter. `project-version`

```

aws rekognition detect-moderation-labels --image "{S3Object:{Bucket:<bucket-
name>,Name:<image-name>}}" \
--profile profile-name \
--project-version "ARN"

```

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda alih-alih tanda kutip tunggal dan hindari tanda kutip ganda bagian dalam dengan garis miring terbalik (yaitu \) untuk mengatasi kesalahan parser yang mungkin Anda temui. Sebagai contoh, lihat berikut ini:

```
aws rekognition detect-moderation-labels --image "{\"S3Object\":{\"Bucket\":  
\"bucket-name\"},\"Name\": \"image-name\"}" \  
--profile profile-name
```

Python

Contoh output ini mendeteksi nama label konten yang tidak pantas atau menyinggung, tingkat kepercayaan, dan label induk untuk label konten yang tidak pantas terdeteksi.

Dalam fungsi main, ganti nilai-nilai bucket dan photo dengan nama bucket S3 dan nama file citra yang Anda gunakan pada langkah 2. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/  
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
import boto3  
  
def moderate_image(photo, bucket):  
  
    session = boto3.Session(profile_name='profile-name')  
    client = session.client('rekognition')  
  
    response = client.detect_moderation_labels(Image={'S3Object':  
{'Bucket':bucket, 'Name':photo}})  
  
    print('Detected labels for ' + photo)  
    for label in response['ModerationLabels']:  
        print (label['Name'] + ' : ' + str(label['Confidence']))  
        print (label['ParentName'])  
    return len(response['ModerationLabels'])  
  
def main():  
  
    photo='image-name'
```

```
bucket='bucket-name'  
label_count=moderate_image(photo, bucket)  
print("Labels detected: " + str(label_count))  
  
if __name__ == "__main__":  
    main()
```

.NET

Contoh output ini mendeteksi nama label konten yang tidak pantas atau menyinggung, tingkat kepercayaan, dan label induk untuk label moderasi yang terdeteksi.

Ganti nilai-nilai bucket dan photo dengan nama bucket S3 dan nama file citra yang Anda gunakan pada langkah 2.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/  
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
using System;  
using Amazon.Rekognition;  
using Amazon.Rekognition.Model;  
  
public class DetectModerationLabels  
{  
    public static void Example()  
    {  
        String photo = "input.jpg";  
        String bucket = "bucket";  
  
        AmazonRekognitionClient rekognitionClient = new  
AmazonRekognitionClient();  
  
        DetectModerationLabelsRequest detectModerationLabelsRequest = new  
DetectModerationLabelsRequest()  
        {  
            Image = new Image()  
            {  
                S3Object = new S3Object()  
                {  
                    Name = photo,  
                    Bucket = bucket  
                },  
            },  
        },  
    }  
}
```

```
        },
        MinConfidence = 60F
    };

    try
    {
        DetectModerationLabelsResponse detectModerationLabelsResponse =
rekognitionClient.DetectModerationLabels(detectModerationLabelsRequest);
        Console.WriteLine("Detected labels for " + photo);
        foreach (ModerationLabel label in
detectModerationLabelsResponse.ModerationLabels)
            Console.WriteLine("Label: {0}\n Confidence: {1}\n Parent: {2}",
label.Name, label.Confidence, label.ParentName);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }
}
}
```

DetectModerationLabels permintaan operasi

Input ke `DetectModerationLabels` adalah citra. Dalam contoh input JSON ini, citra sumber dimuat dari bucket Amazon S3. `MinConfidence` adalah kepercayaan minimum yang harus dimiliki Amazon Rekognition Image dalam keakuratan label yang terdeteksi agar dapat dikembalikan dalam respons.

```
{
  "Image": {
    "S3Object": {
      "Bucket": "bucket",
      "Name": "input.jpg"
    }
  },
  "MinConfidence": 60
}
```

DetectModerationLabels respon operasi

DetectModerationLabels dapat mengambil citra input dari bucket S3, atau Anda dapat menyediakannya sebagai bit citra. Contoh berikut adalah respons dari panggilan ke DetectModerationLabels.

Dalam contoh respons JSON berikut, perhatikan hal berikut:

- Informasi Deteksi Citra yang Tidak Pantas – Contoh menunjukkan daftar label untuk konten yang tidak pantas atau menyinggung yang ditemukan dalam citra. Daftar tersebut mencakup label tingkat atas dan setiap label tingkat kedua yang terdeteksi dalam citra.

Label – Setiap label memiliki nama, estimasi kepercayaan bahwa Amazon Rekognition memiliki label akurat, dan nama label induknya. Nama induk untuk label tingkat atas adalah "".

kepercayaan label – Setiap label memiliki nilai kepercayaan antara 0 hingga 100 yang menunjukkan persentase kepercayaan jika Amazon Rekognition memiliki label yang benar. Anda menentukan tingkat kepercayaan yang diperlukan agar label dikembalikan dalam respons dalam permintaan operasi API.

```
{
  "ModerationLabels": [
    {
      "Confidence": 99.44782257080078,
      "Name": "Smoking",
      "ParentName": "Drugs & Tobacco Paraphernalia & Use",
      "TaxonomyLevel": 3
    },
    {
      "Confidence": 99.44782257080078,
      "Name": "Drugs & Tobacco Paraphernalia & Use",
      "ParentName": "Drugs & Tobacco",
      "TaxonomyLevel": 2
    },
    {
      "Confidence": 99.44782257080078,
      "Name": "Drugs & Tobacco",
      "ParentName": "",
      "TaxonomyLevel": 1
    }
  ],
}
```

```
"ModerationModelVersion": "7.0",
"ContentTypes": [
  {
    "Confidence": 99.9999008178711,
    "Name": "Illustrated"
  }
]
}
```

Mendeteksi video tersimpan yang tidak pantas

Konten Amazon Rekognition Video yang tidak pantas atau menyinggung terdeteksi di dalam video yang disimpan adalah operasi tidak sinkron. [Untuk mulai mendeteksi konten yang tidak pantas atau menyinggung, hubungi StartContent Moderasi](#). Amazon Rekognition Video menerbitkan status penyelesaian analisis video ke topik Amazon Simple Notification Service. Jika analisis video berhasil, hubungi [GetContentModerasi](#) untuk mendapatkan hasil analisis. Untuk informasi selengkapnya tentang memulai analisis video dan mendapatkan hasilnya, lihat [Memanggil operasi Amazon Rekognition Video](#). Untuk daftar label moderasi di Amazon Rekognition, lihat [Menggunakan API moderasi citra dan video](#).

Prosedur ini diperluas pada kode di [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#), yang menggunakan antrean Amazon Simple Queue Service untuk mendapatkan status penyelesaian permintaan analisis video.

Untuk mendeteksi konten yang tidak pantas atau menyinggung dalam video yang disimpan dalam bucket Amazon S3 (SDK)

1. Lakukan [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#).
2. Tambahkan kode berikut ke kelas VideoDetect yang Anda buat di langkah 1.

Java

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

```
//Content moderation
```

```
=====
```

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

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

    StartContentModerationRequest req = new
StartContentModerationRequest()
        .withVideo(new Video()
            .withS3Object(new S3Object()
                .withBucket(bucket)
                .withName(video)))
        .withNotificationChannel(channel);

    StartContentModerationResult startModerationLabelDetectionResult =
rek.startContentModeration(req);
    startJobId=startModerationLabelDetectionResult.getJobId();

}

private static void GetUnsafeContentDetectionResults() throws
Exception{

    int maxResults=10;
    String paginationToken=null;
    GetContentModerationResult moderationLabelDetectionResult =null;

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

        moderationLabelDetectionResult = rek.getContentModeration(
            new GetContentModerationRequest()
                .withJobId(startJobId)
                .withNextToken(paginationToken)
                .withSortBy(ContentModerationSortBy.TIMESTAMP)
                .withMaxResults(maxResults));
    }
```



```

        VideoMetadata
videoMetaData=moderationLabelDetectionResult.getVideoMetadata();

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

        //Show moderated content labels, confidence and detection
times
        List<ContentModerationDetection> moderationLabelsInFrames=
            moderationLabelDetectionResult.getModerationLabels();

        for (ContentModerationDetection label:
moderationLabelsInFrames) {
            long seconds=label.getTimestamp()/1000;
            System.out.print("Sec: " + Long.toString(seconds));
            System.out.println(label.getModerationLabel().toString());
            System.out.println();
        }
        } while (moderationLabelDetectionResult !=null &&
moderationLabelDetectionResult.getNextToken() != null);
    }

```

Dalam fungsi main, ganti baris:

```

StartLabelDetection(bucket, video);

if (GetSQSMessagesSuccess()==true)
    GetLabelDetectionResults();

```

dengan:

```

StartUnsafeContentDetection(bucket, video);

if (GetSQSMessagesSuccess()==true)

```

```
GetUnsafeContentDetectionResults();
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import
    software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

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

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>
```

```
        Where:
            bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
            video - The name of video (for example, people.mp4).\s
            topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
            roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
            """;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

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

    startModerationDetection(rekClient, channel, bucket, video);
    getModResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

public static void startModerationDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {

    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
```

```
        .name(video)
        .build();

    Video vid0b = Video.builder()
        .s3Object(s3Obj)
        .build();

    StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
        .jobTag("Moderation")
        .notificationChannel(channel)
        .video(vid0b)
        .build();

    StartContentModerationResponse startModDetectionResult = rekClient
        .startContentModeration(modDetectionRequest);
    startJobId = startModDetectionResult.jobId();

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

public static void getModResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetContentModerationResponse modDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (modDetectionResponse != null)
                paginationToken = modDetectionResponse.nextToken();

            GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
```

```
        while (!finished) {
            modDetectionResponse =
rekClient.getContentModeration(modRequest);
            status = modDetectionResponse.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }

        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is
null.
        VideoMetadata videoMetaData =
modDetectionResponse.videoMetadata();
        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " +
videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());
        System.out.println("Job");

        List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
        for (ContentModerationDetection mod : mods) {
            long seconds = mod.timestamp() / 1000;
            System.out.print("Mod label: " + seconds + " ");
            System.out.println(mod.moderationLabel().toString());
            System.out.println();
        }

    } while (modDetectionResponse != null &&
modDetectionResponse.nextToken() != null);

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

```
}
```

Python

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

# ===== Unsafe content =====
def StartUnsafeContent(self):
    response=self.rek.start_content_moderation(Video={'S3Object': {'Bucket':
self.bucket, 'Name': self.video}},
        NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
self.snsTopicArn})

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

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

    while finished == False:
        response = self.rek.get_content_moderation(JobId=self.startJobId,
                                                    MaxResults=maxResults,
                                                    NextToken=paginationToken,
                                                    SortBy="NAME",
                                                    AggregateBy="TIMESTAMPS")

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

        for contentModerationDetection in response['ModerationLabels']:
            print('Label: ' +
                str(contentModerationDetection['ModerationLabel']['Name']))
            print('Confidence: ' +
                str(contentModerationDetection['ModerationLabel']
['Confidence']))
```

```

        print('Parent category: ' +
              str(contentModerationDetection['ModerationLabel']
                ['ParentName']))
        print('Timestamp: ' +
              str(contentModerationDetection['Timestamp']))
        print()

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

```

Dalam fungsi main, ganti baris:

```

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

```

dengan:

```

analyzer.StartUnsafeContent()
if analyzer.GetSQSMessageSuccess()==True:
    analyzer.GetUnsafeContentResults()

```

Note

Jika Anda sudah menjalankan contoh video selain [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#), kode yang akan diganti mungkin berbeda.

3. Jalankan kode tersebut. Daftar label konten yang tidak pantas yang terdeteksi di video akan ditampilkan.

GetContentModeration respon operasi

Respon dari `GetContentModeration` adalah `arrayModerationLabels`, dari objek [ContentModerationDeteksi](#). Array berisi elemen untuk setiap kali label konten yang tidak pantas terdeteksi. Dalam suatu `ContentModerationDetectionObject` objek, [ModerationLabel](#) berisi

informasi untuk item yang terdeteksi dari konten yang tidak pantas atau menyinggung.

Timestamp adalah waktu, dalam milidetik dari awal video, ketika label terdeteksi. Label disusun secara hierarkis dengan cara yang sama seperti label yang terdeteksi oleh analisis citra konten yang tidak pantas. Untuk informasi selengkapnya, lihat [Memoderasi konten](#).

Berikut ini adalah contoh respons dari `GetContentModeration`, diurutkan berdasarkan NAME dan dikumpulkan berdasarkan. TIMESTAMPS

```
{
  "JobStatus": "SUCCEEDED",
  "VideoMetadata": {
    "Codec": "h264",
    "DurationMillis": 54100,
    "Format": "QuickTime / MOV",
    "FrameRate": 30.0,
    "FrameHeight": 462,
    "FrameWidth": 884,
    "ColorRange": "LIMITED"
  },
  "ModerationLabels": [
    {
      "Timestamp": 36000,
      "ModerationLabel": {
        "Confidence": 52.451576232910156,
        "Name": "Alcohol",
        "ParentName": "",
        "TaxonomyLevel": 1
      },
      "ContentTypes": [
        {
          "Confidence": 99.9999008178711,
          "Name": "Animated"
        }
      ]
    },
    {
      "Timestamp": 36000,
      "ModerationLabel": {
        "Confidence": 52.451576232910156,
        "Name": "Alcoholic Beverages",
        "ParentName": "Alcohol",
        "TaxonomyLevel": 2
      }
    }
  ]
}
```



```

        "ContentTypes": [
            {
                "Confidence": 99.9999008178711,
                "Name": "Animated"
            }
        ]
    },
    ],
    "ModerationModelVersion": "7.0",
    "JobId": "a1b2c3d4...",
    "Video": {
        "S3Object": {
            "Bucket": "bucket-name",
            "Name": "video-name.mp4"
        }
    },
    "GetRequestMetadata": {
        "SortBy": "TIMESTAMP",
        "AggregateBy": "TIMESTAMPS"
    }
}

```

Berikut ini adalah contoh respons dari `getContentModeration`, diurutkan berdasarkan NAME dan dikumpulkan berdasarkan SEGMENTS

```

{
    "JobStatus": "SUCCEEDED",
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 54100,
        "Format": "QuickTime / MOV",
        "FrameRate": 30.0,
        "FrameHeight": 462,
        "FrameWidth": 884,
        "ColorRange": "LIMITED"
    },
    "ModerationLabels": [
        {
            "Timestamp": 0,
            "ModerationLabel": {
                "Confidence": 0.0003000000142492354,
                "Name": "Alcohol Use",
                "ParentName": "Alcohol",
            }
        }
    ]
}

```

```
        "TaxonomyLevel": 2
    },
    "StartTimestampMillis": 0,
    "EndTimestampMillis": 29520,
    "DurationMillis": 29520,
    "ContentTypes": [
        {
            "Confidence": 99.9999008178711,
            "Name": "Illustrated"
        },
        {
            "Confidence": 99.9999008178711,
            "Name": "Animated"
        }
    ]
}
],
"ModerationModelVersion": "7.0",
"JobId": "a1b2c3d4...",
"Video": {
    "S3Object": {
        "Bucket": "bucket-name",
        "Name": "video-name.mp4"
    }
},
"GetRequestMetadata": {
    "SortBy": "TIMESTAMP",
    "AggregateBy": "SEGMENTS"
}
}
```

Meningkatkan akurasi dengan Custom Moderation

Amazon Rekognition's [DetectModerationLabels](#) API memungkinkan Anda mendeteksi konten yang tidak pantas, tidak diinginkan, atau menyinggung. Fitur Moderasi Kustom Rekognition memungkinkan Anda meningkatkan akurasi Label dengan menggunakan adaptor [DetectModeration](#). Adaptor adalah komponen modular yang dapat ditambahkan ke model pembelajaran mendalam Rekognition yang ada, memperluas kemampuannya untuk tugas-tugas yang dilatihnya. Dengan membuat adaptor dan menyediakannya ke operasi [DetectModerationLabel](#), Anda dapat mencapai akurasi yang lebih baik untuk tugas moderasi konten yang terkait dengan kasus penggunaan spesifik Anda.

Saat menyesuaikan model moderasi konten Rekognition untuk label moderasi tertentu, Anda harus membuat proyek dan melatih adaptor pada sekumpulan gambar yang Anda berikan. Anda kemudian dapat memeriksa kinerja adaptor secara berulang dan melatih kembali adaptor ke tingkat akurasi yang Anda inginkan. Proyek digunakan untuk memuat versi adaptor yang berbeda.

Anda dapat menggunakan konsol Rekognition untuk membuat proyek dan adaptor. Atau, Anda dapat menggunakan AWS SDK dan API terkait untuk membuat proyek, melatih adaptor, dan mengelola adaptor Anda.

Membuat dan menggunakan adaptor

Adaptor adalah komponen modular yang dapat ditambahkan ke model pembelajaran mendalam Rekognition yang ada, memperluas kemampuannya untuk tugas-tugas yang dilatihnya. Dengan melatih model pembelajaran mendalam dengan adaptor, Anda dapat mencapai akurasi yang lebih baik untuk tugas analisis gambar yang terkait dengan kasus penggunaan spesifik Anda.

Untuk membuat dan menggunakan adaptor, Anda harus memberikan pelatihan dan pengujian data untuk Rekognition. Anda dapat melakukannya dengan salah satu dari dua cara berbeda:

- Analisis dan verifikasi massal - Anda dapat membuat kumpulan data pelatihan dengan menganalisis gambar massal yang akan dianalisis dan ditetapkan oleh Rekognition. Anda kemudian dapat meninjau anotasi yang dihasilkan untuk gambar Anda dan memverifikasi atau memperbaiki prediksi. Untuk informasi selengkapnya tentang cara kerja analisis massal gambar, lihat [Analisis massal](#).
- Anotasi manual - Dengan pendekatan ini Anda membuat data pelatihan Anda dengan mengunggah dan membuat anotasi gambar. Anda membuat data pengujian dengan mengunggah dan membuat anotasi gambar atau dengan pemisahan otomatis.

Pilih salah satu topik berikut untuk mempelajari lebih lanjut:

Topik

- [Analisis dan verifikasi massal](#)
- [Anotasi manual](#)

Analisis dan verifikasi massal

Dengan pendekatan ini, Anda mengunggah sejumlah besar gambar yang ingin Anda gunakan sebagai data pelatihan dan kemudian menggunakan Rekognition untuk mendapatkan prediksi untuk gambar-gambar ini, yang secara otomatis memberikan label kepada mereka. Anda dapat menggunakan prediksi ini sebagai titik awal untuk adaptor Anda. Anda dapat memverifikasi keakuratan prediksi, dan kemudian melatih adaptor berdasarkan prediksi yang diverifikasi. Ini bisa dilakukan dengan AWS konsol.

[Analisis Massal dan Moderasi Kustom](#)

Unggah gambar untuk analisis massal

Untuk membuat kumpulan data pelatihan untuk adaptor Anda, unggah gambar secara massal untuk Rekognition untuk memprediksi label. Untuk hasil terbaik, berikan sebanyak mungkin gambar untuk pelatihan hingga batas 10.000, dan pastikan gambar mewakili semua aspek kasus penggunaan Anda.

Saat menggunakan AWS Konsol, Anda dapat mengunggah gambar langsung dari komputer atau menyediakan bucket Amazon Simple Storage Service yang menyimpan gambar Anda. Namun, saat menggunakan API Rekognition dengan SDK, Anda harus menyediakan file manifes yang mereferensikan gambar yang disimpan dalam bucket Amazon Simple Storage Service. Lihat [Analisis massal](#) untuk informasi lebih lanjut.

Tinjau prediksi

Setelah Anda mengunggah gambar Anda ke konsol Rekognition, Rekognition akan menghasilkan label untuk mereka. Anda kemudian dapat memverifikasi prediksi sebagai salah satu kategori berikut: benar positif, positif palsu, negatif benar, negatif palsu. Setelah Anda memverifikasi prediksi, Anda dapat melatih adaptor pada umpan balik Anda.

Latih adaptor

Setelah Anda selesai memverifikasi prediksi yang dikembalikan oleh analisis massal, Anda dapat memulai proses pelatihan untuk adaptor Anda.

Dapatkan AdapterId

Setelah adaptor dilatih, Anda bisa mendapatkan ID unik untuk adaptor Anda untuk digunakan dengan API analisis gambar Rekognition.

Panggil Operasi API

Untuk menerapkan adaptor kustom Anda, berikan ID-nya saat memanggil salah satu API analisis gambar yang mendukung adaptor. Ini meningkatkan akurasi prediksi untuk gambar Anda.

Anotasi manual

Dengan pendekatan ini, Anda membuat data pelatihan dengan mengunggah dan membuat anotasi gambar secara manual. Anda membuat data pengujian dengan mengunggah dan membuat anotasi gambar pengujian atau dengan memisahkan otomatis agar Rekognition secara otomatis menggunakan sebagian data pelatihan Anda sebagai gambar pengujian.

Mengunggah dan membuat anotasi gambar

Untuk melatih adaptor, Anda harus mengunggah satu set gambar sampel yang mewakili kasus penggunaan Anda. Untuk hasil terbaik, berikan sebanyak mungkin gambar untuk pelatihan hingga batas 10.000, dan pastikan gambar mewakili semua aspek kasus penggunaan Anda.

Training images [Info](#)

Import training image dataset
Import your image dataset from one of the below sources. To improve accuracy for a label: 20 images required to improve false-positives, 50 images required improve false-negatives. More images result in higher accuracy.

Import a manifest file
Labels must adhere to the Content moderation label categories, otherwise you will need to reassign labels in the next step.

Import images from S3 bucket
Import new images using a link to an S3 bucket.

Upload images from your computer
Upload 50 images at one time from your computer.

S3 URI

Supported formats: json

Be sure users have read and write permissions for the data location.

Test images [Info](#)

Saat menggunakan AWS Konsol, Anda dapat mengunggah gambar langsung dari komputer, menyediakan file manifes, atau menyediakan bucket Amazon S3 yang menyimpan gambar Anda.

Namun, saat menggunakan API Rekognition dengan SDK, Anda harus menyediakan file manifes yang mereferensikan gambar yang disimpan dalam bucket Amazon S3.

Anda dapat menggunakan antarmuka anotasi konsol [Rekognition](#) untuk membuat anotasi gambar Anda. Beri anotasi gambar Anda dengan menandainya dengan label, ini menetapkan “kebenaran dasar” untuk pelatihan. Anda juga harus menetapkan set pelatihan dan pengujian, atau menggunakan fitur auto-split, sebelum Anda dapat melatih adaptor. Setelah selesai menentukan kumpulan data dan membuat anotasi gambar, Anda dapat membuat adaptor berdasarkan gambar beranotasi di set pengujian Anda. Anda kemudian dapat mengevaluasi kinerja adaptor Anda.

Buat set tes

Anda harus menyediakan set pengujian beranotasi atau menggunakan fitur auto-split. Set pelatihan digunakan untuk benar-benar melatih adaptor. Adaptor mempelajari pola yang terkandung dalam gambar beranotasi ini. Set tes digunakan untuk mengevaluasi kinerja model sebelum menyelesaikan adaptor.

Latih adaptor

Setelah selesai membuat anotasi data pelatihan, atau telah menyediakan file manifes, Anda dapat memulai proses pelatihan untuk adaptor Anda.

Dapatkan ID Adaptor

Setelah adaptor dilatih, Anda bisa mendapatkan ID unik untuk adaptor Anda untuk digunakan dengan API analisis gambar Rekognition.

Panggil operasi API

Untuk menerapkan adaptor kustom Anda, berikan ID-nya saat memanggil salah satu API analisis gambar yang mendukung adaptor. Ini meningkatkan akurasi prediksi untuk gambar Anda.

Mempersiapkan kumpulan data Anda

Membuat adaptor mengharuskan Anda menyediakan Rekognition dengan dua kumpulan data, kumpulan data pelatihan, dan kumpulan data pengujian. Setiap dataset terdiri dari menyediakan dua elemen: gambar dan anotasi/label. Bagian berikut menjelaskan label dan gambar apa yang digunakan dan bagaimana mereka bersatu untuk membuat kumpulan data.

Citra

Anda perlu melatih adaptor pada sampel representatif gambar Anda. Saat Anda memilih gambar untuk pelatihan, cobalah untuk menyertakan setidaknya beberapa gambar yang menunjukkan respons yang diharapkan untuk setiap label yang Anda targetkan dengan adaptor Anda.

Untuk membuat kumpulan data pelatihan, Anda perlu menyediakan salah satu dari dua jenis gambar berikut:

- Gambar dengan prediksi Positif Palsu. Misalnya, ketika model dasar memprediksi bahwa gambar mengandung alkohol, tetapi ternyata tidak.
- Gambar dengan prediksi Negatif Palsu. Misalnya, ketika model dasar memprediksi bahwa gambar tidak mengandung alkohol, tetapi memang demikian.

Untuk membuat kumpulan data seimbang, Anda disarankan untuk menyediakan salah satu dari dua jenis gambar berikut:

- Gambar dengan prediksi Positif Sejati. Misalnya, ketika model dasar memprediksi dengan benar bahwa suatu gambar mengandung alkohol. Disarankan untuk memberikan gambar-gambar ini jika Anda memberikan gambar Positif Palsu.
- Gambar dengan prediksi Negatif Sejati. Misalnya, ketika model dasar memprediksi dengan benar bahwa gambar tidak mengandung alkohol. Disarankan untuk memberikan gambar-gambar ini jika Anda memberikan gambar Negatif Palsu.

Label

Label mengacu pada salah satu dari yang berikut: objek, peristiwa, konsep atau kegiatan. Untuk Moderasi Konten, label adalah contoh konten yang tidak pantas, tidak diinginkan, atau menyinggung.

Dalam konteks membuat adaptor dengan melatih model dasar Rekognition, ketika label ditetapkan ke gambar, itu disebut anotasi. Saat melatih adaptor dengan Konsol Rekognition, Anda akan menggunakan Konsol untuk menambahkan anotasi ke gambar Anda dengan memilih label dan kemudian menandai gambar yang sesuai dengan label. Melalui proses ini, model belajar mengidentifikasi elemen gambar Anda berdasarkan label yang ditetapkan. Proses penautan ini memungkinkan model untuk fokus pada konten yang paling relevan saat adaptor dibuat, yang mengarah pada peningkatan akurasi untuk analisis gambar.

Atau, Anda dapat menyediakan file manifes, yang berisi informasi tentang gambar dan anotasi yang menyertainya.

Kumpulan data pelatihan dan pengujian

Dataset pelatihan adalah dasar untuk menyempurnakan model dan membuat adaptor khusus. Anda harus menyediakan kumpulan data pelatihan beranotasi agar model dapat dipelajari. Model belajar dari kumpulan data ini untuk meningkatkan kinerjanya pada jenis gambar yang Anda berikan.

Untuk meningkatkan akurasi, Anda harus membuat kumpulan data pelatihan dengan anotasi/ pelabelan gambar. Anda dapat melakukan ini menggunakan dua cara:

- Penetapan label manual - Anda dapat menggunakan Konsol Rekognition untuk membuat kumpulan data pelatihan dengan mengunggah gambar yang ingin berisi kumpulan data Anda dan kemudian menetapkan label secara manual ke gambar ini.
- File manifes - Anda dapat menggunakan file manifes untuk melatih adaptor Anda. File manifes berisi informasi tentang anotasi kebenaran dasar untuk gambar pelatihan dan pengujian Anda, serta lokasi gambar pelatihan Anda. Anda dapat memberikan file manifes saat melatih adaptor menggunakan API Rekognition atau saat menggunakan Konsol. AWS

Dataset pengujian digunakan untuk mengevaluasi kinerja adaptor setelah pelatihan. Untuk memastikan evaluasi yang andal, kumpulan data pengujian dibuat dengan menggunakan sepotong kumpulan data pelatihan asli yang belum pernah dilihat model sebelumnya. Proses ini memastikan bahwa kinerja adaptor dinilai dengan data baru, menciptakan pengukuran dan metrik yang akurat. Untuk peningkatan akurasi yang optimal, lihat [Praktik terbaik untuk adaptor pelatihan](#).

Mengelola adaptor dengan AWS CLI dan SDK

Rekognition memungkinkan Anda memanfaatkan beberapa fitur yang memanfaatkan model visi komputer yang telah dilatih sebelumnya. Dengan model ini Anda dapat melakukan tugas-tugas seperti deteksi label dan moderasi konten. Anda juga dapat menyesuaikan model-model tertentu ini menggunakan adaptor.

Anda dapat menggunakan pembuatan proyek Rekognition dan API manajemen proyek (seperti [CreateProject](#) dan [CreateProjectVersi](#)) untuk membuat dan melatih adaptor. Halaman berikut menjelaskan cara menggunakan operasi API untuk membuat, melatih, dan mengelola adaptor Anda, menggunakan AWS Konsol, AWS SDK pilihan Anda, atau AWS CLI.

Setelah Anda melatih adaptor, Anda dapat menggunakannya saat menjalankan inferensi dengan fitur yang didukung. Saat ini, adaptor didukung saat menggunakan fitur Moderasi Konten.

Saat melatih adaptor menggunakan AWS SDK, Anda harus memberikan label kebenaran dasar (anotasi gambar) dalam bentuk file manifes. Atau, Anda dapat menggunakan Konsol Rekognition untuk membuat dan melatih adaptor.

Note

Adaptor tidak dapat disalin. Hanya versi proyek Rekognition Custom Labels yang dapat disalin.

Topik

- [Status adaptor](#)
- [Membuat proyek](#)
- [Menggambarkan proyek](#)
- [Menghapus proyek](#)
- [Membuat versi proyek](#)
- [Menjelaskan versi proyek](#)
- [Menghapus versi proyek](#)

Status adaptor

Adaptor Moderasi Kustom (versi proyek) dapat berada di salah satu status berikut:

- **TRAINING_IN_PROGRESS** - Adaptor sedang dalam proses pelatihan pada file yang Anda berikan sebagai dokumen pelatihan.
- **TRAINING_COMPLETED** - Adaptor telah berhasil menyelesaikan pelatihan dan siap bagi Anda untuk meninjau kinerjanya.
- **TRAINING_FAILED** - Adaptor gagal menyelesaikan pelatihannya karena beberapa alasan, meninjau file manifes keluaran dan ringkasan manifes keluaran untuk informasi tentang penyebab kegagalan.
- **MENGHAPUS** - Adaptor sedang dalam proses dihapus.
- **USANG** - Adaptor dilatih pada versi lama dari model dasar Moderasi Konten. Ini dalam masa tenggang dan akan kedaluwarsa dalam waktu 60 hingga 90 hari setelah rilis versi model dasar

baru. Selama masa tenggang, Anda masih dapat menggunakan adaptor untuk inferensi dengan operasi [DetectModerationLabel](#) atau [StartMediaAnalysisJob](#) API. Lihat Konsol Moderasi Kustom untuk tanggal kedaluwarsa adaptor Anda.

- EXPIRED - Adaptor dilatih pada versi lama dari model dasar Moderasi Konten dan tidak dapat lagi digunakan untuk mendapatkan hasil kustom dengan operasi `DetectModerationLabels` atau `StartMediaAnalysisJob` API. Jika adaptor kedaluwarsa ditentukan dalam permintaan inferensi, adaptor tersebut akan diabaikan dan respons dikembalikan dari versi terbaru model dasar Moderasi Kustom.

Membuat proyek

Dengan [CreateProject](#) operasi ini, Anda dapat membuat proyek yang akan menampung adaptor untuk operasi deteksi label Rekognition. Proyek adalah sekelompok sumber daya dan dalam kasus operasi deteksi label seperti `DetectModerationLabels`, proyek memungkinkan Anda untuk menyimpan adaptor yang dapat Anda gunakan untuk menyesuaikan model Rekognition dasar. Saat memanggil `CreateProject`, Anda memberikan nama proyek yang ingin Anda buat untuk `ProjectName` argumen.

Untuk membuat proyek dengan AWS konsol:

- Masuk ke Konsol Rekognition
- Klik pada Custom Moderation
- Pilih Buat Proyek
- Pilih Buat Proyek Baru atau Tambahkan ke proyek yang sudah ada
- Tambahkan nama Proyek
- Tambahkan nama Adaptor
- Tambahkan deskripsi jika diinginkan
- Pilih bagaimana Anda ingin mengimpor gambar pelatihan Anda: File manifes, dari bucket S3, atau dari komputer Anda
- Pilih apakah Anda ingin Autosplit data pelatihan Anda atau impor file manifes
- Pilih apakah Anda ingin proyek diperbarui secara otomatis atau tidak
- Klik Buat proyek

Untuk membuat proyek dengan AWS CLI dan SDK:

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan SDK. AWS Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan kode berikut untuk membuat proyek:

CLI

```
# Request
# Creating Content Moderation Project
aws rekognition create-project \
  --project-name "project-name" \
  --feature CONTENT_MODERATION \
  --auto-update ENABLED
  --profile profile-name
```

Menggambarkan proyek

Anda dapat menggunakan [DescribeProjects](#) API untuk mendapatkan informasi tentang proyek Anda, termasuk informasi tentang semua adaptor yang terkait dengan proyek.

Untuk mendeskripsikan proyek dengan AWS CLI dan SDK:

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan SDK. AWS Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan kode berikut untuk mendeskripsikan proyek:

CLI

```
# Request
# Getting CONTENT_MODERATION project details
aws rekognition describe-projects \
  --features CONTENT_MODERATION
  --profile profile-name
```

Menghapus proyek

Anda dapat menghapus proyek dengan menggunakan konsol Rekognition atau dengan memanggil API. [DeleteProject](#) Untuk menghapus proyek, Anda harus terlebih dahulu menghapus masing-masing adaptor terkait. Proyek atau model yang dihapus tidak dapat dihapus.

Untuk menghapus proyek dengan AWS konsol:

- Masuk ke Konsol Rekognition.
- Klik pada Custom Moderation.
- Anda harus menghapus setiap adaptor yang terkait dengan proyek Anda sebelum Anda dapat menghapus proyek itu sendiri. Hapus adaptor apa pun yang terkait dengan proyek dengan memilih adaptor dan kemudian memilih Hapus.
- Pilih proyek dan kemudian pilih tombol Hapus.

Untuk menghapus proyek dengan AWS CLI dan SDK:

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan SDK. AWS Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan kode berikut untuk menghapus proyek:

CLI

```
aws rekognition delete-project
  --project-arn project_arn \
  --profile profile-name
```

Membuat versi proyek

Anda dapat melatih adaptor untuk penyebaran dengan menggunakan operasi [CreateProjectVersi](#). `CreateProjectVersion` pertama membuat versi baru dari adaptor yang terkait dengan proyek dan kemudian mulai melatih adaptor. Tanggapan dari `CreateProjectVersion` adalah Amazon Resource Name (ARN) untuk versi model. Membutuhkan waktu beberapa saat untuk menyelesaikan pelatihan. Anda bisa mendapatkan status saat ini dengan menelepon `DescribeProjectVersions`. Saat melatih model, Rekognition menggunakan kumpulan data pelatihan dan pengujian yang terkait dengan

proyek. Anda membuat kumpulan data menggunakan konsol. Untuk informasi selengkapnya, lihat bagian tentang kumpulan data.

Untuk membuat versi proyek dengan konsol Rekognition:

- Masuk ke AWS Rekognition Console
- Klik pada Custom Moderation
- Pilih proyek.
- Pada halaman “Detail proyek”, pilih Buat adaptor
- Pada halaman “Buat proyek”, isi detail yang diperlukan untuk Detail Proyek, gambar Pelatihan, dan gambar Pengujian, lalu pilih Buat proyek.
- Pada halaman “Tetapkan label ke gambar”, tambahkan label ke gambar Anda dan setelah selesai pilih Mulai pelatihan

Untuk membuat versi proyek dengan AWS CLI dan SDK:

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan SDK. AWS Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan kode berikut untuk membuat versi proyek:

CLI

```
# Request
aws rekognition create-project-version \
  --project-arn project-arn \
  --training-data '{Assets=[GroundTruthManifest={S3Object="my-  
bucket",Name="manifest.json"}]}' \
  --output-config S3Bucket=my-output-bucket,S3KeyPrefix=my-results \
  --feature-config "ContentModeration={ConfidenceThreshold=70}"
  --profile profile-name
```

Menjelaskan versi proyek

Anda dapat membuat daftar dan menjelaskan adaptor yang terkait dengan proyek dengan menggunakan operasi [DescribeProjectVersi](#). Anda dapat menentukan hingga 10 versi model di

ProjectVersionArns. Jika Anda tidak menentukan nilai, deskripsi untuk semua versi model dalam proyek akan dikembalikan.

Untuk mendeskripsikan versi proyek dengan AWS CLI dan SDK:

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan SDK. AWS Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Gunakan kode berikut untuk mendeskripsikan versi proyek:

CLI

```
aws rekognition describe-project-versions
--project-arn project_arn \
--version-names [versions]
```

Menghapus versi proyek

[Anda dapat menghapus adaptor Rekognition yang terkait dengan proyek menggunakan DeleteProject operasi Versi](#). Anda tidak dapat menghapus adaptor jika sedang berjalan atau jika sedang dilatih. Untuk memeriksa status adaptor, panggil DescribeProjectVersions operasi dan periksa bidang Status yang dikembalikan olehnya. Untuk menghentikan panggilan adaptor yang sedang berjalan StopProjectVersion. Jika model sedang dilatih, tunggu sampai selesai pelatihan untuk menghapusnya. Anda harus menghapus setiap adaptor yang terkait dengan proyek Anda sebelum Anda dapat menghapus proyek itu sendiri.

Untuk menghapus versi proyek dengan konsol Rekognition:

- Masuk ke Konsol Rekognition
- Klik pada Custom Moderation
- Dari tab Proyek, Anda dapat melihat semua proyek dan adaptor terkait. Pilih adaptor dan kemudian pilih Hapus.

Untuk menghapus versi proyek dengan AWS CLI dan SDK:

1. Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan SDK. AWS Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).

2. Gunakan kode berikut untuk menghapus versi proyek:

CLI

```
# Request
aws rekognition delete-project-version
  --project-version-arn model_arn \
  --profile profile-name
```

Tutorial adaptor Moderasi Kustom

Tutorial ini menunjukkan cara membuat, melatih, mengevaluasi, menggunakan, dan mengelola adaptor menggunakan Konsol Rekognition. Untuk membuat, menggunakan, dan mengelola adaptor dengan AWS SDK, lihat [Mengelola adaptor dengan AWS CLI dan SDK](#).

Adaptor memungkinkan Anda meningkatkan akurasi operasi API Rekognition, menyesuaikan perilaku model agar sesuai dengan kebutuhan dan kasus penggunaan Anda sendiri. Setelah Anda membuat adaptor dengan tutorial ini, Anda akan dapat menggunakannya saat menganalisis gambar Anda sendiri dengan operasi seperti [DetectModerationLabel](#), serta melatih kembali adaptor untuk perbaikan masa depan lebih lanjut.

Dalam tutorial ini Anda akan belajar bagaimana untuk:

- Buat proyek menggunakan Konsol Rekognition
- Beri anotasi data pelatihan Anda
- Latih adaptor Anda pada dataset pelatihan Anda
- Tinjau kinerja adaptor Anda
- Gunakan adaptor Anda untuk analisis gambar

Prasyarat

Sebelum menyelesaikan tutorial ini, Anda disarankan untuk membacanya [Membuat dan menggunakan adaptor](#).

Untuk membuat adaptor, Anda dapat menggunakan alat Konsol Rekognition untuk membuat proyek, mengunggah dan membubuhi keterangan gambar Anda sendiri, lalu melatih adaptor pada gambar-gambar ini. Lihat [Membuat proyek dan melatih adaptor](#) untuk memulai.

Atau, Anda dapat menggunakan Konsol atau API Rekognition untuk mengambil prediksi gambar dan kemudian memverifikasi prediksi sebelum melatih adaptor pada prediksi ini. Lihat [Analisis massal, verifikasi prediksi, dan pelatihan adaptor](#) untuk memulai.

Anotasi gambar

Anda dapat membuat anotasi gambar sendiri dengan memberi label pada gambar dengan konsol Rekognition, atau menggunakan analisis Rekognition Bulk untuk membuat anotasi gambar yang kemudian dapat Anda verifikasi telah diberi label dengan benar. Pilih salah satu topik di bawah ini untuk memulai.

Topik

- [Membuat proyek dan melatih adaptor](#)
- [Analisis massal, verifikasi prediksi, dan pelatihan adaptor](#)

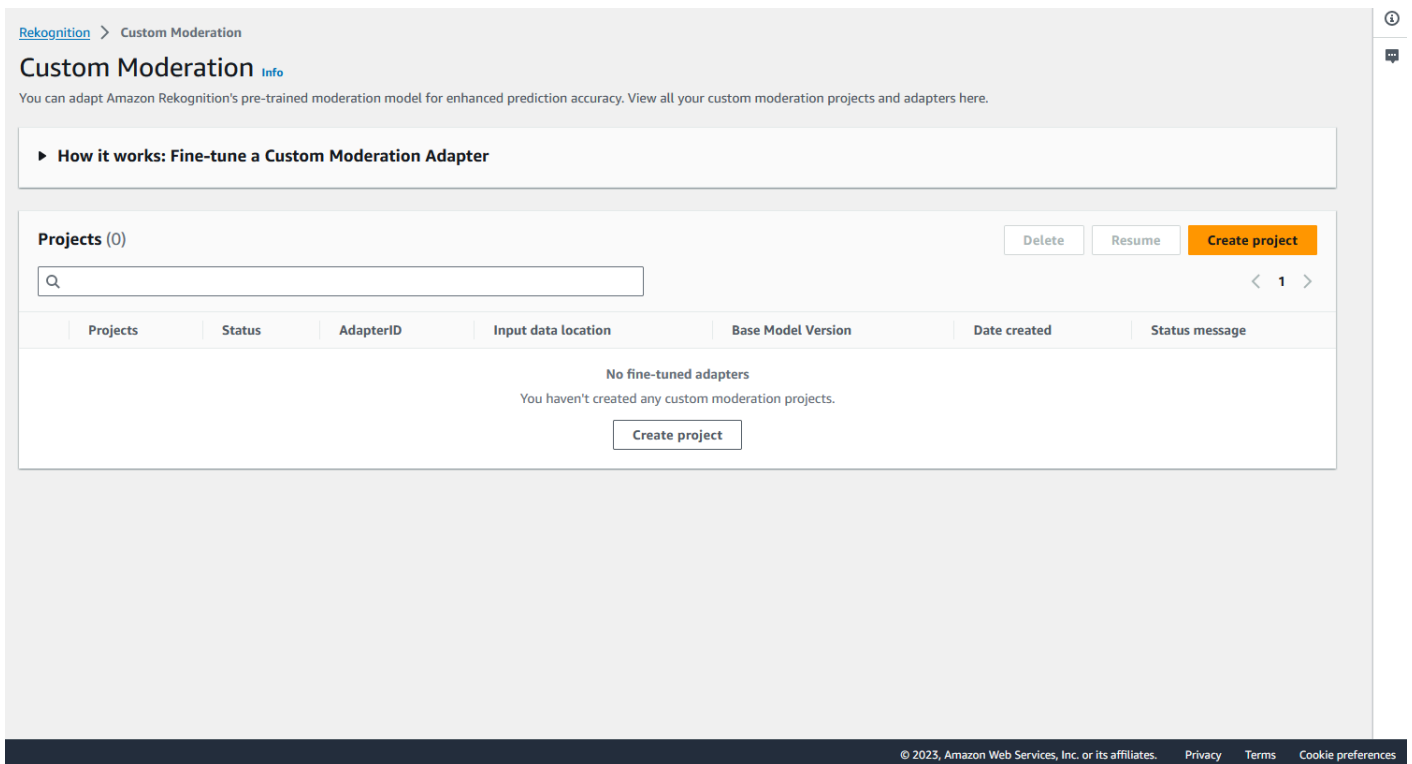
Membuat proyek dan melatih adaptor

Selesaikan langkah-langkah berikut untuk melatih adaptor Anda dengan membuat anotasi gambar menggunakan konsol Rekognition.

Buat proyek

Sebelum Anda dapat melatih atau menggunakan adaptor, Anda harus membuat proyek yang akan berisi itu. Anda juga harus memberikan gambar yang digunakan untuk melatih adaptor Anda. Untuk membuat proyek, adaptor, dan kumpulan data gambar Anda:

1. Masuk ke Konsol AWS Manajemen dan buka konsol Rekognition di <https://console.aws.amazon.com/rekognition/>.
2. Di panel kiri, pilih Moderasi Kustom. Halaman arahan Rekognition Custom Moderation ditampilkan.



Rekognition > Custom Moderation

Custom Moderation Info

You can adapt Amazon Rekognition's pre-trained moderation model for enhanced prediction accuracy. View all your custom moderation projects and adapters here.

► How it works: [Fine-tune a Custom Moderation Adapter](#)

Projects (0) Delete Resume Create project

 < 1 >

Projects	Status	AdapterID	Input data location	Base Model Version	Date created	Status message
<p>No fine-tuned adapters</p> <p>You haven't created any custom moderation projects.</p> <p>Create project</p>						

© 2023, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

- Halaman landing Custom Moderation menunjukkan kepada Anda daftar semua proyek dan adaptor Anda, dan ada juga tombol untuk membuat adaptor. Pilih Buat proyek untuk membuat proyek dan adaptor baru.
- Jika ini adalah pertama kalinya Anda membuat adaptor, Anda akan diminta untuk membuat bucket Amazon S3 untuk menyimpan file yang terkait dengan proyek dan adaptor Anda. Pilih Buat ember Amazon S3.
- Pada halaman berikut, isi nama adaptor dan nama proyek. Berikan deskripsi adaptor jika Anda mau.

Project details

Project name

Name the project that groups your adapters

Project name limited to 255 alphanumeric characters, no spaces or special characters.

Adapter name - Provide a name for the adapter

Adapter name limited to 255 alphanumeric characters, no spaces or special characters.

Adapter description - optional

Enter a description for quick reference

The adapter description can have up to 255 characters.

Training images [Info](#)

Import training image dataset

Import your image dataset from one of the below sources. To improve accuracy for a label: 20 images required to improve false-positives, 50 images required improve false-negatives. More images result in higher accuracy.

Import a manifest file

If you have a labeled dataset in a different format, convert them to a manifest format.

Labels must adhere to the [Content moderation label categories](#), otherwise you will need to reassign labels in the next step.

Import images from S3 bucket

Import new images using a link to an S3 bucket

6. Pada langkah ini, Anda juga akan memberikan gambar untuk adaptor Anda. Anda dapat memilih: Impor gambar dari komputer, Impor file manifes, atau Impor gambar dari bucket Amazon S3. Jika Anda memilih untuk mengimpor gambar dari bucket Amazon S3, berikan jalur ke bucket dan folder yang berisi gambar pelatihan Anda. Jika Anda mengunggah gambar langsung dari komputer, perhatikan bahwa Anda hanya dapat mengunggah hingga 30 gambar sekaligus. Jika Anda menggunakan file manifes yang berisi anotasi, Anda dapat melewati langkah-langkah yang tercantum di bawah ini yang mencakup anotasi gambar dan melanjutkan ke bagian pada [Meninjau kinerja adaptor](#)

- Di bagian Uji detail kumpulan data, pilih Autosplit agar Rekognition secara otomatis memilih persentase gambar yang sesuai sebagai data pengujian, atau Anda dapat memilih Mengimpor file manifes secara manual.
- Setelah mengisi informasi ini, pilih Buat Proyek.

Latih adaptor

Untuk melatih adaptor pada gambar Anda sendiri yang tidak diberi anotasi:

- Pilih proyek yang berisi adaptor Anda dan kemudian pilih opsi untuk Tetapkan label ke gambar.
- Pada halaman Tetapkan label ke gambar, Anda dapat melihat semua gambar yang telah diunggah sebagai gambar pelatihan. Anda dapat memfilter gambar-gambar ini berdasarkan status berlabel/ tidak berlabel dan berdasarkan kategori label menggunakan dua panel pemilihan atribut di sebelah kiri. Anda dapat menambahkan gambar tambahan ke kumpulan data pelatihan Anda dengan memilih tombol Tambahkan gambar.

Rekognition > Custom Moderation > NewTest1 > Assign labels to images


Assign labels to images [Info](#)

Save Draft (0) Delete draft Start fine-tuning

▼ Adapter details


Fine-tuned adapter name	Base Model Version	Data Location
NewAdapter1	Content Moderation v6.1	S3 bucket ↗

▼ How it works: Assign labels to images to create custom moderation adapter




1. Assign ground truth labels to images

To improve accuracy for a label: Assign label to at least 20 images to improve false-positives, 50 images to improve false-negatives.



2. Fine-tune the model and assess performance

Create an adapter (a fine-tuned model). Wait for fine-tuning to complete, then review the adapter's predictions to assess performance and correct errors.



3. Use your adapter

Use the AdapterID of your adapter when doing inference with the DetectModerationLabels API

Filters [Info](#)

Images (0) Add Images Assign labels to images ▼

Select all images on this page

< 1 >

- Setelah menambahkan gambar ke kumpulan data pelatihan, Anda harus membuat anotasi gambar Anda dengan label. Setelah mengunggah gambar Anda, halaman “Tetapkan label ke gambar” akan diperbarui untuk menampilkan gambar yang telah Anda unggah. Anda akan diminta untuk

memilih label yang sesuai untuk gambar Anda dari daftar drop-down label yang didukung oleh Moderasi Rekognition. Anda dapat memilih lebih dari satu label.

- Lanjutkan proses ini sampai Anda menambahkan label ke setiap gambar dalam data pelatihan Anda.
- Setelah Anda memberi label pada semua data Anda, pilih Mulai pelatihan untuk mulai melatih model, yang membuat adaptor Anda.

- Sebelum memulai proses pelatihan, Anda dapat menambahkan Tag apa pun ke adaptor yang Anda inginkan. Anda juga dapat memberikan adaptor dengan kunci enkripsi khusus atau menggunakan kunci AWS KMS. Setelah Anda selesai menambahkan tag apa pun yang Anda inginkan dan menyesuaikan enkripsi sesuai keinginan Anda, pilih Train adapter untuk memulai proses pelatihan untuk adaptor Anda.
- Tunggu adaptor Anda menyelesaikan pelatihan. Setelah pelatihan selesai, Anda akan menerima pemberitahuan bahwa adaptor Anda telah selesai dibuat.

Setelah status adaptor Anda “Pelatihan selesai”, Anda dapat meninjau metrik adaptor Anda

Analisis massal, verifikasi prediksi, dan pelatihan adaptor

Selesaikan langkah-langkah berikut untuk melatih adaptor Anda dengan memverifikasi prediksi analisis massal dari model Moderasi Konten Rekognition.

Untuk melatih adaptor dengan memverifikasi prediksi dari model Moderasi Konten Rekognition, Anda harus:

1. Lakukan analisis Massal pada gambar Anda
2. Verifikasi prediksi yang dikembalikan untuk gambar Anda

Anda dapat memperoleh prediksi untuk gambar dengan melakukan analisis Massal dengan model dasar Rekognition atau adaptor yang telah Anda buat.

Jalankan analisis massal pada gambar Anda

Untuk melatih adaptor pada prediksi yang telah Anda verifikasi, Anda harus terlebih dahulu memulai pekerjaan analisis Massal untuk menganalisis sekumpulan gambar menggunakan model dasar Rekognition atau adaptor pilihan Anda. Untuk menjalankan pekerjaan analisis Massal:

1. [Masuk ke AWS Management Console dan buka konsol Amazon Rekognition di https://console.aws.amazon.com/rekognition/.](https://console.aws.amazon.com/rekognition/)
2. Di panel kiri, pilih Analisis massal. Halaman arahan analisis Massal muncul. Pilih Mulai Analisis Massal. Ikhtisar fitur Analisis Massal yang menunjukkan langkah-langkah untuk mengunggah gambar, menunggu analisis, meninjau hasil, dan memverifikasi prediksi model secara opsional. Daftar pekerjaan Analisis Massal terbaru untuk Moderasi Konten menggunakan model dasar.

Bulk Analysis Info

Analyze up to 10,000 images at once with a supported Rekognition feature.

How it works: Bulk Analysis for Amazon Rekognition

- 1. Upload images**
Upload up to 10K images to process with supported Rekognition features.
- 2. Wait for Bulk Analysis to complete**
The Bulk Analysis job may take 5-60 minutes, depending on the numbers of images processed.
- 3. Review results**
Review the results after the Bulk Analysis job is complete. You can also download the results.
- 4. Verify predictions - Optional**
Verify the model's predictions to assess model performance and/or train a custom adapter for enhanced accuracy.

Bulk Analysis jobs (11) Download results Start Bulk Analysis

Find a job by name

	Name	JobID	Status	Rekognition feature	Selected model	Output data location	Date created
<input type="radio"/>	TestPagination4	JobID	Succeeded	Content Moderation	Base model	S3 URL	October 23, 2023
<input type="radio"/>	TestPagination3	JobID	Succeeded	Content Moderation	Base model	S3 URL	October 23, 2023
<input type="radio"/>	TestPagination2	JobID	Succeeded	Content Moderation	Base model	S3 URL	October 23, 2023
<input type="radio"/>	TestPagination	JobID	Succeeded	Content Moderation	Base model	S3 URL	October 23, 2023

3. Jika ini adalah pertama kalinya Anda membuat adaptor, Anda akan diminta untuk membuat bucket Amazon Simple Storage Service untuk menyimpan file yang terkait dengan proyek dan adaptor Anda. Pilih Buat ember Amazon S3.
4. Pilih adaptor yang ingin Anda gunakan untuk analisis massal dengan menggunakan menu drop-down Pilih adaptor. Jika tidak ada adaptor yang dipilih, model dasar akan digunakan secara default. Untuk keperluan tutorial ini jangan memilih adaptor.

Bulk Analysis details

Choose a Rekognition feature

Content Moderation ▼

Choose an adapter

Choose a Custom Moderation adapter for your Bulk Analysis job. If no adapter is chosen, the base model is used by default.

No adapter chosen ▼

Bulk Analysis job name

Job name

⚠ This field is required.

Job name limited to 63 alphanumeric characters, no spaces or special characters.

Minimum confidence threshold

Minimum confidence (%)

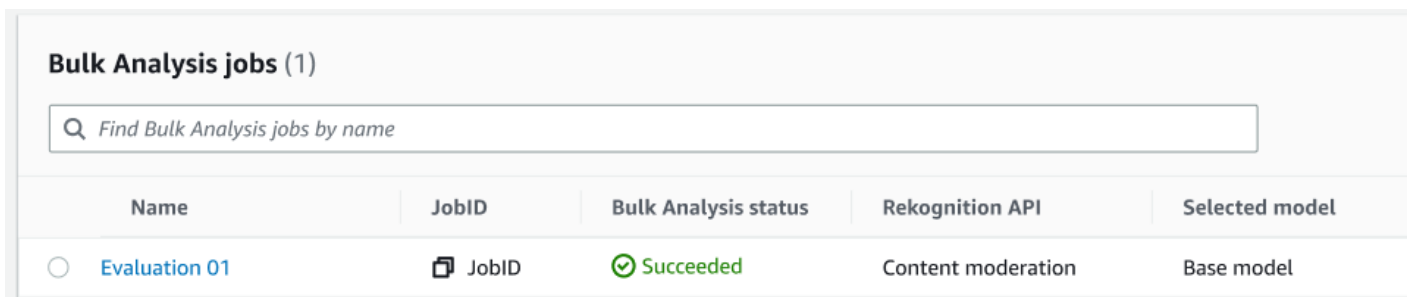
Labels aren't returned for inappropriate content that is detected with a lower confidence than the minimum confidence.

50 ▼



Upload images

5. Di bidang Nama pekerjaan analisis massal, isi nama pekerjaan analisis massal.
6. Pilih nilai untuk ambang kepercayaan minimum. Prediksi label dengan ambang kepercayaan kurang dari yang Anda pilih tidak akan dikembalikan. Perhatikan bahwa saat Anda mengevaluasi kinerja model nanti, Anda tidak akan dapat menyesuaikan ambang kepercayaan di bawah ambang kepercayaan minimum yang Anda pilih.

7. Pada langkah ini, Anda juga akan memberikan gambar yang ingin Anda analisis dengan analisis Massal. Gambar-gambar ini juga dapat digunakan untuk melatih adaptor Anda. Anda dapat memilih Unggah gambar dari komputer atau Impor gambar dari ember Amazon S3. Jika Anda memilih untuk mengimpor dokumen dari bucket Amazon S3, berikan jalur ke bucket dan folder yang berisi gambar pelatihan Anda. Jika Anda mengunggah dokumen langsung dari komputer Anda, perhatikan bahwa Anda hanya dapat mengunggah 50 gambar sekaligus.
8. Setelah mengisi informasi ini, pilih Mulai analisis. Ini akan memulai proses analisis menggunakan model dasar Rekognition.
9. Anda dapat memeriksa status pekerjaan analisis Massal Anda dengan memeriksa status Analisis Massal pekerjaan di halaman Analisis Massal utama. Ketika status Analisis Massal menjadi “Berhasil”, hasil analisis siap untuk ditinjau.



The screenshot shows a web interface for managing Bulk Analysis jobs. At the top, there is a search bar with the placeholder text "Find Bulk Analysis jobs by name". Below the search bar is a table with the following columns: Name, JobID, Bulk Analysis status, Rekognition API, and Selected model. The table contains one row with the following data: Name: Evaluation 01, JobID: JobID, Bulk Analysis status: Succeeded, Rekognition API: Content moderation, and Selected model: Base model.

Name	JobID	Bulk Analysis status	Rekognition API	Selected model
<input type="radio"/> Evaluation 01	 JobID	 Succeeded	Content moderation	Base model

10. Pilih analisis yang Anda buat dari daftar pekerjaan Analisis Massal.
11. Pada halaman detail Analisis Massal, Anda dapat melihat prediksi yang dibuat oleh model dasar Rekognition untuk gambar yang Anda unggah.
12. Tinjau kinerja model dasar. Anda dapat mengubah ambang kepercayaan yang harus dimiliki adaptor Anda untuk menetapkan label ke gambar dengan menggunakan slider ambang kepercayaan. Jumlah instans yang ditandai dan tidak ditandai akan berubah saat Anda menyesuaikan ambang kepercayaan. Panel Kategori Label menampilkan kategori tingkat atas yang diakui Rekognition, dan Anda dapat memilih kategori dalam daftar ini untuk menampilkan gambar apa pun yang telah ditetapkan label tersebut.

▼ Bulk Analysis details

Job name TestPagination4	Date created October 23, 2023	Recognition feature Content Moderation
Model version 6.1	Input location S3 URL	Output location S3 URL

Threshold [Info](#)

Confidence threshold

50%

Flagged (91)
Confidence greater than or equal to 50%

Unflagged (72)
Confidence less than 50%

Label categories [Info](#)

Explicit Nudity (21)

Suggestive (63)

Violence (0)

Hate Symbols (0)

Alcohol (34)

▼ Count of flagged images per label

Label	Count
Explicit Nudity	21
Suggestive	63
Violence	0
Hate Symbols	0
Alcohol	34
Drugs	1
Tobacco	0
Rude Gestures	0
Gambling	0

Count

Images (34)

< 1 2 3 4 >

Verifikasi prediksi

Jika Anda telah meninjau keakuratan model dasar Rekognition atau adaptor yang dipilih, dan ingin meningkatkan akurasi ini, Anda dapat menggunakan alur kerja verifikasi:

1. Setelah Anda selesai meninjau kinerja model dasar, Anda akan ingin memverifikasi prediksi. Memperbaiki prediksi akan memungkinkan Anda untuk melatih adaptor. Pilih Verifikasi prediksi dari bagian atas halaman Analisis massal.

Info You can verify model predictions with a confidence threshold of 50% or greater.

1. Verify model predictions to calculate model false positive rate and false negative rate.

2. To train a custom moderation adapter for enhanced accuracy, verify at least 20 false positives or 50 false negatives for one or more labels.

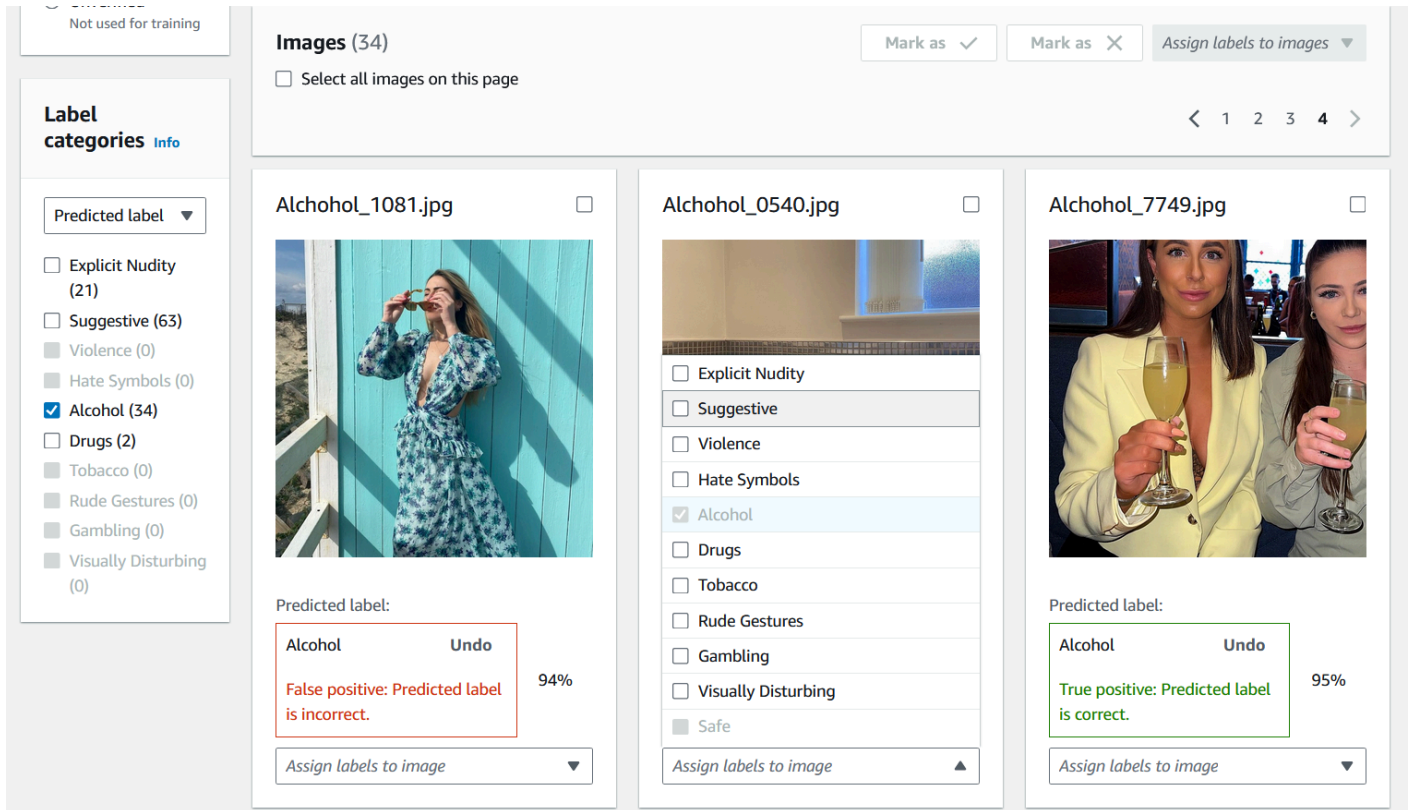
[Verify predictions](#)

2. Pada halaman Verifikasi prediksi, Anda dapat melihat semua gambar yang Anda berikan ke model dasar Rekognition, atau adaptor yang dipilih, bersama dengan label yang diprediksi untuk setiap gambar. Anda harus memverifikasi setiap prediksi sebagai benar atau salah menggunakan tombol di bawah gambar. Gunakan tombol "X" untuk menandai prediksi sebagai salah dan tombol tanda

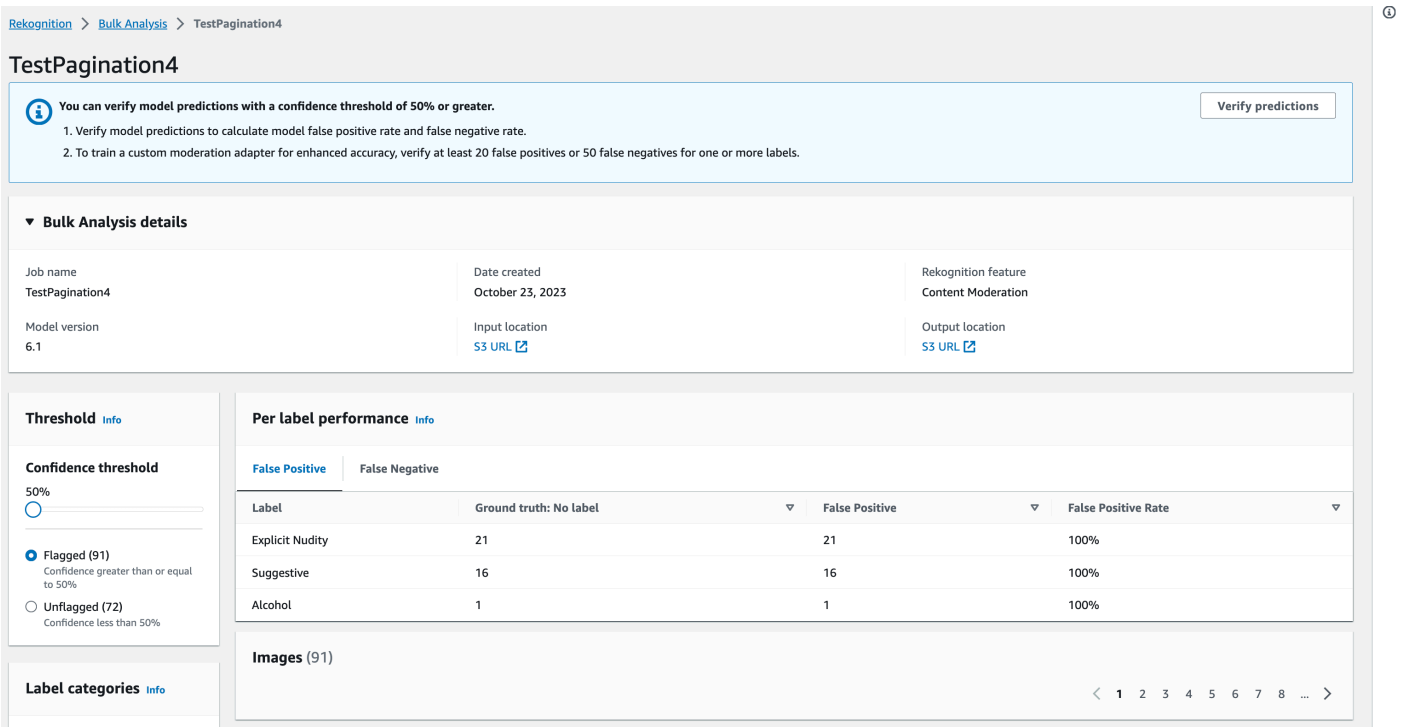
centang untuk menandai prediksi sebagai benar. Untuk melatih adaptor, Anda perlu memverifikasi setidaknya 20 prediksi positif palsu dan 50 prediksi negatif palsu untuk label tertentu. Semakin banyak prediksi yang Anda verifikasi, semakin baik kinerja adaptor.

The screenshot displays the Amazon Rekognition console interface for managing image labels. On the left, a sidebar titled 'Label categories' lists various categories with their respective counts: Explicit Nudity (21), Suggestive (63), Violence (0), Hate Symbols (0), Alcohol (34), Drugs (2), Tobacco (0), Rude Gestures (0), Gambling (0), and Visually Disturbing (0). The 'Alcohol' category is selected. The main area shows 'Images (34)' with a 'Select all images on this page' checkbox. Three image thumbnails are displayed, each with a predicted label of 'Alcohol' and a confidence score: 50%, 51%, and 55%. Below each image is an 'Assign labels to image' button. The image filenames are 'Alcohol_2955.jpg', 'Alcohol_1581.jpg', and 'Alcohol_1425.jpg'.

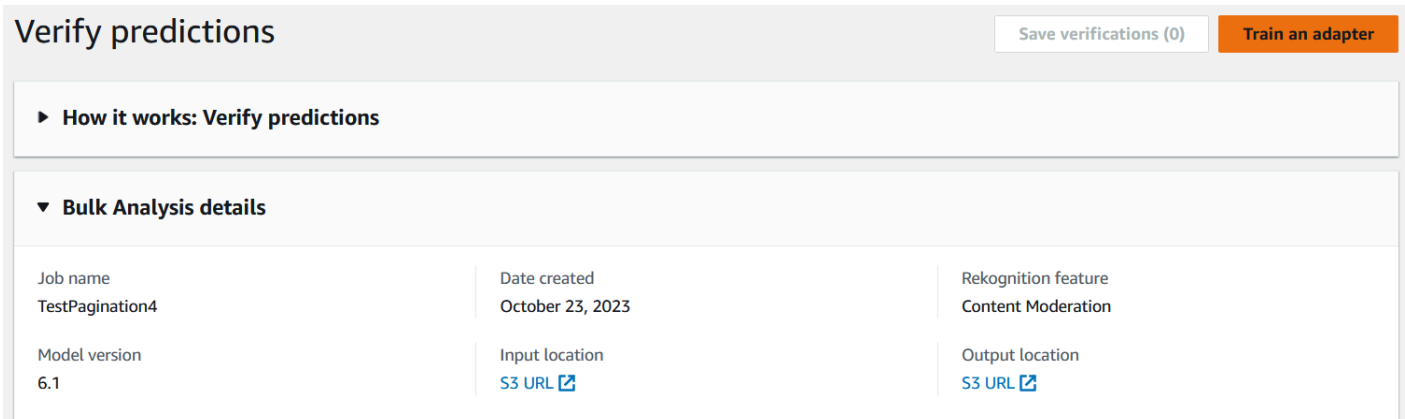
Setelah Anda memverifikasi prediksi, teks di bawah gambar akan berubah untuk menunjukkan jenis prediksi yang telah Anda verifikasi. Setelah Anda memverifikasi gambar, Anda juga dapat menambahkan label tambahan ke gambar menggunakan menu Tetapkan label ke gambar. Anda dapat melihat gambar mana yang ditandai atau tidak ditandai oleh model untuk ambang kepercayaan yang Anda pilih atau memfilter gambar berdasarkan kategori.



3. Setelah Anda selesai memverifikasi semua prediksi yang ingin Anda verifikasi, Anda dapat melihat statistik mengenai prediksi terverifikasi Anda di bagian kinerja Per label pada halaman Verifikasi. Anda juga dapat kembali ke halaman rincian analisis massal untuk melihat statistik ini.



4. Ketika Anda puas dengan statistik mengenai kinerja Per label, buka halaman Verifikasi prediksi lagi lalu pilih tombol Latih adaptor untuk mulai melatih adaptor Anda.



Bulk Analysis details		
Job name	Date created	Recognition feature
TestPagination4	October 23, 2023	Content Moderation
Model version	Input location	Output location
6.1	S3 URL	S3 URL

5. Pada halaman Latih adaptor Anda akan diminta untuk membuat proyek atau memilih proyek yang ada. Beri nama proyek dan adaptor yang akan dimuat oleh proyek. Anda juga harus menentukan sumber gambar pengujian Anda. Saat menentukan gambar, Anda dapat memilih Autosplit agar Rekognition secara otomatis menggunakan sebagian data pelatihan Anda sebagai gambar uji, atau Anda dapat menentukan file manifes secara manual. Disarankan untuk memilih Autosplit.

Train an adapter [Info](#)

Train an adapter using your verified predictions to enhance model accuracy.

Project details

Projects

Create a new project

Choose from an existing project

Project name

Name the project that groups your adapters.

Project name limited to 255 alphanumeric characters, no spaces or special characters.

Adapter name

Adapter name limited to 255 alphanumeric characters, no spaces or special characters.

Adapter description - *Optional*

Enter a description for quick reference

The adapter description can have up to 255 characters.

Test images

Provide test data

Test data is used to analyze the performance of your adapter.

Autosplit (Recommended)
Autosplit your data into test and training data.

Manually import manifest file
Labels must adhere to the Content Moderation label categories.

6. Tentukan tag apa pun yang Anda inginkan, serta AWS KMS kunci jika Anda tidak ingin menggunakan AWS kunci default. Disarankan untuk membiarkan Pembaruan otomatis diaktifkan.
7. Pilih Adaptor kereta.

Tag - *Optional*


A tag is a label you can assign to your adapter. Each tag consists of a key and an optional value.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 tags.


Image data encryption

Your data is encrypted by default with a key that AWS owns and manages for you. To choose a different key, customize your encryption settings. [Learn more](#) 

Customize encryption settings (advanced)

Confidence threshold

Confidence threshold
Adapter threshold was set on training manifest creation.

50 

Auto-update

Configure automatic retraining
Enable auto-update to automatically retrain your active adapters whenever a new version of moderation model is released.

Enable auto-update

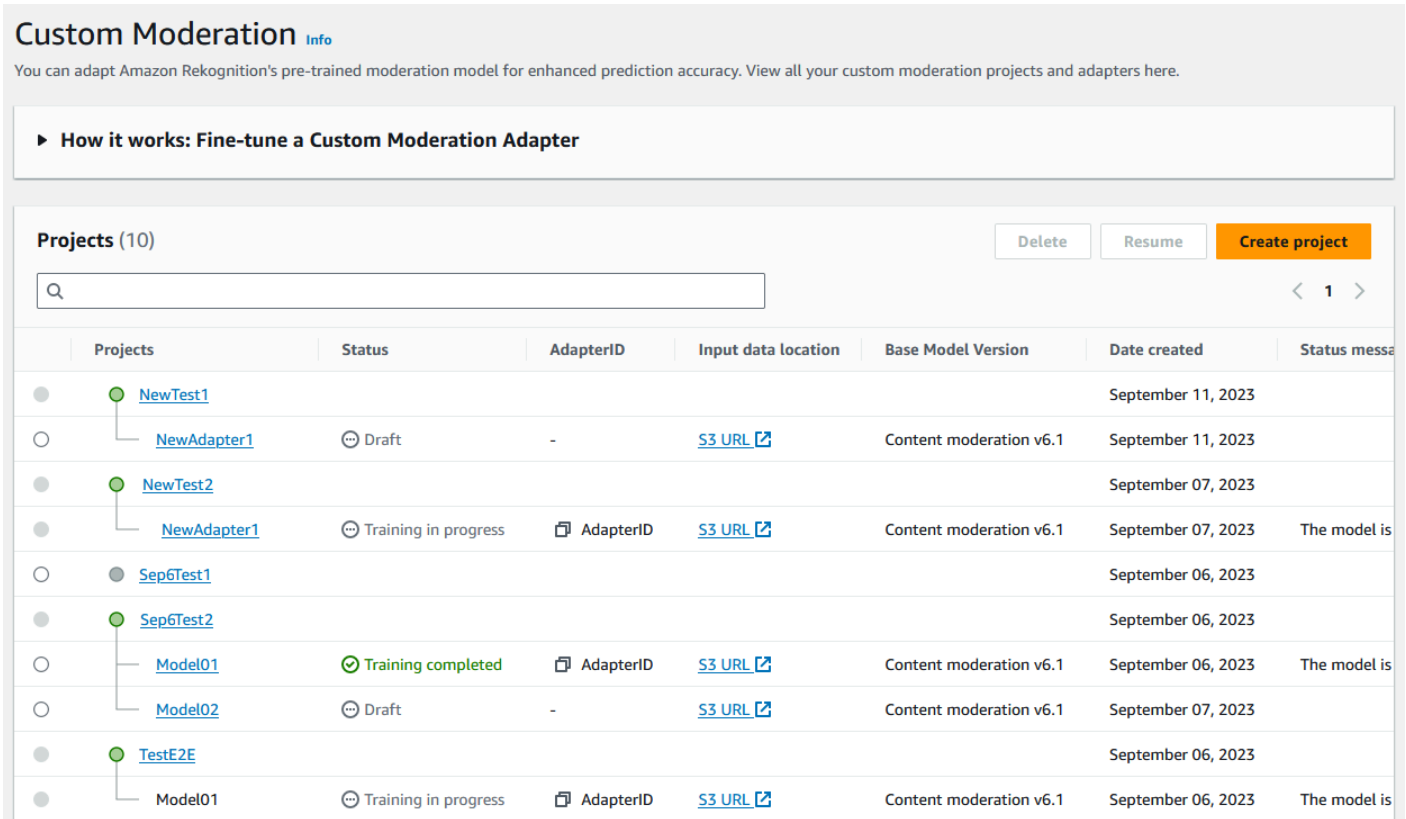
[Cancel](#) [Train adapter](#)

8. Setelah status adaptor Anda di halaman landing Moderasi Kustom menjadi “Pelatihan selesai”, Anda dapat meninjau kinerja adaptor Anda. Untuk informasi selengkapnya, lihat [Meninjau kinerja adaptor](#).

Meninjau kinerja adaptor

Untuk meninjau kinerja adaptor Anda:

1. Saat menggunakan konsol, Anda akan dapat melihat status adaptor apa pun yang terkait dengan proyek di bawah tab Projects di halaman landing Custom Moderation. Arahkan ke halaman landing Custom Moderation.



The screenshot shows the 'Custom Moderation' console page. At the top, there is a header 'Custom Moderation' with an 'Info' link. Below the header is a sub-header 'How it works: Fine-tune a Custom Moderation Adapter'. The main content area is titled 'Projects (10)' and contains a search bar, a 'Delete' button, a 'Resume' button, and a 'Create project' button. Below this is a table with the following columns: Projects, Status, AdapterID, Input data location, Base Model Version, Date created, and Status message. The table lists several projects and their associated adapters, including 'NewTest1', 'NewAdapter1', 'NewTest2', 'Sep6Test1', 'Sep6Test2', 'Model01', 'Model02', and 'TestE2E'. The status of each project and adapter is indicated by a colored circle and text, such as 'Draft', 'Training in progress', and 'Training completed'.

Projects	Status	AdapterID	Input data location	Base Model Version	Date created	Status message
NewTest1					September 11, 2023	
NewAdapter1	Draft	-	S3 URL	Content moderation v6.1	September 11, 2023	
NewTest2					September 07, 2023	
NewAdapter1	Training in progress	AdapterID	S3 URL	Content moderation v6.1	September 07, 2023	The model is
Sep6Test1					September 06, 2023	
Sep6Test2					September 06, 2023	
Model01	Training completed	AdapterID	S3 URL	Content moderation v6.1	September 06, 2023	The model is
Model02	Draft	-	S3 URL	Content moderation v6.1	September 07, 2023	
TestE2E					September 06, 2023	
Model01	Training in progress	AdapterID	S3 URL	Content moderation v6.1	September 06, 2023	The model is

2. Pilih adaptor yang ingin Anda tinjau dari daftar ini. Pada halaman Detail Adaptor berikut, Anda dapat melihat berbagai metrik untuk adaptor.

Threshold Info

Confidence Threshold
50%

Flagged (3)
Confidence more than 50%

Unflagged (26)
Confidence less than 50%

Label Categories Info

Predictions Label

- Explicit Nudity (0)
- Suggestive (1)
- Violence (0)
- Hate Symbols (0)
- Alcohol (0)
- Drugs (0)

▼ Adapter performance

False Positive Improvement: **25%**

False Negative Improvement: **-24%**

Per Label Performance

False Positive | **False Negative**

Label	Ground Truth: True Positives	Base Model False Negative	Adapter False Negative	False Negative Improvement
Suggestive	13	11	13	-15%
Alcohol	17	15	17	-12%

Images (21)

< 1 2 3 >

- Dengan panel Threshold Anda dapat mengubah ambang kepercayaan minimum yang harus dimiliki adaptor Anda untuk menetapkan label ke gambar. Jumlah instans yang ditandai dan tidak ditandai akan berubah saat Anda menyesuaikan ambang kepercayaan. Anda juga dapat memfilter berdasarkan kategori label untuk melihat metrik untuk kategori yang telah Anda pilih. Tetapkan ambang batas yang Anda pilih.
- Anda dapat menilai kinerja adaptor pada data pengujian dengan memeriksa metrik di panel Kinerja Adaptor. Metrik ini dihitung dengan membandingkan ekstraksi adaptor dengan anotasi “kebenaran dasar” pada set pengujian.

Panel kinerja adaptor menunjukkan tingkat Peningkatan Positif Palsu dan Peningkatan Negatif Palsu untuk adaptor yang Anda buat. Tab Kinerja Per Label dapat digunakan untuk membandingkan kinerja adaptor dan model dasar pada setiap kategori label. Ini menunjukkan jumlah prediksi positif palsu dan negatif palsu oleh model dasar dan adaptor, dikelompokkan berdasarkan kategori label. Dengan meninjau metrik ini, Anda dapat menentukan di mana adaptor perlu ditingkatkan. Untuk informasi selengkapnya tentang metrik ini, lihat [Mengevaluasi dan meningkatkan adaptor Anda](#).

Untuk meningkatkan kinerja, Anda dapat mengumpulkan lebih banyak gambar pelatihan dan kemudian membuat adaptor baru berbasis di dalam proyek. Cukup kembali ke halaman landing Custom Moderation dan buat adaptor baru di dalam proyek Anda, berikan lebih banyak gambar pelatihan untuk adaptor yang akan dilatih. Kali ini pilih opsi Tambahkan ke proyek yang ada alih-alih Buat proyek baru, dan pilih proyek yang ingin Anda buat adaptor baru dari menu tarik-turun nama Proyek. Seperti sebelumnya, beri anotasi gambar Anda atau berikan file manifes dengan anotasi.

Base Model Version [Info](#)

Base Model Version
You can only fine-tune the latest content moderation API

Content moderation v6.1 ▼

Project details

Projects

Create a new project Add to an existing project

Project name
Name the project that groups your adapters

TestE2E ▼

Adapter name - Provide a name for the adapter

Adapter name limited to 255 alphanumeric characters, no spaces or special characters.

Adapter description - *optional*

Enter a description for quick reference

The adapter description can have up to 255 characters.


Menggunakan adaptor Anda

[Setelah Anda membuat adaptor, Anda dapat menyediakannya ke operasi Rekognition yang didukung seperti Label. DetectModeration](#) Untuk melihat contoh kode yang dapat Anda gunakan untuk melakukan inferensi dengan adaptor Anda, pilih tab “Gunakan adaptor”, di mana Anda dapat melihat contoh kode untuk AWS CLI dan Python. Anda juga dapat mengunjungi bagian masing-masing dokumentasi untuk operasi yang telah Anda buat adaptor untuk melihat lebih banyak contoh kode, instruksi pengaturan, dan contoh JSON.

Test data location S3 URL ↗	Training data location S3 URL ↗	Output data location S3 URL ↗
--	--	--

Adapter performance | Training images | **Use adapter** | Tags

Use your adapter [Info](#)

 AdapterID
arn:aws:rekognition:us-east-1:000000000000:project/foo/version/bar/1692563172495

▼ API code

Use your trained adapter by calling the following AWS CLI commands or Python scripts.

AWS CLI command
 Python

```
aws rekognition detect-moderation-labels \
--image "s3object={Bucket=image-bucket,Name=image-name.jpg}" \
--project-version "arn:aws:rekognition:us-east-1:000000000000:project/foo/version
/bar/1692563172495"
```

Menghapus adaptor dan proyek Anda

Anda dapat menghapus adaptor individual, atau menghapus proyek Anda. Anda harus menghapus setiap adaptor yang terkait dengan proyek Anda sebelum Anda dapat menghapus proyek itu sendiri.

1. Untuk menghapus adaptor yang terkait dengan proyek, pilih adaptor dan kemudian pilih Hapus.
2. Untuk menghapus proyek, pilih proyek yang ingin Anda hapus lalu pilih Hapus.

Mengevaluasi dan meningkatkan adaptor Anda

Setelah setiap putaran pelatihan adaptor, Anda akan ingin meninjau metrik kinerja di alat Konsol Rekognition untuk menentukan seberapa dekat adaptor dengan tingkat kinerja yang Anda inginkan. Anda kemudian dapat lebih meningkatkan akurasi adaptor untuk gambar Anda dengan mengunggah kumpulan gambar pelatihan baru dan melatih adaptor baru di dalam proyek Anda. Setelah Anda membuat versi adaptor yang ditingkatkan, Anda dapat menggunakan konsol untuk menghapus versi adaptor yang lebih lama yang tidak lagi Anda perlukan.

Anda juga dapat mengambil metrik menggunakan operasi [DescribeProjectVersions](#) API.

Metrik kinerja

Setelah Anda menyelesaikan proses pelatihan dan membuat adaptor Anda, penting untuk mengevaluasi seberapa baik adaptor mengekstraksi informasi dari gambar Anda.

Dua metrik disediakan di Konsol Rekognition untuk membantu Anda menganalisis kinerja adaptor Anda: peningkatan positif palsu dan peningkatan negatif palsu.

Anda dapat melihat metrik ini untuk adaptor apa pun dengan memilih tab “Kinerja adaptor” di bagian adaptor konsol. Panel kinerja adaptor menunjukkan tingkat Peningkatan Positif Palsu dan Peningkatan Negatif Palsu untuk adaptor yang Anda buat.

Peningkatan positif palsu mengukur seberapa besar pengakuan adaptor terhadap positif palsu telah meningkat dibandingkan model dasar. Jika nilai peningkatan positif palsu adalah 25%, itu berarti adaptor meningkatkan pengakuannya atas positif palsu sebesar 25% pada kumpulan data pengujian.

Peningkatan negatif palsu mengukur seberapa besar pengakuan adaptor terhadap negatif palsu telah meningkat dibandingkan model dasar. Jika nilai peningkatan negatif palsu adalah 25%, itu berarti adaptor meningkatkan pengakuannya terhadap negatif palsu sebesar 25% pada kumpulan data pengujian.

Tab Kinerja Per Label dapat digunakan untuk membandingkan kinerja adaptor dan model dasar pada setiap kategori label. Ini menunjukkan jumlah prediksi positif palsu dan negatif palsu oleh model dasar dan adaptor, dikelompokkan berdasarkan kategori label. Dengan meninjau metrik ini, Anda dapat menentukan di mana adaptor perlu ditingkatkan.

Misalnya, jika tingkat Negatif Palsu Model Dasar untuk kategori label Alkohol adalah 15 sedangkan Tingkat Negatif Palsu Adaptor adalah 15 atau lebih tinggi, Anda tahu bahwa Anda harus fokus untuk menambahkan lebih banyak gambar yang berisi label Alkohol saat membuat adaptor baru.

[Saat menggunakan operasi API Rekognition, metrik F1-Score ditampilkan saat memanggil operasi Versions. DescribeProject](#)

Meningkatkan model Anda

Penerapan adaptor adalah proses berulang, karena Anda mungkin perlu melatih adaptor beberapa kali untuk mencapai tingkat akurasi target Anda. Setelah Anda membuat dan melatih adaptor Anda, Anda akan ingin menguji dan mengevaluasi kinerja adaptor Anda pada berbagai jenis label.

Jika akurasi adaptor Anda kurang di area mana pun, tambahkan contoh baru gambar tersebut untuk meningkatkan kinerja adaptor untuk label tersebut. Cobalah untuk memberikan adaptor dengan

contoh tambahan dan beragam yang mencerminkan kasus-kasus di mana ia berjuang. Menyediakan adaptor Anda dengan gambar yang representatif dan bervariasi memungkinkannya menangani beragam contoh dunia nyata.

Setelah menambahkan gambar baru ke set pelatihan Anda, latih ulang adaptor, lalu evaluasi ulang pada set pengujian dan label Anda. Ulangi proses ini hingga adaptor mencapai tingkat kinerja yang Anda inginkan. Jika Anda memberikan gambar dan anotasi yang lebih representatif, skor positif palsu dan negatif palsu secara bertahap akan meningkat dibandingkan iterasi pelatihan berturut-turut.

Format file manifes

Bagian berikut menunjukkan contoh format file manifes untuk file input, output, dan evaluasi.

Masukan manifes

File manifes adalah file yang dibatasi json-line, dengan setiap baris berisi JSON yang menyimpan informasi tentang satu gambar.

Setiap entri dalam Manifes Input harus berisi `source-ref` bidang dengan jalur ke gambar di bucket Amazon S3 dan, untuk Moderasi Kustom, `content-moderation-groundtruth` bidang dengan anotasi dasar. Semua gambar dalam satu kumpulan data diharapkan berada di ember yang sama. Struktur ini umum untuk pelatihan dan pengujian file manifes.

`CreateProjectVersionOperasi` untuk Moderasi Kustom menggunakan informasi yang disediakan dalam Manifes Input untuk melatih adaptor.

Contoh berikut adalah satu baris file manifes untuk satu gambar yang berisi satu kelas tidak aman:

```
{
  "source-ref": "s3://foo/bar/1.jpg",
  "content-moderation-groundtruth": {
    "ModerationLabels": [
      {
        "Name": "Rude Gesture"
      }
    ]
  }
}
```

Contoh berikut adalah satu baris file manifes untuk satu gambar tidak aman yang berisi beberapa kelas tidak aman, khususnya Nudity dan Rude Gesture.

```
{
  "source-ref": "s3://foo/bar/1.jpg",
  "content-moderation-groundtruth": {
    "ModerationLabels": [
      {
        "Name": "Rude Gesture"
      },
      {
        "Name": "Nudity"
      }
    ]
  }
}
```

Contoh berikut adalah satu baris file manifes untuk satu gambar yang tidak berisi kelas yang tidak aman:

```
{
  "source-ref": "s3://foo/bar/1.jpg",
  "content-moderation-groundtruth": {
    "ModerationLabels": []
  }
}
```

Untuk daftar lengkap label yang didukung, lihat [Memoderasi konten](#).

Manifes keluaran

Setelah menyelesaikan pekerjaan pelatihan, file manifes keluaran dikembalikan. File manifes keluaran adalah file yang dibatasi JSON-line dengan setiap baris berisi JSON yang menyimpan informasi untuk satu gambar. Amazon S3 Path to the OutputManifest dapat diperoleh dari DescribeProjectVersion respon:

- `TrainingDataResult.Output.Assets[0].GroundTruthManifest.S3Object` untuk dataset pelatihan
- `TestingDataResult.Output.Assets[0].GroundTruthManifest.S3Object` untuk menguji dataset

Informasi berikut dikembalikan untuk setiap entri di Output Manifest:

Nama Kunci	Deskripsi
<code>source-ref</code>	Referensi ke gambar di s3 yang disediakan di manifest input
<code>content-moderation-groundtruth</code>	Anotasi kebenaran dasar yang disediakan dalam manifest masukan
<code>detect-moderation-labels</code>	Prediksi adaptor, bagian dari dataset pengujian saja
<code>detect-moderation-labels-base-model</code>	Prediksi model dasar, bagian dari dataset pengujian saja

Prediksi model Adaptor dan Base dikembalikan pada ConfidenceThreshold 5.0 dalam format yang mirip dengan respons [DetectModerationLabel](#).

Contoh berikut menunjukkan struktur prediksi model Adaptor dan Base:

```
{
  "ModerationLabels": [
    {
      "Confidence": number,
      "Name": "string",
      "ParentName": "string"
    }
  ],
  "ModerationModelVersion": "string",
  "ProjectVersion": "string"
}
```

Untuk daftar lengkap label yang dikembalikan lihat [Memoderasi konten](#).

Hasil evaluasi terwujud

Setelah menyelesaikan pekerjaan pelatihan, file manifest hasil evaluasi dikembalikan. Manifest hasil evaluasi adalah output file JSON oleh pekerjaan pelatihan, dan berisi informasi tentang seberapa baik kinerja adaptor pada data pengujian.

Amazon S3 Path ke manifes hasil evaluasi dapat diperoleh dari `EvaluationResult.Summary.S3Object` lapangan dalam respons. `DescribeProjectVersion`

Contoh berikut menunjukkan struktur manifes hasil evaluasi:

```
{
  "AggregatedEvaluationResults": {
    "F1Score": number
  },
  "EvaluationDetails": {
    "EvaluationEndTimestamp": "datetime",
    "Labels": [
      "string"
    ],
    "NumberOfTestingImages": number,
    "NumberOfTrainingImages": number,
    "ProjectVersionArn": "string"
  },
  "ContentModeration": {
    "InputConfidenceThresholdEvalResults": {
      "ConfidenceThreshold": float,
      "AggregatedEvaluationResults": {
        "BaseModel": {
          "TruePositive": int,
          "TrueNegative": int,
          "FalsePositive": int,
          "FalseNegative": int
        },
        "Adapter": {
          "TruePositive": int,
          "TrueNegative": int,
          "FalsePositive": int,
          "FalseNegative": int
        }
      }
    },
    "LabelEvaluationResults": [
      {
        "Label": "string",
        "BaseModel": {
          "TruePositive": int,
          "TrueNegative": int,
          "FalsePositive": int,

```

```
        "FalseNegative": int
    },
    "Adapter": {
        "TruePositive": int,
        "TrueNegative": int,
        "FalsePositive": int,
        "FalseNegative": int
    }
}
]
}
"AllConfidenceThresholdsEvalResults": [
    {
        "ConfidenceThreshold": float,
        "AggregatedEvaluationResults": {
            "BaseModel": {
                "TruePositive": int,
                "TrueNegative": int,
                "FalsePositive": int,
                "FalseNegative": int
            },
            "Adapter": {
                "TruePositive": int,
                "TrueNegative": int,
                "FalsePositive": int,
                "FalseNegative": int
            }
        },
        "LabelEvaluationResults": [
            {
                "Label": "string",
                "BaseModel": {
                    "TruePositive": int,
                    "TrueNegative": int,
                    "FalsePositive": int,
                    "FalseNegative": int
                },
                "Adapter": {
                    "TruePositive": int,
                    "TrueNegative": int,
                    "FalsePositive": int,
                    "FalseNegative": int
                }
            }
        ]
    }
]
```

```
}  
  }  
    ]  
      }  
        ]  
          ]
```

File manifes evaluasi berisi:

- Hasil agregat seperti yang didefinisikan oleh F1Score
- Detail untuk pekerjaan evaluasi termasuk ProjectVersionArn, jumlah gambar pelatihan, jumlah gambar pengujian, dan label tempat adaptor dilatih.
- Agregat TruePositive TrueNegative, FalsePositive,, dan FalseNegative hasil untuk model dasar dan kinerja adaptor.
- Per label TruePositive TrueNegative, FalsePositive,, dan FalseNegative hasil untuk model dasar dan kinerja adaptor, dihitung pada ambang kepercayaan input.
- Agregat dan per label TruePositive, TrueNegative, FalsePositive, dan FalseNegative hasil untuk model dasar dan kinerja adaptor pada ambang kepercayaan yang berbeda. Ambang batas kepercayaan berkisar dari 5 hingga 100 dalam langkah 5.

Praktik terbaik untuk adaptor pelatihan

Disarankan agar Anda mematuhi praktik terbaik yang menyenangkan saat membuat, melatih, dan menggunakan adaptor Anda:

1. Data gambar sampel harus menangkap kesalahan representatif yang ingin ditekan oleh pelanggan. Jika model membuat kesalahan berulang pada gambar yang mirip secara visual, pastikan untuk membawa banyak gambar tersebut untuk pelatihan.
2. Alih-alih hanya membawa gambar bahwa model membuat kesalahan pada label Moderasi tertentu, pastikan juga untuk membawa gambar bahwa model tidak membuat kesalahan pada label Moderasi itu.
3. Berikan minimal 50 sampel Negatif Palsu ATAU 20 sampel Positif Palsu untuk pelatihan dan minimal 20 sampel untuk pengujian. Namun, sediakan gambar beranotasi sebanyak mungkin untuk kinerja adaptor yang lebih baik.

4. Memanotasi semua label yang penting bagi Anda untuk semua gambar - jika Anda memutuskan bahwa Anda perlu membubuhi keterangan kemunculan label pada gambar, pastikan untuk membubuhi keterangan kemunculan label ini pada semua gambar lainnya.
5. Data gambar sampel harus berisi sebanyak mungkin variasi pada label, dengan fokus pada contoh yang mewakili gambar yang akan dianalisis dalam pengaturan produksi.

Menyiapkan AutoUpdate izin

Rekognition mendukung fitur untuk adaptor AutoUpdate khusus. Ini berarti pelatihan ulang otomatis diberikan upaya terbaik saat AutoUpdate flag DIAKTIFKAN pada proyek. Pembaruan otomatis ini memerlukan izin untuk mengakses kumpulan data Pelatihan/Pengujian Anda dan AWS KMS kunci yang Anda gunakan untuk melatih adaptor pelanggan Anda. Anda dapat memberikan izin ini dengan mengikuti langkah-langkah di bawah ini.

Izin Bucket Amazon S3

Semua objek dan bucket Amazon S3 secara default bersifat privat. Hanya pemilik sumber daya, AWS akun yang membuat bucket, yang dapat mengakses bucket dan objek apa pun yang dikandungnya. Namun, pemilik sumber daya dapat memilih untuk memberikan izin akses ke sumber daya dan pengguna lain dengan menulis kebijakan bucket.

Jika Anda ingin membuat atau memodifikasi bucket Amazon S3 untuk digunakan sebagai sumber kumpulan data input dan tujuan hasil pelatihan dalam pelatihan adaptor khusus, Anda harus memodifikasi kebijakan bucket lebih lanjut. Untuk membaca dari atau menulis ke bucket Amazon S3, Rekognition harus memiliki izin berikut.

Rekognition Diperlukan Kebijakan Amazon S3

Rekognition memerlukan kebijakan izin dengan atribut berikut:

- Pernyataan SID
- Nama ember
- Nama utama layanan untuk Rekognition.
- Sumber daya yang dibutuhkan untuk Rekognition ember dan semua isinya
- Tindakan yang diperlukan yang perlu diambil oleh Rekognition.

Kebijakan berikut memungkinkan Rekognition mengakses bucket Amazon S3 selama pelatihan ulang otomatis.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "AllowRekognitionAutoUpdateActions",
      "Principal": {
        "Service": "rekognition.amazonaws.com"
      },
      "Action": [
        "s3:ListBucket",
        "s3:GetObject",
        "s3:PutObject",
        "s3:HeadObject",
        "s3:HeadBucket"
      ],
      "Resource": [
        "arn:aws:s3:::myBucketName",
        "arn:aws:s3:::myBucketName/*"
      ]
    }
  ]
}
```

Anda dapat mengikuti [panduan ini](#) untuk menambahkan kebijakan bucket di atas ke bucket S3 Anda.

Lihat informasi lebih lanjut tentang kebijakan bucket [di sini](#).

AWS KMS Izin Kunci

Rekognition memungkinkan Anda memberikan KmsKeyId opsional saat melatih adaptor khusus. Ketika disediakan, Rekognition menggunakan kunci ini untuk mengenkripsi pelatihan dan menguji gambar yang disalin ke dalam layanan untuk pelatihan model. Kunci ini juga digunakan untuk mengenkripsi hasil pelatihan dan file manifes yang ditulis ke bucket OutputConfig Amazon S3 keluaran ().

Jika Anda memilih untuk memberikan kunci KMS sebagai input ke pelatihan adaptor kustom Anda (yaitu `Rekognition:CreateProjectVersion`), Anda harus memodifikasi kebijakan Kunci KMS lebih lanjut untuk mengizinkan Principal Layanan Rekognition menggunakan kunci ini untuk pelatihan ulang otomatis di masa depan. Rekognition harus memiliki izin berikut.

Rekognition Diperlukan Kebijakan Kunci AWS KMS

Amazon Rekognition memerlukan kebijakan izin dengan atribut berikut:

- Pernyataan SID
- Nama utama layanan untuk Amazon Rekognition.
- Tindakan yang diperlukan yang perlu diambil oleh Amazon Rekognition.

Kebijakan utama berikut memungkinkan Amazon Rekognition mengakses kunci Amazon KMS selama pelatihan ulang otomatis:

```
{
  "Version": "2023-10-06",
  "Statement": [
    {
      "Sid": "KeyPermissions",
      "Effect": "Allow",
      "Principal": {
        "Service": "rekognition.amazonaws.com"
      },
      "Action": [
        "kms:DescribeKey",
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*"
    }
  ]
}
```

Anda dapat mengikuti [panduan ini](#) untuk menambahkan AWS KMS kebijakan di atas ke AWS KMS kunci Anda.

Lihat informasi lebih lanjut tentang AWS KMS kebijakan [di sini](#).

AWS Notifikasi Dasbor Kesehatan untuk Rekognition

Dasbor AWS Kesehatan Anda menyediakan dukungan untuk notifikasi yang berasal dari Rekognition. Pemberitahuan ini memberikan panduan kesadaran dan remediasi tentang perubahan terjadwal dalam Model Rekognition yang dapat memengaruhi aplikasi Anda. Hanya acara yang khusus untuk fitur Moderasi Konten Rekognition yang saat ini tersedia.

Dashboard AWS Kesehatan adalah bagian dari layanan AWS Kesehatan. Tidak perlu penyiapan dan dapat dilihat oleh pengguna yang diautentikasi di akun Anda. Untuk informasi selengkapnya, lihat [Memulai dengan Dasbor AWS Health](#).

Jika Anda menerima pesan pemberitahuan yang mirip dengan pesan berikut, itu harus diperlakukan sebagai alarm untuk mengambil tindakan.

Contoh pemberitahuan: Versi model baru tersedia untuk Moderasi Konten Rekognition.

Rekognition menerbitkan `AWS_MODERATION_MODEL_VERSION_UPDATE_NOTIFICATION` acara tersebut ke Dasbor AWS Kesehatan untuk menunjukkan bahwa versi baru dari model moderasi telah dirilis. Acara ini penting jika Anda menggunakan `DetectModerationLabels` API dan adaptor dengan API ini. Model baru dapat memengaruhi kualitas tergantung pada kasus penggunaan Anda, dan pada akhirnya akan menggantikan versi model sebelumnya. Disarankan untuk memvalidasi kualitas model Anda dan mengetahui jadwal pembaruan model saat Anda mendapatkan peringatan ini.

Jika Anda menerima pemberitahuan pembaruan versi model, Anda harus memperlakukannya sebagai alarm untuk mengambil tindakan. Jika Anda tidak menggunakan adaptor, Anda harus mengevaluasi kualitas model yang diperbarui pada kasus penggunaan yang ada. Jika Anda menggunakan adaptor, Anda harus melatih adaptor baru dengan model yang diperbarui dan mengevaluasi kualitasnya. Jika Anda memiliki set kereta otomatis, adaptor baru akan dilatih secara otomatis, dan kemudian Anda dapat mengevaluasi kualitasnya.

```
{
  "version": "0",
  "id": "id-number",
  "detail-type": "AWS Health Event",
  "source": "aws.health",
  "account": "123456789012",
  "time": "2023-10-06T06:27:57Z",
  "region": "region",
  "resources": [],
  "detail": {
    "eventArn": "arn:aws:health:us-east-1::event/
AWS_MODERATION_MODEL_UPDATE_NOTIFICATION_event-number",
    "service": "Rekognition",
    "eventTypeCode": "AWS_MODERATION_MODEL_VERSION_UPDATE_NOTIFICATION",
    "eventScopeCode": "ACCOUNT_SPECIFIC",
    "communicationId": "communication-id-number",
    "eventTypeCategory": "scheduledChange",
    "startTime": "Fri, 05 Apr 2023 12:00:00 GMT",
```

```
    "lastUpdatedTime": "Fri, 05 Apr 2023 12:00:00 GMT",
    "statusCode": "open",
    "eventRegion": "us-east-1",
    "eventDescription": [
      {
        "language": "en_US",
        "latestDescription": "A new model version is available for Rekognition
Content Moderation."
      }
    ]
  }
}
```

Lihat [Memantau peristiwa AWS Health dengan Amazon EventBridge](#) untuk mendeteksi dan bereaksi terhadap peristiwa AWS Kesehatan yang digunakan EventBridge.

Meninjau konten yang tidak sesuai dengan Amazon Augmented AI

Amazon Augmented AI (Amazon A2I) memungkinkan Anda untuk membangun alur kerja yang diperlukan untuk peninjauan manual prediksi machine learning.

Amazon Rekognition terintegrasi secara langsung dengan Amazon A2I sehingga Anda dapat dengan mudah menerapkan peninjauan manual untuk kasus penggunaan pendeteksian citra yang tidak aman. Amazon A2I menyediakan alur kerja peninjauan manual untuk moderasi citra. Hal ini memungkinkan Anda untuk dengan mudah meninjau prediksi dari Amazon Rekognition. Anda dapat menentukan ambang batas kepercayaan untuk kasus penggunaan Anda dan menyesuaikannya dari waktu ke waktu. Dengan Amazon A2I, Anda dapat menggunakan kolom penampil dalam organisasi Anda sendiri atau Amazon Mechanical Turk. Anda juga dapat menggunakan vendor tenaga kerja yang diprasaring oleh AWS untuk kualitas dan ketaatan terhadap prosedur keamanan.

Langkah-langkah berikut memandu Anda dalam mengatur Amazon A2I dengan Amazon Rekognition. Pertama, Anda membuat definisi aliran dengan Amazon A2I yang memiliki syarat yang memicu peninjauan manual. Kemudian, Anda melewati definisi aliran Amazon Resource Name (ARN) untuk operasi Amazon Rekognition `DetectModerationLabel`. Pada respons `DetectModerationLabel`, Anda dapat melihat apakah peninjauan manual diperlukan. Hasil peninjauan manual tersedia dalam bucket Amazon S3 yang diatur oleh definisi aliran.

Untuk melihat end-to-end demonstrasi cara menggunakan Amazon A2I dengan Amazon Rekognition, lihat salah satu tutorial berikut di Panduan Pengembang Amazon. SageMaker

- [Demo: Memulai di Konsol Amazon A2I](#)
- [Demo: Mulai Menggunakan Amazon A2I API](#)

Untuk mulai menggunakan API, Anda juga dapat menjalankan notebook Jupyter contoh. Lihat [Menggunakan Instans SageMaker Notebook dengan Notebook Amazon A2I Jupyter untuk menggunakan integrasi Amazon Augmented AI \(Amazon A2I\) notebook dengan Amazon Rekognition](#) [Contoh] dalam instance notebook. SageMaker

Berjalan DetectModerationLabels dengan Amazon A2I

Note

Buat semua sumber daya Amazon A2I dan sumber daya Amazon Rekognition Anda di Wilayah AWS yang sama.

1. Lengkapi prasyarat yang tercantum di [Memulai dengan Amazon Augmented AI](#) di Dokumentasi. SageMaker

Selain itu, ingatlah untuk mengatur izin IAM Anda seperti di halaman Izin [dan Keamanan di Amazon Augmented AI di Dokumentasi](#). SageMaker

2. Ikuti petunjuk untuk [Membuat Alur Kerja Tinjauan Manusia](#) di SageMakerDokumentasi.

Alur kerja peninjauan manual mengelola pengolahan citra. Hal ini mempertahankan syarat yang memicu peninjauan manual, tim kerja yang dikirim citra, templat UI yang digunakan tim kerja, dan bucket Amazon S3 yang dikirim hasil tim kerja.

Dalam `CreateFlowDefinition` panggilan Anda, Anda perlu mengatur ke `"DetectModerationLabelsAWS/Rekognition/ HumanLoopRequestSource /Image/V3"`. Setelah itu, Anda harus memutuskan bagaimana Anda ingin mengatur syarat yang memicu peninjauan manual.

Dengan Amazon Rekognition Anda memiliki dua opsi untuk `ConditionType`: `ModerationLabelConfidenceCheck`, dan `Sampling`.

`ModerationLabelConfidenceCheck` menciptakan loop manual ketika kepercayaan dari label moderasi berada dalam jangkauan. Pada akhirnya, `Sampling` mengirimkan persen dokumen

acak yang diproses untuk peninjauan manual. Setiap `ConditionType` menggunakan satu set `ConditionParameters` yang berbeda untuk menetapkan hasil dalam peninjauan manual.

`ModerationLabelConfidenceCheck` memiliki `ConditionParameters` `ModerationLabelName` yang mengatur kunci yang perlu ditinjau secara manual. Selain itu, ia memiliki kepercayaan diri, yang menetapkan kisaran persentase untuk mengirim ke tinjauan manusia dengan `LessThan`, `GreaterThan`, dan `Equals`. `Sampling` memiliki `RandomSamplingPercentage` yang menetapkan persen dokumen yang akan dikirim ke tinjauan manusia.

Contoh kode berikut adalah panggilan parsial `CreateFlowDefinition`. Ini mengirimkan citra untuk peninjauan manual jika diberi nilai kurang dari 98% pada label "Sugestif", dan lebih dari 95% pada label "Pakaian Renang atau Pakaian Dalam Wanita". Artinya bahwa jika citra tidak dianggap sugestif tetapi memang menampilkan citra wanita yang memakai pakaian dalam atau pakaian renang, Anda dapat memeriksa ulang citra dengan menggunakan peninjauan manual.

```
def create_flow_definition():
    '''
    Creates a Flow Definition resource

    Returns:
    struct: FlowDefinitionArn
    '''
    humanLoopActivationConditions = json.dumps(
        {
            "Conditions": [
                {
                    "And": [
                        {
                            "ConditionType": "ModerationLabelConfidenceCheck",
                            "ConditionParameters": {
                                "ModerationLabelName": "Suggestive",
                                "ConfidenceLessThan": 98
                            }
                        },
                        {
                            "ConditionType": "ModerationLabelConfidenceCheck",
                            "ConditionParameters": {
                                "ModerationLabelName": "Female Swimwear Or Underwear",
                                "ConfidenceGreaterThan": 95
                            }
                        }
                    ]
                }
            ]
        }
```

```

    }
  }
]
)

```

CreateFlowDefinition mengembalikan FlowDefinitionArn, yang Anda gunakan pada langkah berikutnya saat Anda memanggil DetectModerationLabels.

Untuk informasi selengkapnya lihat [CreateFlowDefinition](#) di Referensi SageMaker API.

3. Atur parameter HumanLoopConfig ketika Anda memanggil DetectModerationLabels, seperti dalam [Mendeteksi citra yang tidak pantas](#). Lihat langkah 4 untuk contoh panggilan DetectModerationLabels dengan pengaturan HumanLoopConfig.
 - a. Dalam parameter HumanLoopConfig, atur FlowDefinitionArn ke ARN definisi aliran yang Anda buat di langkah 2.
 - b. Atur HumanLoopName Anda. Harus unik dalam satu Wilayah dan harus dalam huruf kecil.
 - c. (Opsional) Anda dapat menggunakan DataAttributes untuk mengatur apakah citra yang Anda teruskan ke Amazon Rekognition tidak mengandung informasi pribadi. Anda harus mengatur parameter ini untuk mengirim informasi ke Amazon Mechanical Turk.
4. Jalankan DetectModerationLabels.

Contoh berikut menunjukkan cara menggunakan AWS CLI dan AWS SDK for Python (Boto3) untuk menjalankan DetectModerationLabels dengan pengaturan HumanLoopConfig.

AWS SDK for Python (Boto3)

Contoh permintaan berikut menggunakan SDK for Python (Boto3). Untuk informasi selengkapnya, lihat [detect_moderation_labels](#) di Referensi API AWSSDK for Python (Boto).

```

import boto3

rekognition = boto3.client("rekognition", aws-region)

response = rekognition.detect_moderation_labels( \
    Image={'S3Object': {'Bucket': bucket_name, 'Name': image_name}}, \

```



```
HumanLoopConfig={ \
  'HumanLoopName': 'human_loop_name', \
  'FlowDefinitionArn': , "arn:aws:sagemaker:aws-
region:aws_account_number:flow-definition/flow_def_name" \
  'DataAttributes': {'ContentClassifiers':
['FreeOfPersonallyIdentifiableInformation', 'FreeOfAdultContent']}
  })
```

AWS CLI

Contoh permintaan berikut menggunakan AWS CLI. Untuk informasi selengkapnya, lihat [detect-moderation-labels](#) di [Referensi AWS CLI Perintah](#).

```
$ aws rekognition detect-moderation-labels \
  --image "S3Object={Bucket='bucket_name',Name='image_name'}" \
  --human-loop-config
  HumanLoopName="human_loop_name",FlowDefinitionArn="arn:aws:sagemaker:aws-
region:aws_account_number:flow-
definition/
flow_def_name",DataAttributes='{"ContentClassifiers":["FreeOfPersonallyIdentifiableInforma
"FreeOfAdultContent"]}'
```

```
$ aws rekognition detect-moderation-labels \
  --image "S3Object={Bucket='bucket_name',Name='image_name'}" \
  --human-loop-config \
  '{"HumanLoopName": "human_loop_name", "FlowDefinitionArn":
"arn:aws:sagemaker:aws-region:aws_account_number:flow-
definition/flow_def_name", "DataAttributes": {"ContentClassifiers":
["FreeOfPersonallyIdentifiableInformation", "FreeOfAdultContent"]}]}'
```

Saat Anda menjalankan `DetectModerationLabels` dengan `HumanLoopConfig` diaktifkan, Amazon Rekognition memanggil SageMaker operasi API. `StartHumanLoop` Perintah ini mengambil respons dari `DetectModerationLabels` dan memeriksa syarat definisi aliran dalam contoh. Jika memenuhi persyaratan untuk ditinjau, ia akan mengirimkan `HumanLoopArn`. Artinya bahwa anggota tim kerja yang Anda tetapkan dalam definisi alur sekarang dapat meninjau citra. Memanggil `DescribeHumanLoop` operasi waktu aktif Amazon Augmented AI memberikan informasi tentang hasil loop. Untuk informasi selengkapnya, lihat [DescribeHumanLoop](#) di dokumentasi [Referensi API Augmented AI Amazon](#).

Setelah citra ditinjau, Anda dapat melihat hasilnya di bucket yang ditentukan dalam alur keluaran definisi aliran. Amazon A2I juga akan memberi tahu Anda dengan Amazon CloudWatch Events saat peninjauan selesai. Untuk melihat peristiwa apa yang harus dicari, lihat [CloudWatch Peristiwa](#) di SageMakerDokumentasi.

Untuk informasi selengkapnya, lihat [Memulai dengan Amazon Augmented AI](#) di Dokumentasi SageMaker.

Mendeteksi teks

Amazon Rekognition dapat mendeteksi teks dalam citra dan video. Hal ini kemudian dapat mengonversi teks yang terdeteksi menjadi teks yang dapat dibaca oleh mesin. Anda dapat menggunakan deteksi teks yang dapat dibaca mesin dalam citra untuk menerapkan solusi seperti:

- Pencarian visual. Misalnya, mengambil dan menampilkan citra yang berisi teks yang sama.
- Wawasan konten. Misalnya, menyediakan wawasan tentang tema yang muncul dalam teks yang dikenali dalam bingkai video yang diekstraksi. Aplikasi Anda dapat mencari teks yang dikenali untuk konten yang relevan, seperti berita, skor olahraga, nomor atlet, dan keterangan.
- Navigasi. Misalnya, mengembangkan aplikasi seluler berkemampuan bicara untuk penyandang disabilitas yang mengenali nama restoran, toko, atau rambu jalan.
- Keamanan publik dan dukungan transportasi. Misalnya, mendeteksi nomor plat mobil dari citra kamera lalu lintas.
- Memfilter. Misalnya, mem-filter informasi pengenal pribadi (PII) dari citra.

Untuk deteksi teks dalam video, Anda dapat menerapkan solusi seperti:

- Mencari video untuk klip dengan kata kunci teks tertentu, seperti nama tamu pada grafik di acara berita.
- Memoderasi konten untuk kepatuhan dengan standar organisasi dengan cara mendeteksi teks, kata tidak senonoh, atau spam yang tidak disengaja.
- Menemukan semua lapisan teks pada lini masa video untuk pemrosesan lebih lanjut, seperti mengganti teks dengan teks dalam bahasa lain untuk internasionalisasi konten.
- Menemukan lokasi teks, sehingga grafik lain dapat disejajarkan dengan tepat.

Untuk mendeteksi teks dalam gambar dalam format JPEG atau PNG, gunakan

[DetectText](#) operasi. Untuk mendeteksi teks dalam video secara asinkron, gunakan dan operasi [StartTextDetectionGetTextDetection](#) Operasi deteksi teks citra dan video mendukung sebagian besar font, termasuk font yang sangat bergaya. Setelah mendeteksi teks, Amazon Rekognition membuat representasi kata-kata dan baris teks yang terdeteksi, menunjukkan hubungan di antaranya, dan memberi tahu Anda tempat teks berada pada bingkai citra atau video.

Operasi `DetectText` dan `GetTextDetection` mendeteksi kata-kata dan baris. Sebuah kata adalah satu atau lebih karakter skrip yang tidak dipisahkan oleh spasi. `DetectText` dapat

mendeteksi hingga 100 kata dalam sebuah gambar. `GetTextDetection` juga dapat mendeteksi hingga 100 kata per frame video.

Kata adalah satu atau lebih karakter skrip yang tidak dipisahkan oleh spasi. Amazon Rekognition dirancang untuk mendeteksi kata-kata dalam bahasa Inggris, Arab, Rusia, Jerman, Prancis, Italia, Portugis, dan Spanyol.

Baris adalah rangkaian kata yang berjarak sama. Sebuah baris belum tentu merupakan kalimat yang lengkap (titik tidak menunjukkan akhir dari sebuah baris). Misalnya, Amazon Rekognition mendeteksi nomor lisensi pengemudi sebagai sebuah baris. Sebuah baris berakhir ketika tidak ada teks yang diselaraskan setelahnya atau ketika ada jarak yang besar di antara kata-kata, relatif terhadap panjang kata. Tergantung pada celah antar kata, Amazon Rekognition mungkin mendeteksi beberapa baris dalam teks yang diselaraskan dalam arah yang sama. Jika kalimat mencakup beberapa baris, maka operasi mengembalikan beberapa baris.

Pertimbangkan citra berikut.



Kotak biru mewakili informasi tentang teks yang terdeteksi dan lokasi teks yang dikembalikan oleh operasi `DetectText`. Dalam contoh ini, Amazon Rekognition mendeteksi "IT'S", "SENIN", "tapi", "tetap", dan "Tersenyum" sebagai kata. Amazon Rekognition mendeteksi "IT'S", "SENIN", "tapi tetap",

dan "Tersenyum" sebagai baris. Agar terdeteksi, teks harus berada dalam orientasi +/- 90 derajat dari sumbu horizontal.

Sebagai contoh, lihat [Mendeteksi teks dalam sebuah citra](#).

Topik

- [Mendeteksi teks dalam sebuah citra](#)
- [Mendeteksi teks dalam video yang tersimpan](#)

Mendeteksi teks dalam sebuah citra

Anda dapat menyediakan citra input sebagai array bit citra (bit citra yang dikodekan base64), atau sebagai objek Amazon S3. Dalam prosedur ini, Anda mengunggah citra JPEG atau PNG ke bucket S3 Anda dan menentukan nama file.

Untuk mendeteksi teks dalam citra (API)

1. Jika Anda belum melakukannya, selesaikan prasyarat berikut.
 - a. Buat atau perbarui pengguna dengan `AmazonRekognitionFullAccess` dan `AmazonS3ReadOnlyAccess` izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS Command Line Interface dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Unggah citra yang berisi teks ke bucket S3 Anda.

Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

3. Gunakan contoh berikut untuk memanggil operasi `DetectText`.

Java

Kode contoh berikut menampilkan baris dan kata-kata yang terdeteksi dalam citra.

Mengganti nilai-nilai bucket dan photo dengan nama bucket S3 dan citra yang Anda gunakan pada langkah 2.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)
```

```
package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.DetectTextRequest;
import com.amazonaws.services.rekognition.model.DetectTextResult;
import com.amazonaws.services.rekognition.model.TextDetection;
import java.util.List;
```

```
public class DetectText {

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

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

        AmazonRekognition rekognitionClient =
        AmazonRekognitionClientBuilder.defaultClient();

        DetectTextRequest request = new DetectTextRequest()
            .withImage(new Image()
                .withS3Object(new S3Object()
                    .withName(photo)
                    .withBucket(bucket)));

        try {
            DetectTextResult result = rekognitionClient.detectText(request);
            List<TextDetection> textDetections = result.getTextDetections();

            System.out.println("Detected lines and words for " + photo);
            for (TextDetection text: textDetections) {

                System.out.println("Detected: " + text.getDetectedText());
            }
        }
    }
}
```

```
        System.out.println("Confidence: " +
text.getConfidence().toString());
        System.out.println("Id : " + text.getId());
        System.out.println("Parent Id: " + text.getParentId());
        System.out.println("Type: " + text.getType());
        System.out.println();
    }
} catch (AmazonRekognitionException e) {
    e.printStackTrace();
}
}
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-
 * sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */

//snippet-start:[rekognition.java2.detect_text.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DetectTextRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectTextResponse;
import software.amazon.awssdk.services.rekognition.model.TextDetection;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
//snippet-end:[rekognition.java2.detect_text.import]

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

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "  <sourceImage>\n\n" +
            "Where:\n" +
            "  sourceImage - The path to the image that contains text (for
example, C:\\AWS\\pic1.png). \n\n";

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

        String sourceImage = args[0] ;
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("default"))
            .build();

        detectTextLabels(rekClient, sourceImage );
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.detect_text.main]
```



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

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectTextRequest textRequest = DetectTextRequest.builder()
            .image(souImage)
            .build();

        DetectTextResponse textResponse = rekClient.detectText(textRequest);
        List<TextDetection> textCollection = textResponse.textDetections();
        System.out.println("Detected lines and words");
        for (TextDetection text: textCollection) {
            System.out.println("Detected: " + text.detectedText());
            System.out.println("Confidence: " + text.confidence().toString());
            System.out.println("Id : " + text.id());
            System.out.println("Parent Id: " + text.parentId());
            System.out.println("Type: " + text.type());
            System.out.println();
        }

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

AWS CLI

AWS CLI Perintah ini menampilkan output JSON untuk operasi detect-text CLI.

Mengganti nilai-nilai Bucket dan Name dengan nama bucket S3 dan citra yang Anda gunakan pada langkah 2.

Ganti nilai profile_name dengan nama profil pengembang Anda.

```
aws rekognition detect-text --image '{"S3Object":{"Bucket":"bucket-name","Name":"image-name"}}' --profile default
```

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda alih-alih tanda kutip tunggal dan hindari tanda kutip ganda bagian dalam dengan garis miring terbalik (yaitu \) untuk mengatasi kesalahan parser yang mungkin Anda temui. Sebagai contoh, lihat berikut ini:

```
aws rekognition detect-text --image "{\"S3Object\":{\"Bucket\":\"bucket-name\", \"Name\":\"image-name\"}}" --profile default
```

Python

Contoh kode berikut menampilkan baris dan kata-kata yang terdeteksi dalam sebuah citra.

Mengganti nilai-nilai bucket dan photo dengan nama S3bucket dan citra yang Anda gunakan pada langkah 2. Ganti nilai `profile_name` di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

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

import boto3

def detect_text(photo, bucket):

    session = boto3.Session(profile_name='default')
    client = session.client('rekognition')

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

    textDetections = response['TextDetections']
    print('Detected text\n-----')
    for text in textDetections:
        print('Detected text:' + text['DetectedText'])
        print('Confidence: ' + "{:.2f}".format(text['Confidence']) + "%")
        print('Id: {}'.format(text['Id']))
        if 'ParentId' in text:
```

```
        print('Parent Id: {}'.format(text['ParentId']))
        print('Type:' + text['Type'])
        print()
    return len(textDetections)

def main():
    bucket = 'bucket-name'
    photo = 'photo-name'
    text_count = detect_text(photo, bucket)
    print("Text detected: " + str(text_count))

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

.NET

Contoh kode berikut menampilkan baris dan kata-kata yang terdeteksi dalam sebuah citra.

Ganti nilai-nilai bucket dan photo dengan nama bucket S3 dan citra yang Anda gunakan pada langkah 2.

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

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

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

        AmazonRekognitionClient rekognitionClient = new
AmazonRekognitionClient();

        DetectTextRequest detectTextRequest = new DetectTextRequest()
        {
            Image = new Image()
            {
```

```

        S3Object = new S3Object()
        {
            Name = photo,
            Bucket = bucket
        }
    };

    try
    {
        DetectTextResponse detectTextResponse =
rekognitionClient.DetectText(detectTextRequest);
        Console.WriteLine("Detected lines and words for " + photo);
        foreach (TextDetection text in detectTextResponse.TextDetections)
        {
            Console.WriteLine("Detected: " + text.DetectedText);
            Console.WriteLine("Confidence: " + text.Confidence);
            Console.WriteLine("Id : " + text.Id);
            Console.WriteLine("Parent Id: " + text.ParentId);
            Console.WriteLine("Type: " + text.Type);
        }
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }
}
}

```

Node.JS

Contoh kode berikut menampilkan baris dan kata-kata yang terdeteksi dalam sebuah citra.

Mengganti nilai-nilai bucket dan photo dengan nama S3bucket dan citra yang Anda gunakan pada langkah 2. Ganti nilai region dengan wilayah yang ditemukan di kredensial.aws Anda. Ganti nilai profile_name di baris yang membuat sesi Rekognition dengan nama profil pengembang Anda.

```

var AWS = require('aws-sdk');

const bucket = 'bucket' // the bucketname without s3://
const photo = 'photo' // the name of file

```

```
const config = new AWS.Config({
  accessKeyId: process.env.AWS_ACCESS_KEY_ID,
  secretAccessKey: process.env.AWS_SECRET_ACCESS_KEY,
})
AWS.config.update({region:'region'});
const client = new AWS.Rekognition();
const params = {
  Image: {
    S3object: {
      Bucket: bucket,
      Name: photo
    },
  },
}
client.detectText(params, function(err, response) {
  if (err) {
    console.log(err, err.stack); // handle error if an error occurred
  } else {
    console.log(`Detected Text for: ${photo}`)
    console.log(response)
    response.TextDetections.forEach(label => {
      console.log(`Detected Text: ${label.DetectedText}`),
      console.log(`Type: ${label.Type}`),
      console.log(`ID: ${label.Id}`),
      console.log(`Parent ID: ${label.ParentId}`),
      console.log(`Confidence: ${label.Confidence}`),
      console.log(`Polygon: `)
      console.log(label.Geometry.Polygon)
    }
  )
}
});
```

DetectText permintaan operasi

Dalam operasi DetectText, Anda menyediakan citra input, sebagai array bit yang dikodekan base64 atau sebagai citra yang disimpan dalam bucket Amazon S3. Contoh permintaan JSON berikut menunjukkan citra yang dimuat dari bucket Amazon S3.

```
{
  "Image": {
    "S3object": {
```

```
        "Bucket": "bucket",
        "Name": "inputtext.jpg"
    }
}
```

Filter

Pemfilteran berdasarkan wilayah teks, ukuran, dan skor kepercayaan memberikan Anda fleksibilitas tambahan untuk mengendalikan output deteksi teks Anda. Dengan menggunakan wilayah yang diminati, Anda dapat dengan mudah membatasi deteksi teks ke wilayah yang relevan bagi Anda, misalnya, kanan atas foto profil atau lokasi tetap terkait dengan titik referensi saat membaca nomor suku cadang dari citra mesin. Filter ukuran kotak pembatas kata dapat digunakan untuk menghindari teks latar belakang kecil yang mungkin ramai atau tidak relevan. Filter kepercayaan kata memungkinkan Anda untuk menghapus hasil yang mungkin tidak dapat diandalkan karena buram atau ternoda.

Untuk informasi mengenai nilai filter, lihat [DetectTextFilters](#).

Anda dapat menggunakan filter berikut:

- **MinConfidence**—Menetapkan tingkat kepercayaan deteksi kata. Kata-kata dengan kepercayaan deteksi di bawah tingkat ini dikeluarkan dari hasil. Nilai harus antara 0 hingga 100.
- **MinBoundingBoxWidth**— Mengatur lebar minimum kotak pembatas kata. Kata-kata dengan kotak pembatas yang lebih kecil dari nilai ini dikeluarkan dari hasil. Nilainya relatif terhadap lebar bingkai citra.
- **MinBoundingBoxHeight**— Mengatur tinggi minimum kotak pembatas kata. Kata-kata dengan tinggi kotak pembatas kurang dari nilai ini dikeluarkan dari hasil. Nilainya relatif terhadap tinggi bingkai citra.
- **RegionsOfInterest**— Membatasi deteksi ke wilayah tertentu dari bingkai gambar. Nilai-nilainya relatif terhadap dimensi bingkai. Untuk teks yang hanya sebagian dalam suatu wilayah, responsnya tidak ditentukan.

DetectText respon operasi

DetectTextOperasi menganalisis gambar dan mengembalikan array, `TextDetections`, di mana setiap elemen ([TextDetection](#)) mewakili garis atau kata yang terdeteksi dalam gambar. Untuk setiap elemen, DetectText mengembalikan informasi berikut:

- Teks yang terdeteksi (`DetectedText`)
- Hubungan antara kata-kata dan baris (`Id` dan `ParentId`)
- Lokasi teks pada citra (`Geometry`)
- kepercayaan Amazon Rekognition memiliki keakuratan teks yang terdeteksi dan kotak pembatas (`Confidence`)
- tipe teks yang terdeteksi (`Type`)

Teks yang terdeteksi

Setiap elemen `TextDetection` berisi teks yang dikenali (kata atau baris) dalam bidang `DetectedText`. Sebuah kata adalah satu atau lebih karakter skrip yang tidak dipisahkan oleh spasi. `DetectText` dapat mendeteksi hingga 100 kata dalam sebuah gambar. Teks yang dikembalikan mungkin menyertakan karakter yang membuat kata tidak dapat dikenali. Misalnya, `C@t` alih-alih `Cat`. Untuk menentukan apakah elemen `TextDetection` mewakili baris teks atau kata, gunakan bidang `Type`.

Setiap elemen `TextDetection` termasuk nilai persentase yang mewakili tingkat kepercayaan bahwa Amazon Rekognition memiliki keakuratan teks yang terdeteksi dan kotak pembatas yang mengelilingi teks.

Hubungan kata dan baris

Setiap elemen `TextDetection` memiliki bidang pengenalan, `Id`. `Id` menunjukkan posisi kata dalam satu baris. Jika elemennya adalah kata, bidang pengenalan induk, `ParentId`, mengidentifikasi baris tempat kata tersebut terdeteksi. `ParentId` untuk baris adalah nol. Sebagai contoh, baris "tapi tetap" dalam citra contoh memiliki nilai `Id` dan `ParentId` berikut:

Teks	ID	ID Induk
tapi tetap	3	
tapi	8	3
tetap	9	3

Lokasi teks pada citra

Untuk menentukan di mana teks yang dikenali berada pada gambar, gunakan informasi kotak pembatas ([Geometri](#)) yang dikembalikan oleh DetectText Objek Geometry berisi dua tipe informasi kotak pembatas untuk baris dan kata yang terdeteksi:

- Garis persegi panjang kasar sejajar sumbu dalam suatu objek [BoundingBox](#)
- [Poligon berbutir halus yang terdiri dari beberapa koordinat X dan Y dalam array Titik](#)

Koordinat kotak pembatas dan poligon menunjukkan lokasi teks terletak pada citra sumber. Nilai koordinat adalah rasio ukuran citra secara keseluruhan. Untuk informasi lebih lanjut, lihat [BoundingBox](#).

Berikut respons JSON dari operasi DetectText yang menunjukkan kata-kata dan baris yang terdeteksi pada citra berikut.



```
{
  'TextDetections': [{
    'Confidence': 99.35693359375,
    'DetectedText': "IT'S",
    'Geometry': {
      'BoundingBox': {
        'Height': 0.09988046437501907,
```



```
        'Left': 0.6684935688972473,
        'Top': 0.18226495385169983,
        'Width': 0.1461552083492279},
    'Polygon': [{ 'X': 0.6684935688972473,
                  'Y': 0.1838926374912262},
                { 'X': 0.8141663074493408,
                  'Y': 0.18226495385169983},
                { 'X': 0.8146487474441528,
                  'Y': 0.28051772713661194},
                { 'X': 0.6689760088920593,
                  'Y': 0.2821454107761383}]],
    'Id': 0,
    'Type': 'LINE'},
{'Confidence': 99.6207275390625,
 'DetectedText': 'MONDAY',
 'Geometry': {'BoundingBox': {'Height': 0.11442459374666214,
                              'Left': 0.5566731691360474,
                              'Top': 0.3525116443634033,
                              'Width': 0.39574965834617615},
              'Polygon': [{ 'X': 0.5566731691360474,
                            'Y': 0.353712260723114},
                          { 'X': 0.9522717595100403,
                            'Y': 0.3525116443634033},
                          { 'X': 0.9524227976799011,
                            'Y': 0.4657355844974518},
                          { 'X': 0.5568241477012634,
                            'Y': 0.46693623065948486}]],
    'Id': 1,
    'Type': 'LINE'},
{'Confidence': 99.6160888671875,
 'DetectedText': 'but keep',
 'Geometry': {'BoundingBox': {'Height': 0.08314694464206696,
                              'Left': 0.6398131847381592,
                              'Top': 0.5267938375473022,
                              'Width': 0.2021435648202896},
              'Polygon': [{ 'X': 0.640289306640625,
                            'Y': 0.5267938375473022},
                          { 'X': 0.8419567942619324,
                            'Y': 0.5295097827911377},
                          { 'X': 0.8414806723594666,
                            'Y': 0.609940767288208},
                          { 'X': 0.6398131847381592,
                            'Y': 0.6072247624397278}]],
    'Id': 2,
```

```
'Type': 'LINE'},
{'Confidence': 88.95134735107422,
 'DetectedText': 'Smiling',
 'Geometry': {'BoundingBox': {'Height': 0.4326171875,
                               'Left': 0.46289217472076416,
                               'Top': 0.5634765625,
                               'Width': 0.5371078252792358},
              'Polygon': [{'X': 0.46289217472076416,
                           'Y': 0.5634765625},
                          {'X': 1.0, 'Y': 0.5634765625},
                          {'X': 1.0, 'Y': 0.99609375},
                          {'X': 0.46289217472076416,
                           'Y': 0.99609375}]},

 'Id': 3,
 'Type': 'LINE'},
{'Confidence': 99.35693359375,
 'DetectedText': "IT'S",
 'Geometry': {'BoundingBox': {'Height': 0.09988046437501907,
                               'Left': 0.6684935688972473,
                               'Top': 0.18226495385169983,
                               'Width': 0.1461552083492279},
              'Polygon': [{'X': 0.6684935688972473,
                           'Y': 0.1838926374912262},
                          {'X': 0.8141663074493408,
                           'Y': 0.18226495385169983},
                          {'X': 0.8146487474441528,
                           'Y': 0.28051772713661194},
                          {'X': 0.6689760088920593,
                           'Y': 0.2821454107761383}]},

 'Id': 4,
 'ParentId': 0,
 'Type': 'WORD'},
{'Confidence': 99.6207275390625,
 'DetectedText': 'MONDAY',
 'Geometry': {'BoundingBox': {'Height': 0.11442466825246811,
                               'Left': 0.5566731691360474,
                               'Top': 0.35251158475875854,
                               'Width': 0.39574965834617615},
              'Polygon': [{'X': 0.5566731691360474,
                           'Y': 0.3537122905254364},
                          {'X': 0.9522718787193298,
                           'Y': 0.35251158475875854},
                          {'X': 0.9524227976799011,
                           'Y': 0.4657355546951294},
```

```
        {'X': 0.5568241477012634,
         'Y': 0.46693626046180725}}],
    'Id': 5,
    'ParentId': 1,
    'Type': 'WORD'},
{'Confidence': 99.96778869628906,
 'DetectedText': 'but',
 'Geometry': {'BoundingBox': {'Height': 0.0625,
                              'Left': 0.6402802467346191,
                              'Top': 0.5283203125,
                              'Width': 0.08027780801057816},
              'Polygon': [{'X': 0.6402802467346191,
                           'Y': 0.5283203125},
                          {'X': 0.7205580472946167,
                           'Y': 0.5283203125},
                          {'X': 0.7205580472946167,
                           'Y': 0.5908203125},
                          {'X': 0.6402802467346191,
                           'Y': 0.5908203125}]}],
    'Id': 6,
    'ParentId': 2,
    'Type': 'WORD'},
{'Confidence': 99.26438903808594,
 'DetectedText': 'keep',
 'Geometry': {'BoundingBox': {'Height': 0.0818721204996109,
                              'Left': 0.7344760298728943,
                              'Top': 0.5280686020851135,
                              'Width': 0.10748066753149033},
              'Polygon': [{'X': 0.7349520921707153,
                           'Y': 0.5280686020851135},
                          {'X': 0.8419566750526428,
                           'Y': 0.5295097827911377},
                          {'X': 0.8414806127548218,
                           'Y': 0.6099407076835632},
                          {'X': 0.7344760298728943,
                           'Y': 0.6084995269775391}]}],
    'Id': 7,
    'ParentId': 2,
    'Type': 'WORD'},
{'Confidence': 88.95134735107422,
 'DetectedText': 'Smiling',
 'Geometry': {'BoundingBox': {'Height': 0.4326171875,
                              'Left': 0.46289217472076416,
                              'Top': 0.5634765625,
```

```
        'Width': 0.5371078252792358},
        'Polygon': [{ 'X': 0.46289217472076416,
                       'Y': 0.5634765625},
                    { 'X': 1.0, 'Y': 0.5634765625},
                    { 'X': 1.0, 'Y': 0.99609375},
                    { 'X': 0.46289217472076416,
                       'Y': 0.99609375}]],
        'Id': 8,
        'ParentId': 3,
        'Type': 'WORD'}],
'TextModelVersion': '3.0'}
```

Mendeteksi teks dalam video yang tersimpan

Pendeteksi teks Amazon Rekognition Video dalam video yang disimpan adalah operasi tidak sinkron. Untuk mulai mendeteksi teks, hubungi [StartTextDetection](#). Amazon Rekognition Video menerbitkan status penyelesaian analisis video ke topik Amazon SNS. Jika analisis video berhasil, hubungi [GetTextDetection](#) untuk mendapatkan hasil analisis. Untuk informasi selengkapnya tentang memulai analisis video dan mendapatkan hasilnya, lihat [Memanggil operasi Amazon Rekognition Video](#).

Prosedur ini memperluas kode di [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#). Menggunakan antrean Amazon SQS untuk mendapatkan status penyelesaian permintaan analisis video.

Untuk mendeteksi teks dalam video yang disimpan di bucket Amazon S3 (SDK)

1. Lakukan langkah-langkah pada [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#).
2. Tambahkan kode berikut ke kelas VideoDetect pada langkah 1.

Java

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

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

    NotificationChannel channel= new NotificationChannel()
```

```
        .withSNSTopicArn(snsTopicArn)
        .withRoleArn(roleArn);

    StartTextDetectionRequest req = new StartTextDetectionRequest()
        .withVideo(new Video()
            .withS3Object(new S3Object()
                .withBucket(bucket)
                .withName(video)))
        .withNotificationChannel(channel);

    StartTextDetectionResult startTextDetectionResult =
rek.startTextDetection(req);
    startJobId=startTextDetectionResult.getJobId();
}

private static void GetTextDetectionResults() throws Exception{

    int maxResults=10;
    String paginationToken=null;
    GetTextDetectionResult textDetectionResult=null;

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

        }

        textDetectionResult = rek.getTextDetection(new
GetTextDetectionRequest()
            .withJobId(startJobId)
            .withNextToken(paginationToken)
            .withMaxResults(maxResults));

        VideoMetadata videoMetaData=textDetectionResult.getVideoMetadata();

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

```
//Show text, confidence values
List<TextDetectionResult> textDetections =
textDetectionResult.getTextDetections();

for (TextDetectionResult text: textDetections) {
    long seconds=text.getTimestamp()/1000;
    System.out.println("Sec: " + Long.toString(seconds) + " ");
    TextDetection detectedText=text.getTextDetection();

    System.out.println("Text Detected: " +
detectedText.getDetectedText());
        System.out.println("Confidence: " +
detectedText.getConfidence().toString());
        System.out.println("Id : " + detectedText.getId());
        System.out.println("Parent Id: " + detectedText.getParentId());
        System.out.println("Bounding Box" +
detectedText.getGeometry().getBoundingBox().toString());
        System.out.println("Type: " + detectedText.getType());
        System.out.println();
    }
} while (textDetectionResult !=null && textDetectionResult.getNextToken() !=
null);

}
```

Dalam fungsi main, ganti baris:

```
StartLabelDetection(bucket, video);

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

dengan:

```
StartTextDetection(bucket, video);

if (GetSQSMessagesSuccess()==true)
    GetTextDetectionResults();
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya [di sini](#).

```
//snippet-start:[rekognition.java2.recognize_video_text.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import
    software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;
//snippet-end:[rekognition.java2.recognize_video_text.import]

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

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

        final String usage = "\n" +
            "Usage: " +
            "  <bucket> <video> <topicArn> <roleArn>\n\n" +
```

```
        "Where:\n" +
        "    bucket - The name of the bucket in which the video is located (for
example, (for example, myBucket). \n\n"+
        "    video - The name of video (for example, people.mp4). \n\n" +
        "    topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic. \n\n" +
        "    roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use. \n\n" ;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];

    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
        .build();

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

    startTextLabels(rekClient, channel, bucket, video);
    GetTextResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

// snippet-start:[rekognition.java2.recognize_video_text.main]
public static void startTextLabels(RekognitionClient rekClient,
                                   NotificationChannel channel,
                                   String bucket,
                                   String video) {

    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
```



```
        .name(video)
        .build();

Video vid0b = Video.builder()
    .s3Object(s3Obj)
    .build();

StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
    .jobTag("DetectingLabels")
    .notificationChannel(channel)
    .video(vid0b)
    .build();

StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
startJobId = labelDetectionResponse.jobId();

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

public static void GetTextResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetTextDetectionResponse textDetectionResponse=null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (textDetectionResponse !=null)
                paginationToken = textDetectionResponse.nextToken();

            GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();
```

```
        // Wait until the job succeeds.
        while (!finished) {
            textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
            status = textDetectionResponse.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }

        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null.
        VideoMetadata videoMetaData=textDetectionResponse.videoMetadata();
        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());
        System.out.println("Job");

        List<TextDetectionResult> labels=
textDetectionResponse.textDetections();
        for (TextDetectionResult detectedText: labels) {
            System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
            System.out.println("Id : " +
detectedText.textDetection().id());
            System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
            System.out.println("Type: " +
detectedText.textDetection().type());
            System.out.println("Text: " +
detectedText.textDetection().detectedText());
            System.out.println();
        }

    } while (textDetectionResponse !=null &&
textDetectionResponse.nextToken() != null);
```

```

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

```

Python

```

#Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

def StartTextDetection(self):
    response=self.rek.start_text_detection(Video={'S3Object': {'Bucket':
self.bucket, 'Name': self.video}},
        NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
self.snsTopicArn})

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

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

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

        print('Codec: ' + response['VideoMetadata']['Codec'])

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

        for textDetection in response['TextDetections']:
            text=textDetection['TextDetection']

```

```

        print("Timestamp: " + str(textDetection['Timestamp']))
        print("  Text Detected: " + text['DetectedText'])
        print("  Confidence: " + str(text['Confidence']))
        print ("    Bounding box")
        print ("      Top: " + str(text['Geometry']['BoundingBox']
['Top']))
        print ("      Left: " + str(text['Geometry']['BoundingBox']
['Left']))
        print ("      Width: " + str(text['Geometry']['BoundingBox']
['Width']))
        print ("      Height: " + str(text['Geometry']['BoundingBox']
['Height']))
        print ("  Type: " + str(text['Type']) )
        print()

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

```

Dalam fungsi main, ganti baris:

```

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

```

dengan:

```

analyzer.StartTextDetection()
if analyzer.GetSQSMessageSuccess()==True:
    analyzer.GetTextDetectionResults()

```

CLI

Jalankan AWS CLI perintah berikut untuk mulai mendeteksi teks dalam video.

```

aws rekognition start-text-detection --video '{"S3Object":{"Bucket":"bucket-
name","Name":"video-name"}}'\
--notification-channel '{"SNSTopicArn":"topic-arn","RoleArn":"role-arn"}' \
--region region-name --profile profile-name

```

Perbarui nilai berikut:

- Ubah `bucket-name` dan `video-name` ke nama bucket Amazon S3 dan nama file yang Anda tentukan pada langkah 2.
- Ubah `region-name` ke wilayah AWS yang Anda gunakan.
- Ganti nilai `profile-name` dengan nama profil pengembang Anda.
- Ubah `topic-ARN` ke ARN dari topik Amazon SNS yang Anda buat pada langkah 3 [Mengonfigurasi Amazon Rekognition Video](#).
- Perubahan `role-ARN` ke ARN dari peran layanan IAM yang Anda buat di langkah 7 [Mengonfigurasi Amazon Rekognition Video](#).

Jika Anda mengakses CLI pada perangkat Windows, gunakan tanda kutip ganda alih-alih tanda kutip tunggal dan hindari tanda kutip ganda bagian dalam dengan garis miring terbalik (yaitu `\`) untuk mengatasi kesalahan parser yang mungkin Anda temui. Sebagai contoh, lihat di bawah:

```
aws rekognition start-text-detection --video \  
  "{ \"S3Object\": { \"Bucket\": \"bucket-name\", \"Name\": \"video-name\" } }" \  
  --notification-channel "{ \"SNSTopicArn\": \"topic-arn\", \"RoleArn\": \"role-arn\" }" \  
  --region region-name --profile profile-name
```

Setelah menjalankan contoh kode prosiding, salin yang dikembalikan `jobID` dan berikan ke `GetTextDetection` perintah berikut di bawah ini untuk mendapatkan hasil Anda, ganti `job-id-number` dengan yang `jobID` Anda terima sebelumnya:

```
aws rekognition get-text-detection --job-id job-id-number --profile profile-name
```

Note

Jika Anda sudah menjalankan contoh video selain [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#), kode yang akan diganti mungkin berbeda.

3. Jalankan kode tersebut. Teks yang terdeteksi dalam video ditampilkan dalam daftar.

Filter

Filter adalah parameter permintaan opsional yang dapat digunakan ketika Anda memanggil `StartTextDetection`. Pemfilteran berdasarkan wilayah teks, ukuran, dan skor kepercayaan memberikan Anda fleksibilitas tambahan untuk mengendalikan output deteksi teks Anda. Dengan menggunakan wilayah yang diminati, Anda dengan mudah membatasi deteksi teks ke wilayah yang relevan, misalnya, wilayah ketiga terbawah untuk grafik atau sudut kiri atas untuk membaca papan skor dalam pertandingan sepak bola. Filter ukuran kotak pembatas kata dapat digunakan untuk menghindari teks latar belakang kecil yang mungkin ramai atau tidak relevan. Dan terakhir, filter kepercayaan kata memungkinkan Anda untuk menghapus hasil yang mungkin tidak dapat diandalkan karena buram atau tercoreng.

Untuk informasi mengenai nilai filter, lihat [DetectTextFilters](#).

Anda dapat menggunakan filter berikut:

- `MinConfidence`—Menetapkan tingkat kepercayaan deteksi kata. Kata-kata dengan kepercayaan deteksi di bawah tingkat ini dikeluarkan dari hasil. Nilai harus antara 0 hingga 100.
- `MinBoundingBoxWidth`— Mengatur lebar minimum kotak pembatas kata. Kata-kata dengan kotak pembatas yang lebih kecil dari nilai ini dikeluarkan dari hasil. Nilainya relatif terhadap lebar bingkai video.
- `MinBoundingBoxHeight`— Mengatur tinggi minimum kotak pembatas kata. Kata-kata dengan tinggi kotak pembatas kurang dari nilai ini dikeluarkan dari hasil. Nilai relatif terhadap tinggi bingkai video.
- `RegionsOfInterest`— Membatasi deteksi ke wilayah tertentu dari bingkai. Nilainya relatif terhadap dimensi bingkai. Untuk objek yang hanya sebagian di dalam wilayah, responsnya tidak ditentukan.

GetTextDetection respon

`GetTextDetection` mengembalikan sebuah array (`TextDetectionResults`) yang berisi informasi tentang teks yang terdeteksi dalam video. Sebuah elemen array, [TextDetection](#), ada untuk setiap kali kata atau baris terdeteksi dalam video. Elemen array diurutkan berdasarkan waktu (dalam milidetik) sejak awal video.

Berikut ini adalah respons JSON parsial dari `GetTextDetection`. Dalam respons, perhatikan hal berikut:

- Informasi teks - Elemen `TextDetectionResult` array berisi informasi tentang teks yang terdeteksi ([TextDetection](#)) dan waktu teks terdeteksi dalam video (`Timestamp`).
- Informasi halaman – Contoh tersebut menunjukkan satu halaman informasi deteksi teks. Anda dapat menentukan seberapa banyak elemen teks yang akan dikembalikan dalam parameter input `MaxResults` untuk `GetTextDetection`. Jika ada lebih banyak hasil daripada `MaxResults`, atau ada lebih banyak hasil daripada maksimum default, `GetTextDetection` mengembalikan token (`NextToken`) yang digunakan untuk mendapatkan halaman hasil berikutnya. Untuk informasi selengkapnya, lihat [Mendapatkan hasil analisis Amazon Rekognition Video](#).
- Informasi video – Respons mencakup informasi tentang format video (`VideoMetadata`) di setiap halaman informasi yang dikembalikan oleh `GetTextDetection`.

```
{
  "JobStatus": "SUCCEEDED",
  "VideoMetadata": {
    "Codec": "h264",
    "DurationMillis": 174441,
    "Format": "QuickTime / MOV",
    "FrameRate": 29.970029830932617,
    "FrameHeight": 480,
    "FrameWidth": 854
  },
  "TextDetections": [
    {
      "Timestamp": 967,
      "TextDetection": {
        "DetectedText": "Twinkle Twinkle Little Star",
        "Type": "LINE",
        "Id": 0,
        "Confidence": 99.91780090332031,
        "Geometry": {
          "BoundingBox": {
            "Width": 0.8337579369544983,
            "Height": 0.08365312218666077,
            "Left": 0.08313830941915512,
            "Top": 0.4663468301296234
          },
          "Polygon": [
            {
              "X": 0.08313830941915512,
```

```

        "Y": 0.4663468301296234
      },
      {
        "X": 0.9168962240219116,
        "Y": 0.4674469828605652
      },
      {
        "X": 0.916861355304718,
        "Y": 0.5511001348495483
      },
      {
        "X": 0.08310343325138092,
        "Y": 0.5499999523162842
      }
    ]
  }
},
{
  "Timestamp": 967,
  "TextDetection": {
    "DetectedText": "Twinkle",
    "Type": "WORD",
    "Id": 1,
    "ParentId": 0,
    "Confidence": 99.98338317871094,
    "Geometry": {
      "BoundingBox": {
        "Width": 0.2423887550830841,
        "Height": 0.0833333358168602,
        "Left": 0.08313817530870438,
        "Top": 0.46666666865348816
      },
      "Polygon": [
        {
          "X": 0.08313817530870438,
          "Y": 0.46666666865348816
        },
        {
          "X": 0.3255269229412079,
          "Y": 0.46666666865348816
        },
        {
          "X": 0.3255269229412079,

```



```
        "Y": 0.550000011920929
      },
      {
        "X": 0.08313817530870438,
        "Y": 0.550000011920929
      }
    ]
  }
},
{
  "Timestamp": 967,
  "TextDetection": {
    "DetectedText": "Twinkle",
    "Type": "WORD",
    "Id": 2,
    "ParentId": 0,
    "Confidence": 99.982666015625,
    "Geometry": {
      "BoundingBox": {
        "Width": 0.2423887550830841,
        "Height": 0.08124999701976776,
        "Left": 0.3454332649707794,
        "Top": 0.46875
      },
      "Polygon": [
        {
          "X": 0.3454332649707794,
          "Y": 0.46875
        },
        {
          "X": 0.5878220200538635,
          "Y": 0.46875
        },
        {
          "X": 0.5878220200538635,
          "Y": 0.550000011920929
        },
        {
          "X": 0.3454332649707794,
          "Y": 0.550000011920929
        }
      ]
    }
  }
}
```

```
    }
  },
  {
    "Timestamp": 967,
    "TextDetection": {
      "DetectedText": "Little",
      "Type": "WORD",
      "Id": 3,
      "ParentId": 0,
      "Confidence": 99.8787612915039,
      "Geometry": {
        "BoundingBox": {
          "Width": 0.16627635061740875,
          "Height": 0.08124999701976776,
          "Left": 0.6053864359855652,
          "Top": 0.46875
        },
        "Polygon": [
          {
            "X": 0.6053864359855652,
            "Y": 0.46875
          },
          {
            "X": 0.7716627717018127,
            "Y": 0.46875
          },
          {
            "X": 0.7716627717018127,
            "Y": 0.550000011920929
          },
          {
            "X": 0.6053864359855652,
            "Y": 0.550000011920929
          }
        ]
      }
    }
  },
  {
    "Timestamp": 967,
    "TextDetection": {
      "DetectedText": "Star",
      "Type": "WORD",
      "Id": 4,
```

```
    "ParentId": 0,
    "Confidence": 99.82640075683594,
    "Geometry": {
      "BoundingBox": {
        "Width": 0.12997658550739288,
        "Height": 0.08124999701976776,
        "Left": 0.7868852615356445,
        "Top": 0.46875
      },
      "Polygon": [
        {
          "X": 0.7868852615356445,
          "Y": 0.46875
        },
        {
          "X": 0.9168618321418762,
          "Y": 0.46875
        },
        {
          "X": 0.9168618321418762,
          "Y": 0.550000011920929
        },
        {
          "X": 0.7868852615356445,
          "Y": 0.550000011920929
        }
      ]
    }
  },
  "NextToken": "NiHpGbZFnkM/S8kLcukMni15wb05iKtquu/Mwc+Qg1LVlMjjKN0D0Z0GusSPg7TONLe+0Z3P",
  "TextModelVersion": "3.0"
}
```

Mendeteksi segmen video dalam video yang tersimpan

Amazon Rekognition Video menyediakan API yang mengidentifikasi segmen video yang berguna, seperti bingkai hitam dan kredit akhir.

Penampil menonton lebih banyak konten dari sebelumnya. Secara khusus, platform Over-The-Top (OTT) dan Video Sesuai Permintaan (VOD) menyediakan beragam pilihan konten kapan saja, di mana saja, dan di layar apa pun. Dengan volume konten yang terus bertambah, perusahaan media menghadapi tantangan dalam mempersiapkan dan mengelola konten mereka. Hal ini sangat penting untuk memberikan pengalaman menonton berkualitas tinggi dan konten monetisasi yang lebih baik. Saat ini, perusahaan menggunakan tim besar yang terdiri dari tenaga kerja manusia terlatih untuk melakukan tugas-tugas seperti berikut.

- Menemukan di mana kredit pembukaan dan akhir berada dalam sepotong konten
- Memilih tempat yang tepat untuk menyisipkan iklan, seperti dalam urutan bingkai hitam diam
- Memecah video menjadi klip yang lebih kecil untuk pengindeksan yang lebih baik

Proses manual ini mahal, lambat, dan tidak dapat disesuaikan dengan volume konten yang diproduksi, dilisensikan, dan diambil dari arsip setiap hari.

Anda dapat menggunakan Amazon Rekognition Video untuk mengotomatiskan tugas analisis media operasional menggunakan API deteksi segmen video yang dikelola sepenuhnya dan dibuat khusus yang didukung oleh pembelajaran mesin (ML). Dengan menggunakan API segmen Amazon Rekognition Video, Anda dapat dengan mudah menganalisis volume video besar dan mendeteksi penanda seperti bingkai hitam atau perubahan sorotan. Anda mendapatkan kode waktu, stempel waktu, dan nomor bingkai SMPTE (Society of Motion Picture and Television Engineers) dari setiap deteksi. Tidak diperlukan pengalaman ML.

Amazon Rekognition Video menganalisis video yang disimpan dalam bucket Amazon Simple Storage Service (Amazon S3). Kode waktu SMPTE yang dikembalikan akurat dalam bingkai - Amazon Rekognition Video menyediakan nomor bingkai yang tepat dari segmen video yang terdeteksi, dan menangani berbagai format frame rate video secara otomatis. Anda dapat menggunakan metadata akurat bingkai dari Amazon Rekognition Video untuk mengotomatiskan tugas tertentu sepenuhnya, atau untuk secara signifikan mengurangi beban kerja peninjauan operator manusia terlatih, sehingga mereka dapat fokus pada pekerjaan yang lebih kreatif. Anda dapat melakukan tugas-tugas seperti menyiapkan konten, menyisipkan iklan, dan menambahkan “binge-marker” ke konten dalam skala besar di cloud.

Untuk informasi lengkap tentang harga, lihat [harga Amazon Rekognition](#).

Deteksi segmen Video Rekognition Amazon mendukung dua jenis tugas segmentasi — deteksi dan. [Isyarat teknis Deteksi sorotan](#)

Topik

- [Isyarat teknis](#)
- [Deteksi sorotan](#)
- [Tentang API deteksi Segmen Video Rekognition Amazon](#)
- [Menggunakan API Segmen Amazon Rekognition](#)
- [Contoh: Mendeteksi segmen dalam video yang tersimpan](#)

Isyarat teknis

Isyarat teknis mengidentifikasi bingkai hitam, bilah warna, kredit pembuka, kredit akhir, logo studio, dan konten program utama dalam video.

Frame hitam

Video sering berisi bingkai hitam kosong tanpa audio yang digunakan sebagai isyarat untuk menyisipkan iklan, atau untuk menandai akhir segmen program, seperti adegan atau kredit pembuka. Dengan Amazon Rekognition Video, Anda dapat mendeteksi urutan bingkai hitam untuk mengotomatiskan penyisipan iklan, konten paket untuk VOD, dan membatasi berbagai segmen atau adegan program. Bingkai hitam dengan audio (seperti memudar atau pengisi suara) dianggap sebagai konten dan tidak dikembalikan.

Kredit

Amazon Rekognition Video dapat secara otomatis mengidentifikasi frame yang tepat di mana kredit pembukaan dan penutupan dimulai dan berakhir untuk film atau acara TV. Dengan informasi ini, Anda dapat menghasilkan “penanda pesta” atau petunjuk penampil interaktif, seperti “Episode Berikutnya” atau “Lewati Intro,” dalam aplikasi video on demand (VOD). Anda juga dapat mendeteksi bingkai pertama dan terakhir dari konten program dalam video. Amazon Rekognition Video dilatih untuk menangani berbagai macam gaya kredit pembukaan dan akhir mulai dari kredit bergulir sederhana hingga kredit yang lebih menantang di samping konten.

Diagram warna

Amazon Rekognition Video memungkinkan Anda untuk mendeteksi bagian video yang menampilkan diagram warna SMPTE, yang merupakan seperangkat warna yang ditampilkan dalam pola tertentu untuk memastikan warna dikalibrasi dengan benar pada monitor siaran, program, dan kamera. Untuk informasi selengkapnya tentang diagram warna SMPTE, lihat [Bilah warna SMPTE](#). Metadata ini berguna untuk mempersiapkan konten untuk aplikasi VOD dengan menghapus segmen diagram warna dari konten, atau untuk mendeteksi masalah seperti hilangnya sinyal siaran dalam rekaman, ketika diagram warna ditampilkan terus menerus sebagai sinyal default bukan konten.

Papan tulis

Slate adalah bagian dari video, biasanya di dekat awal, yang berisi metadata teks tentang episode, studio, format video, saluran audio, dan banyak lagi. Amazon Rekognition Video dapat mengidentifikasi awal dan akhir papan tulis, sehingga mudah untuk menggunakan metadata teks atau menghapus batu tulis saat menyiapkan konten untuk tampilan akhir.

Logo studio

Logo studio adalah urutan yang menunjukkan logo atau lambang studio produksi yang terlibat dalam pembuatan pertunjukan. Amazon Rekognition Video dapat mendeteksi urutan ini sehingga pengguna dapat memeriksanya untuk mengidentifikasi studio.

Daftar isi

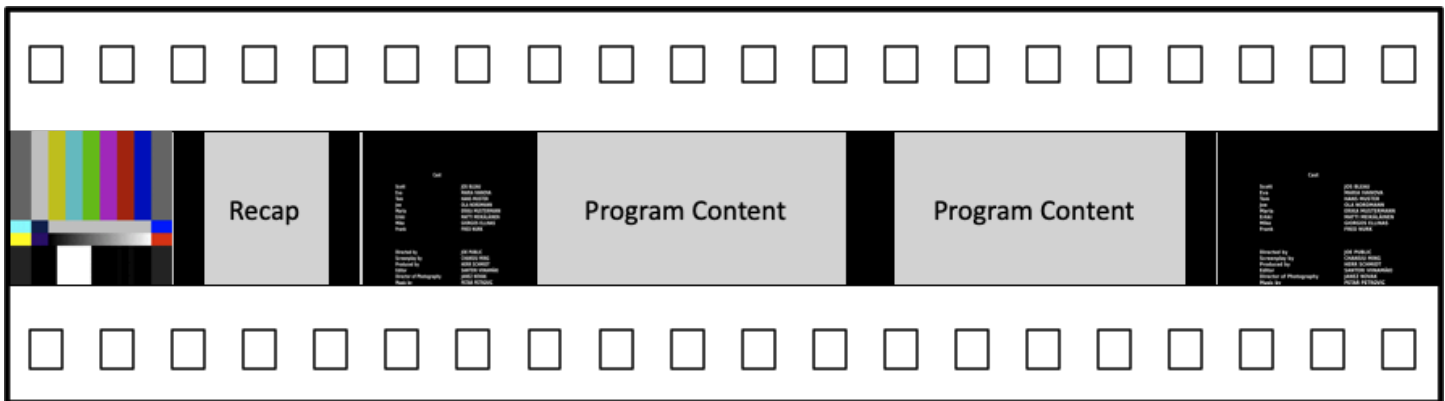
Konten adalah bagian dari acara TV atau film yang berisi program atau elemen terkait. Bingkai hitam, kredit, bilah warna, papan tulis, dan logo studio tidak dianggap konten. Amazon Rekognition Video dapat mendeteksi awal dan akhir setiap segmen konten dalam video, sehingga Anda dapat menemukan waktu berjalan program atau segmen tertentu.

Segmen konten mencakup, tetapi tidak terbatas pada, hal-hal berikut:

- Adegan program antara dua jeda iklan
- Rekap singkat dari episode sebelumnya di awal video
- Konten bonus pasca-kredit
- Konten “tanpa teks”, seperti sekumpulan semua adegan program yang awalnya berisi teks overlay, tetapi teks telah dihapus untuk mendukung terjemahan ke bahasa lain.

Setelah Amazon Rekognition Video selesai mendeteksi semua segmen konten, Anda dapat menerapkan pengetahuan domain atau mengirimkannya untuk ditinjau manusia untuk mengkategorikan lebih lanjut setiap segmen. Misalnya, jika Anda menggunakan video yang selalu dimulai dengan rekap, Anda dapat mengkategorikan segmen konten pertama sebagai rekap.

Diagram berikut menunjukkan segmen isyarat teknis pada garis waktu acara atau film. Perhatikan bilah warna dan kredit pembuka, segmen konten seperti rekap dan program utama, bingkai hitam di seluruh video, dan kredit akhir.



Deteksi sorotan

Sorotan adalah serangkaian citra berturut-turut yang saling terkait, yang diambil secara bersebelahan oleh satu kamera dan mewakili tindakan berkesinambungan dalam ruang dan waktu. Dengan Amazon Rekognition Video, Anda dapat mendeteksi awal, akhir, dan durasi setiap sorotan, serta hitungan untuk semua sorotan dalam sepotong konten. Anda dapat menggunakan metadata sorotan untuk tugas-tugas seperti berikut.

- Membuat video promosi menggunakan citra yang dipilih.
- Memasukkan iklan di lokasi yang tidak mengganggu pengalaman penampil, seperti tengah sorotan saat seseorang berbicara.
- Menghasilkan set citra mini pratinjau yang menghindari konten transisi di antara sorotan.

Deteksi sorotan ditandai pada bingkai yang tepat, tempat terdapat potongan kasar pada kamera yang berbeda. Jika ada transisi halus dari satu kamera ke kamera lain, Amazon Rekognition Video menghilangkan transisi. Hal ini memastikan bahwa waktu mulai dan akhir sorotan tidak memasukan bagian tanpa konten yang sebenarnya.

Diagram berikut menggambarkan segmen deteksi sorotan pada strip film. Perhatikan bahwa setiap sorotan diidentifikasi dengan potongan dari satu sudut kamera atau lokasi ke lokasi berikutnya.



Tentang API deteksi Segmen Video Rekognition Amazon

Untuk mengelompokkan video tersimpan, Anda menggunakan operasi asinkron [StartSegmentDetection](#) dan [GetSegmentDetection](#) API untuk memulai pekerjaan segmentasi dan mengambil hasilnya. Deteksi segmen menerima video yang disimpan dalam bucket Amazon S3 dan mengembalikan output JSON. Anda dapat memilih untuk mendeteksi hanya isyarat teknis, hanya memotret perubahan, atau keduanya bersama-sama dengan mengonfigurasi permintaan [StartSegmentDetection](#) API. Anda juga dapat memfilter segmen yang terdeteksi dengan mengatur ambang batas untuk kepercayaan prediksi minimum. Untuk informasi selengkapnya, lihat [Menggunakan API Segmen Amazon Rekognition](#). Untuk kode sampel, lihat [Contoh: Mendeteksi segmen dalam video yang tersimpan](#).

Menggunakan API Segmen Amazon Rekognition

Deteksi segmen Amazon Rekognition Video dalam video yang disimpan adalah operasi Amazon Rekognition Video yang tidak sinkron. API Segmen Amazon Rekognition adalah API komposit tempat Anda memilih tipe analisis (isyarat teknis atau deteksi sorotan) dari satu panggilan API. Untuk informasi tentang memanggil operasi yang tidak sinkron, lihat [Memanggil operasi Amazon Rekognition Video](#).

Topik

- [Memulai analisis segmen](#)
- [Mendapatkan hasil analisis segmen](#)

Memulai analisis segmen

Untuk memulai deteksi segmen dalam panggilan video yang disimpan [StartSegmentDetection](#). Parameter input yang sama seperti operasi Amazon Rekognition Video lainnya dengan penambahan seleksi tipe segmen dan pemfilteran hasil. Untuk informasi selengkapnya, lihat [Memulai analisis video](#).

Berikut ini adalah contoh JSON diteruskan oleh `StartSegmentDetection`. Permintaan menyatakan bahwa kedua isyarat teknis dan segmen deteksi sorotan terdeteksi. Filter yang berbeda untuk kepercayaan pendeteksian minimum diminta untuk segmen isyarat teknis (90%) dan segmen deteksi bidikan (80%).

```
{
  "Video": {
    "S3Object": {
      "Bucket": "test_files",
      "Name": "test_file.mp4"
    }
  },
  "SegmentTypes": ["TECHNICAL_CUES", "SHOT"]
  "Filters": {
    "TechnicalCueFilter": {
      "MinSegmentConfidence": 90,
      "BlackFrame": {
        "MaxPixelThreshold": 0.1,
        "MinCoveragePercentage": 95
      }
    },
    "ShotFilter": {
      "MinSegmentConfidence": 60
    }
  }
}
```

Memilih tipe segmen

Gunakan parameter input array `SegmentTypes` untuk mendeteksi isyarat teknis dan/atau segmen deteksi sorotan dalam video input.

- `TECHNICAL_CUE` — mengidentifikasi stempel waktu akurat bingkai untuk awal, akhir, dan durasi isyarat teknis (bingkai hitam, bilah warna, kredit pembuka, kredit akhir, logo studio, dan konten

program utama) yang terdeteksi dalam video. Misalnya, Anda dapat menggunakan isyarat teknis untuk menemukan awal kredit akhir. Untuk informasi selengkapnya, lihat [Isyarat teknis](#).

- SHOT — Mengidentifikasi awal, akhir, dan durasi sorotan. Misalnya, Anda dapat menggunakan deteksi pengambilan citra untuk mengidentifikasi calon sorotan untuk pengeditan akhir video. Untuk informasi selengkapnya, lihat [Deteksi sorotan](#).

Memfilter hasil analisis

Anda dapat menggunakan parameter input `Filters` ([StartSegmentDetectionFilters](#)) untuk menentukan kepercayaan deteksi minimum yang dikembalikan dalam respons. Di dalam `Filters`, gunakan `ShotFilter` ([StartShotDetectionFilter](#)) untuk memfilter bidikan yang terdeteksi. Gunakan `TechnicalCueFilter` ([StartTechnicalCueDetectionFilter](#)) untuk memfilter isyarat teknis.

Untuk kode sampel, lihat [Contoh: Mendeteksi segmen dalam video yang tersimpan](#).

Mendapatkan hasil analisis segmen

Amazon Rekognition Video menerbitkan status penyelesaian analisis video ke topik Amazon Simple Notification Service. Jika analisis video berhasil, hubungi [GetSegmentDetection](#) untuk mendapatkan hasil analisis video.

Berikut ini adalah contoh permintaan `GetSegmentDetection`. `JobId` adalah pengenal tugas yang dikembalikan dari panggilan ke `StartSegmentDetection`. Untuk informasi tentang mengatur parameter, lihat [Mendapatkan hasil analisis Amazon Rekognition Video](#).

```
{
  "JobId": "270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1784ca90fc180e3",
  "MaxResults": 10,
  "NextToken": "XfXnZKiyM0GDhzBzYUhS5puM+g1IgezqFeYpv/H/+5noP/LmM57FitUAwSQ5D6G4AB/PNwo1rw=="
}
```

`GetSegmentDetection` memberikan hasil untuk analisis yang diminta dan informasi umum tentang video yang disimpan.

Informasi umum

`GetSegmentDetection` mengembalikan informasi umum berikut.

- Informasi audio — Respons mencakup metadata audio dalam array, `AudioMetadata`, objek. [AudioMetadata](#) Ada beberapa pengaliran audio. Setiap objek `AudioMetadata` berisi metadata untuk aliran audio tunggal. Informasi audio di objek `AudioMetadata` termasuk codec audio, jumlah saluran audio, durasi pengaliran audio, dan laju sampel. Metadata audio dikembalikan di setiap halaman informasi yang dikembalikan oleh `GetSegmentDetection`.
- Informasi video — Saat ini, Amazon Rekognition Video mengembalikan satu [VideoMetadata](#) objek dalam array. `VideoMetadata` Objek berisi informasi tentang pengaliran video dalam file input yang dipilih Amazon Rekognition Video untuk dianalisis. Objek `VideoMetadata` mencakup codec video, format video dan informasi lainnya. Metadata video dikembalikan di setiap halaman informasi yang dikembalikan oleh `GetSegmentDetection`.
- Informasi halaman — Contoh tersebut menunjukkan satu halaman informasi segmen. Anda dapat menentukan berapa banyak elemen yang bisa dikembalikan dalam parameter input `MaxResults` untuk `GetSegmentDetection`. Jika hasilnya lebih dari `MaxResults`, `GetSegmentDetection` mengembalikan sebuah token (`NextToken`) yang digunakan untuk mendapatkan halaman hasil berikutnya. Untuk informasi selengkapnya, lihat [Mendapatkan hasil analisis Amazon Rekognition Video](#).
- Meminta informasi — tipe analisis yang diminta dalam panggilan ke `StartSegmentDetection` dikembalikan dalam bidang `SelectedSegmentTypes`.

Segmen

Isyarat teknis dan informasi bidikan yang terdeteksi dalam video dikembalikan dalam `arraySegments`, [SegmentDetection](#) objek. Array diurutkan berdasarkan tipe segmen (`TECHNICAL_CUE` atau `SHOT`) yang ditentukan dalam parameter input `SegmentTypes` dari `StartSegmentDetection`. Dalam setiap tipe segmen, array diurutkan berdasarkan nilai-nilai timestamp. Setiap objek `SegmentDetection` meliputi informasi tentang tipe segmen yang terdeteksi (isyarat Teknis atau deteksi sorotan) dan informasi umum, seperti waktu mulai, waktu akhir, dan durasi segmen.

Informasi waktu dikembalikan dalam tiga format.

- Milidetik

Jumlah milidetik sejak awal video. Format bidang `DurationMillis`, `StartTimestampMillis`, dan `EndTimestampMillis` adalah milidetik.

- Timecode

Format Amazon Rekognition Video timecodes adalah [SMPTTE](#) yang memungkinkan setiap frame video memiliki nilai kode waktu yang unik. Formatnya adalah `jj:mm:dd:bingkai`. Misalnya, nilai kode waktu `01:05:40:07`, akan dibaca sebagai satu jam, lima menit, empat puluh detik dan tujuh frame. Kasus penggunaan kecepatan [frame yang menurun](#) didukung oleh Amazon Rekognition Video. Format timecode tingkat pengurangan adalah `jj:mm:dd;bingkai`. Format bidang `DurationSMPTTE`, `StartTimecodeSMPTTE`, dan `EndTimecodeSMPTTE` adalah timecode.

- Penghitung Bingkai

Durasi setiap segmen video juga dinyatakan dengan jumlah frame. Bidang `StartFrameNumber` memberikan nomor bingkai di awal segmen video, dan `EndFrameNumber` memberikan nomor bingkai di akhir segmen video. `DurationFrames` memberikan jumlah total frame dalam segmen video. Nilai-nilai ini dihitung menggunakan indeks bingkai yang dimulai dengan 0.

Anda dapat menggunakan bidang `SegmentType` untuk menentukan tipe segmen yang dikembalikan oleh Amazon Rekognition Video.

- Isyarat Teknis — **TechnicalCueSegment** bidang adalah [TechnicalCueSegment](#) objek yang berisi kepercayaan deteksi dan jenis isyarat teknis. Jenis isyarat teknis adalah `ColorBars`, `EndCredits`, `BlackFramesOpeningCredits`, `StudioLogoSlate`, dan `Content`.
- Shot — `ShotSegment` bidang adalah [ShotSegment](#) objek yang berisi kepercayaan deteksi dan pengenalan untuk segmen pemotretan dalam video.

Berikut ini adalah contoh respons JSON dari `GetSegmentDetection`.

```
{
  "SelectedSegmentTypes": [
    {
      "ModelVersion": "2.0",
      "Type": "SHOT"
    },
    {
      "ModelVersion": "2.0",
      "Type": "TECHNICAL_CUE"
    }
  ],
  "Segments": [
    {
```

```
    "DurationFrames": 299,
    "DurationSMPTE": "00:00:09;29",
    "StartFrameNumber": 0,
    "EndFrameNumber": 299,
    "EndTimecodeSMPTE": "00:00:09;29",
    "EndTimestampMillis": 9976,
    "StartTimeStampMillis": 0,
    "DurationMillis": 9976,
    "StartTimecodeSMPTE": "00:00:00;00",
    "Type": "TECHNICAL_CUE",
    "TechnicalCueSegment": {
      "Confidence": 90.45006561279297,
      "Type": "BlackFrames"
    }
  },
  {
    "DurationFrames": 150,
    "DurationSMPTE": "00:00:05;00",
    "StartFrameNumber": 299,
    "EndFrameNumber": 449,
    "EndTimecodeSMPTE": "00:00:14;29",
    "EndTimestampMillis": 14981,
    "StartTimeStampMillis": 9976,
    "DurationMillis": 5005,
    "StartTimecodeSMPTE": "00:00:09;29",
    "Type": "TECHNICAL_CUE",
    "TechnicalCueSegment": {
      "Confidence": 100.0,
      "Type": "Content"
    }
  },
  {
    "DurationFrames": 299,
    "ShotSegment": {
      "Index": 0,
      "Confidence": 99.9982681274414
    },
    "DurationSMPTE": "00:00:09;29",
    "StartFrameNumber": 0,
    "EndFrameNumber": 299,
    "EndTimecodeSMPTE": "00:00:09;29",
    "EndTimestampMillis": 9976,
    "StartTimeStampMillis": 0,
    "DurationMillis": 9976,
```

```
    "StartTimecodeSMPTE": "00:00:00;00",
    "Type": "SHOT"
  },
  {
    "DurationFrames": 149,
    "ShotSegment": {
      "Index": 1,
      "Confidence": 99.9982681274414
    },
    "DurationSMPTE": "00:00:04;29",
    "StartFrameNumber": 300,
    "EndFrameNumber": 449,
    "EndTimecodeSMPTE": "00:00:14;29",
    "EndTimestampMillis": 14981,
    "StartTimestampMillis": 10010,
    "DurationMillis": 4971,
    "StartTimecodeSMPTE": "00:00:10;00",
    "Type": "SHOT"
  }
],
"JobStatus": "SUCCEEDED",
"VideoMetadata": [
  {
    "Format": "QuickTime / MOV",
    "FrameRate": 29.970029830932617,
    "Codec": "h264",
    "DurationMillis": 15015,
    "FrameHeight": 1080,
    "FrameWidth": 1920,
    "ColorRange": "LIMITED"
  }
],
"AudioMetadata": [
  {
    "NumberOfChannels": 1,
    "SampleRate": 48000,
    "Codec": "aac",
    "DurationMillis": 15007
  }
]
}
```

Untuk kode sampel, lihat [Contoh: Mendeteksi segmen dalam video yang tersimpan](#).

Contoh: Mendeteksi segmen dalam video yang tersimpan

Prosedur berikut menunjukkan cara mendeteksi segmen isyarat teknis dan segmen deteksi sorotan dalam video yang disimpan dalam bucket Amazon S3. Prosedur ini juga menunjukkan cara memfilter segmen yang terdeteksi berdasarkan kepercayaan yang dimiliki Amazon Rekognition Video tentang keakuratan deteksi.

Contoh meluas pada kode di [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#) yang menggunakan antrean Amazon Simple Queue Service untuk mendapatkan status penyelesaian permintaan analisis video.

Untuk mendeteksi segmen dalam video yang disimpan dalam bucket Amazon S3 (SDK)

1. Lakukan [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#).
2. Tambahkan berikut ke kode yang Anda gunakan pada langkah 1.

Java

1. Tambahkan impor berikut.

```
import com.amazonaws.services.rekognition.model.GetSegmentDetectionRequest;
import com.amazonaws.services.rekognition.model.GetSegmentDetectionResult;
import com.amazonaws.services.rekognition.model.SegmentDetection;
import com.amazonaws.services.rekognition.model.SegmentType;
import com.amazonaws.services.rekognition.model.SegmentTypeInfo;
import com.amazonaws.services.rekognition.model.ShotSegment;
import
    com.amazonaws.services.rekognition.model.StartSegmentDetectionFilters;
import
    com.amazonaws.services.rekognition.model.StartSegmentDetectionRequest;
import com.amazonaws.services.rekognition.model.StartSegmentDetectionResult;
import com.amazonaws.services.rekognition.model.StartShotDetectionFilter;
import
    com.amazonaws.services.rekognition.model.StartTechnicalCueDetectionFilter;
import com.amazonaws.services.rekognition.model.TechnicalCueSegment;
import com.amazonaws.services.rekognition.model.AudioMetadata;
```

2. Tambahkan kode berikut ke kelas VideoDetect.

```

//Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights
Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

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

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

    float minTechnicalCueConfidence = 80F;
    float minShotConfidence = 80F;

    StartSegmentDetectionRequest req = new
StartSegmentDetectionRequest()
        .withVideo(new Video()
            .withS3Object(new S3Object()
                .withBucket(bucket)
                .withName(video)))
        .withSegmentTypes("TECHNICAL_CUE" , "SHOT")
        .withFilters(new StartSegmentDetectionFilters()
            .withTechnicalCueFilter(new
StartTechnicalCueDetectionFilter()

.withMinSegmentConfidence(minTechnicalCueConfidence))
            .withShotFilter(new StartShotDetectionFilter()

.withMinSegmentConfidence(minShotConfidence)))
        .withJobTag("DetectingVideoSegments")
        .withNotificationChannel(channel);

    StartSegmentDetectionResult startLabelDetectionResult =
rek.startSegmentDetection(req);
    startJobId=startLabelDetectionResult.getJobId();

}

private static void GetSegmentDetectionResults() throws Exception{

    int maxResults=10;

```



```
String paginationToken=null;
GetSegmentDetectionResult segmentDetectionResult=null;
Boolean firstTime=true;

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

    GetSegmentDetectionRequest segmentDetectionRequest= new
GetSegmentDetectionRequest()
        .withJobId(startJobId)
        .withMaxResults(maxResults)
        .withNextToken(paginationToken);

    segmentDetectionResult =
rek.getSegmentDetection(segmentDetectionRequest);

    if(firstTime) {
        System.out.println("\nStatus\n-----");
        System.out.println(segmentDetectionResult.getJobStatus());
        System.out.println("\nRequested features
\n-----");
        for (SegmentTypeInfo requestedFeatures :
segmentDetectionResult.getSelectedSegmentTypes()) {
            System.out.println(requestedFeatures.getType());
        }
        int count=1;
        List<VideoMetadata> videoMeta dataList =
segmentDetectionResult.getVideoMetadata();
        System.out.println("\nVideo Streams\n-----");
        for (VideoMetadata videoMetaData: videoMetaList) {
            System.out.println("Stream: " + count++);
            System.out.println("\tFormat: " +
videoMetaData.getFormat());
            System.out.println("\tCodec: " +
videoMetaData.getCodec());
            System.out.println("\tDuration: " +
videoMetaData.getDurationMillis());
            System.out.println("\tFrameRate: " +
videoMetaData.getFrameRate());
        }
    }
}
```

```
        List<AudioMetadata> audioMetadataList =
segmentDetectionResult.getAudioMetadata();
        System.out.println("\nAudio streams\n-----");

        count=1;
        for (AudioMetadata audioMetaData: audioMetadataList) {
            System.out.println("Stream: " + count++);
            System.out.println("\tSample Rate: " +
audioMetaData.getSampleRate());
            System.out.println("\tCodec: " +
audioMetaData.getCodec());
            System.out.println("\tDuration: " +
audioMetaData.getDurationMillis());
            System.out.println("\tNumber of Channels: " +
audioMetaData.getNumberOfChannels());
        }
        System.out.println("\nSegments\n-----");

        firstTime=false;
    }

    //Show segment information

    List<SegmentDetection> detectedSegments=
segmentDetectionResult.getSegments();

    for (SegmentDetection detectedSegment: detectedSegments) {

        if
(detectedSegment.getType().contains(SegmentType.TECHNICAL_CUE.toString()))
        {
            System.out.println("Technical Cue");
            TechnicalCueSegment
segmentCue=detectedSegment.getTechnicalCueSegment();
            System.out.println("\tType: " + segmentCue.getType());
            System.out.println("\tConfidence: " +
segmentCue.getConfidence().toString());
        }
        if
(detectedSegment.getType().contains(SegmentType.SHOT.toString())) {
            System.out.println("Shot");
        }
    }
}
```

```
        ShotSegment
        segmentShot=detectedSegment.getShotSegment();
            System.out.println("\tIndex " +
segmentShot.getIndex());
            System.out.println("\tConfidence: " +
segmentShot.getConfidence().toString());
        }
        long seconds=detectedSegment.getDurationMillis();
        System.out.println("\tDuration : " + Long.toString(seconds)
+ " milliseconds");
        System.out.println("\tStart time code: " +
detectedSegment.getStartTimecodeSMPTE());
        System.out.println("\tEnd time code: " +
detectedSegment.getEndTimecodeSMPTE());
        System.out.println("\tDuration time code: " +
detectedSegment.getDurationSMPTE());
        System.out.println();
    }

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

}
```

3. Dalam fungsi main, ganti baris:

```
StartLabelDetection(bucket, video);

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

dengan:

```
StartSegmentDetection(bucket, video);

if (GetSQSMessagesSuccess()==true)
    GetSegmentDetectionResults();
```

Java V2

```
//snippet-start:[rekognition.java2.recognize_video_text.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import
    software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;
//snippet-end:[rekognition.java2.recognize_video_text.import]

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

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

        final String usage = "\n" +
            "Usage: " +
            "  <bucket> <video> <topicArn> <roleArn>\n\n" +
            "Where:\n" +
```

```
    " bucket - The name of the bucket in which the video is located (for
example, (for example, myBucket). \n\n"+
    " video - The name of video (for example, people.mp4). \n\n" +
    " topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic. \n\n" +
    " roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use. \n\n" ;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];

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

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

    startTextLabels(rekClient, channel, bucket, video);
    GetTextResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

// snippet-start:[rekognition.java2.recognize_video_text.main]
public static void startTextLabels(RekognitionClient rekClient,
                                   NotificationChannel channel,
                                   String bucket,
                                   String video) {

    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
```

```
        .build();

        Video vid0b = Video.builder()
            .s3Object(s3obj)
            .build();

        StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vid0b)
            .build();

        StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

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

public static void GetTextResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetTextDetectionResponse textDetectionResponse=null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (textDetectionResponse !=null)
                paginationToken = textDetectionResponse.nextToken();

            GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
```

```
        while (!finished) {
            textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
            status = textDetectionResponse.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }

        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null.
        VideoMetadata videoMeta=textDetectionResponse.videoMetadata();
        System.out.println("Format: " + videoMeta.format());
        System.out.println("Codec: " + videoMeta.codec());
        System.out.println("Duration: " + videoMeta.durationMillis());
        System.out.println("FrameRate: " + videoMeta.frameRate());
        System.out.println("Job");

        List<TextDetectionResult> labels=
textDetectionResponse.textDetections();
        for (TextDetectionResult detectedText: labels) {
            System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
            System.out.println("Id : " +
detectedText.textDetection().id());
            System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
            System.out.println("Type: " +
detectedText.textDetection().type());
            System.out.println("Text: " +
detectedText.textDetection().detectedText());
            System.out.println();
        }

        } while (textDetectionResponse !=null &&
textDetectionResponse.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
```

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

Python

1. Tambahkan kode berikut ke kelas VideoDetect yang Anda buat di langkah 1.

```
# Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

def StartSegmentDetection(self):

    min_Technical_Cue_Confidence = 80.0
    min_Shot_Confidence = 80.0
    max_pixel_threshold = 0.1
    min_coverage_percentage = 60

    response = self.rek.start_segment_detection(
        Video={"S3Object": {"Bucket": self.bucket, "Name": self.video}},
        NotificationChannel={
            "RoleArn": self.roleArn,
            "SNSTopicArn": self.snsTopicArn,
        },
        SegmentTypes=["TECHNICAL_CUE", "SHOT"],
        Filters={
            "TechnicalCueFilter": {
                "BlackFrame": {
                    "MaxPixelThreshold": max_pixel_threshold,
                    "MinCoveragePercentage": min_coverage_percentage,
                },
                "MinSegmentConfidence": min_Technical_Cue_Confidence,
            },
            "ShotFilter": {"MinSegmentConfidence": min_Shot_Confidence},
        }
    )

    self.startJobId = response["JobId"]
    print(f"Start Job Id: {self.startJobId}")
```



```
def GetSegmentDetectionResults(self):
    maxResults = 10
    paginationToken = ""
    finished = False
    firstTime = True

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

        if firstTime == True:
            print(f"Status\n-----\n{response['JobStatus']}")
            print("\nRequested Types\n-----")
            for selectedSegmentType in response['SelectedSegmentTypes']:
                print(f"\tType: {selectedSegmentType['Type']}")
                print(f"\t\tModel Version:
{selectedSegmentType['ModelVersion']}")

            print()
            print("\nAudio metadata\n-----")
            for audioMetadata in response['AudioMetadata']:
                print(f"\tCodec: {audioMetadata['Codec']}")
                print(f"\tDuration: {audioMetadata['DurationMillis']}")
                print(f"\tNumber of Channels:
{audioMetadata['NumberOfChannels']}")
                print(f"\tSample rate: {audioMetadata['SampleRate']}")
            print()
            print("\nVideo metadata\n-----")
            for videoMetadata in response["VideoMetadata"]:
                print(f"\tCodec: {videoMetadata['Codec']}")
                print(f"\tColor Range: {videoMetadata['ColorRange']}")
                print(f"\tDuration: {videoMetadata['DurationMillis']}")
                print(f"\tFormat: {videoMetadata['Format']}")
                print(f"\tFrame rate: {videoMetadata['FrameRate']}")
                print("\nSegments\n-----")

            firstTime = False

        for segment in response['Segments']:

            if segment["Type"] == "TECHNICAL_CUE":
```

```

        print("Technical Cue")
        print(f"\tConfidence: {segment['TechnicalCueSegment']
['Confidence']}")
        print(f"\tType: {segment['TechnicalCueSegment']
['Type']}")

        if segment["Type"] == "SHOT":
            print("Shot")
            print(f"\tConfidence: {segment['ShotSegment']
['Confidence']}")
            print(f"\tIndex: " + str(segment["ShotSegment"]
["Index"]))

            print(f"\tDuration (milliseconds):
{segment['DurationMillis']}")
            print(f"\tStart Timestamp (milliseconds):
{segment['StartTimestampMillis']}")
            print(f"\tEnd Timestamp (milliseconds):
{segment['EndTimestampMillis']}")

            print(f"\tStart timecode: {segment['StartTimecodeSMPTE']}")
            print(f"\tEnd timecode: {segment['EndTimecodeSMPTE']}")
            print(f"\tDuration timecode: {segment['DurationSMPTE']}")

            print(f"\tStart frame number {segment['StartFrameNumber']}")
            print(f"\tEnd frame number: {segment['EndFrameNumber']}")
            print(f"\tDuration frames: {segment['DurationFrames']}")

            print()

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

```

2. Dalam fungsi main, ganti baris:


```

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

```

dengan:

```
analyzer.StartSegmentDetection()  
if analyzer.GetSQSMessageSuccess()==True:  
    analyzer.GetSegmentDetectionResults()
```

 Note

Jika Anda sudah menjalankan contoh video selain [Menganalisis video yang disimpan di bucket Amazon S3 dengan Java atau Python \(SDK\)](#), kode yang akan diganti mungkin berbeda.

3. Jalankan kode tersebut. Informasi tentang segmen yang terdeteksi dalam video input akan ditampilkan.

Mendeteksi keaktifan wajah

Amazon Rekognition Face Liveness membantu Anda memverifikasi bahwa pengguna yang melalui verifikasi wajah hadir secara fisik di depan kamera. Ini mendeteksi serangan spoof yang disajikan ke kamera atau mencoba mem-bypass kamera. Pengguna dapat menyelesaikan pemeriksaan Face Liveness dengan mengambil selfie video pendek di mana mereka mengikuti serangkaian petunjuk yang dimaksudkan untuk memverifikasi keberadaan mereka.

Face Liveness ditentukan dengan perhitungan probabilistik, dan kemudian skor kepercayaan (antara 0-100) dikembalikan setelah cek. Semakin tinggi skor, semakin besar kepercayaan diri bahwa orang yang mengambil cek itu hidup. Face Liveness juga mengembalikan bingkai, yang disebut gambar referensi yang dapat digunakan untuk perbandingan wajah dan pencarian. Seperti halnya sistem berbasis probabilitas, Face Liveness tidak dapat menjamin hasil yang sempurna. Gunakan dengan faktor lain untuk membuat keputusan berbasis risiko tentang identitas pribadi pengguna.

Face Liveness menggunakan beberapa komponen:

- AWS Amplify SDK ([React](#), [Swift \(iOS\)](#), [dan](#) [Android](#)) dengan komponen `FaceLivenessDetector`
- AWS SDK
- AWS API Cloud

Saat Anda mengonfigurasi aplikasi untuk diintegrasikan dengan fitur Face Liveness, aplikasi ini menggunakan operasi API berikut:

- [CreateFaceLivenessSession](#)- Memulai sesi Face Liveness, membiarkan model deteksi Face Liveness digunakan dalam aplikasi Anda. Mengembalikan a `SessionId` untuk sesi yang dibuat.
- [StartFaceLivenessSession](#)- Disebut oleh `AWS Amplify FaceLivenessDetector`. Memulai aliran acara yang berisi informasi tentang peristiwa dan atribut yang relevan dalam sesi saat ini.
- [GetFaceLivenessSessionHasil](#) - Mengambil hasil sesi Face Liveness tertentu, termasuk skor kepercayaan Face Liveness, gambar referensi, dan gambar audit.

Anda akan menggunakan AWS Amplify SDK untuk mengintegrasikan fitur Face Liveness dengan alur kerja verifikasi berbasis wajah Anda untuk aplikasi web. Saat pengguna melakukan onboard atau mengautentikasi melalui aplikasi Anda, kirimkan mereka ke alur kerja pemeriksaan Face Liveness di Amplify SDK. Amplify SDK menangani antarmuka pengguna dan umpan balik real-time untuk pengguna saat mereka merekam selfie video mereka.

Saat wajah pengguna bergerak ke oval yang ditampilkan di perangkat mereka, Amplify SDK menampilkan urutan lampu berwarna di layar. Kemudian dengan aman mengalirkan video selfie ke API cloud. API cloud melakukan analisis waktu nyata dengan model HTML tingkat lanjut. Setelah analisis selesai, Anda menerima yang berikut di backend:

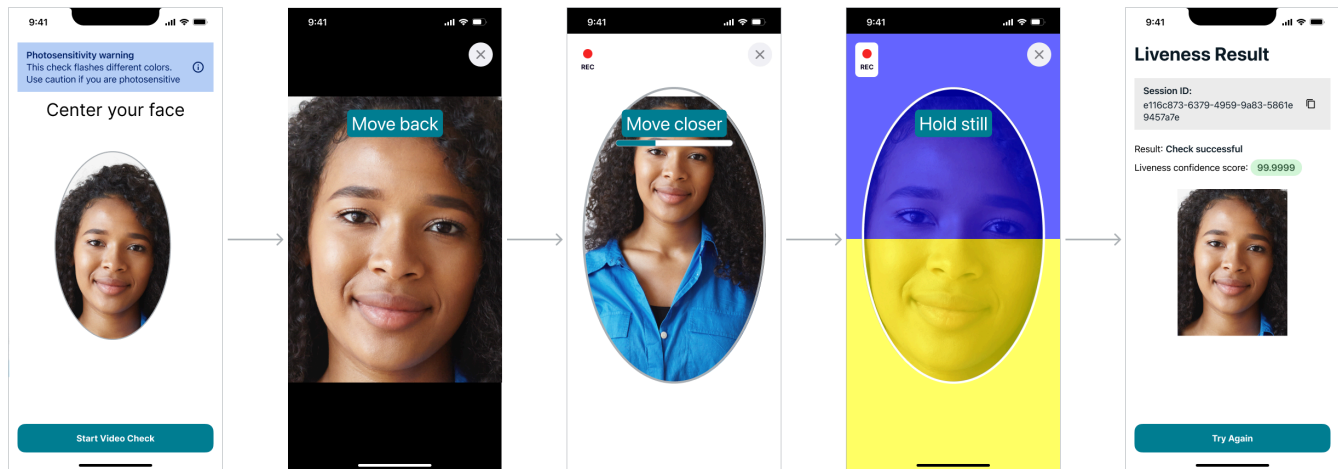
- Skor kepercayaan diri Face Liveness (antara 0 dan 100)
- Gambar berkualitas tinggi yang disebut gambar referensi yang dapat digunakan untuk pencocokan wajah atau pencarian wajah
- Satu set hingga empat gambar, yang disebut gambar audit, dipilih dari video selfie

Face Liveness dapat dimanfaatkan untuk berbagai kasus penggunaan. Misalnya, Face Liveness dapat digunakan bersama dengan pencocokan wajah (dengan [CompareFaces](#) dan [SearchFacesByImage](#)) untuk verifikasi identitas, untuk [estimasi usia](#) pada platform dengan pembatasan akses berbasis usia, dan untuk mendeteksi pengguna manusia nyata sambil menghalangi bot.

Anda dapat mempelajari lebih lanjut tentang kasus penggunaan yang dimaksudkan untuk layanan ini, bagaimana pembelajaran mesin (ML) digunakan oleh layanan, dan pertimbangan utama dalam desain dan penggunaan layanan yang bertanggung jawab dalam kartu layanan [Rekognition Face Liveness AI](#).

Anda dapat menetapkan ambang batas untuk skor kepercayaan Face Liveness dan face match. Ambang batas yang Anda pilih harus mencerminkan kasus penggunaan Anda. Anda kemudian mengirim persetujuan/penolakan verifikasi identitas kepada pengguna berdasarkan skor yang berada di atas atau di bawah ambang batas. Jika ditolak, minta pengguna untuk mencoba lagi atau mengirimkannya ke metode lain.

Grafik berikut menunjukkan alur pengguna, dari instruksi hingga pemeriksaan keaktifan hingga hasil yang dikembalikan:



Persyaratan Keaktifan Wajah Sisi Pengguna

Amazon Rekognition Face Liveness membutuhkan spesifikasi minimum berikut:

Perangkat:

- Perangkat harus memiliki kamera yang menghadap ke depan
- Kecepatan refresh minimum tampilan perangkat: 60 Hz
- Tampilan minimum atau ukuran layar: 4 inci
- Perangkat tidak boleh di-jail-break atau di-root

Spesifikasi kamera:

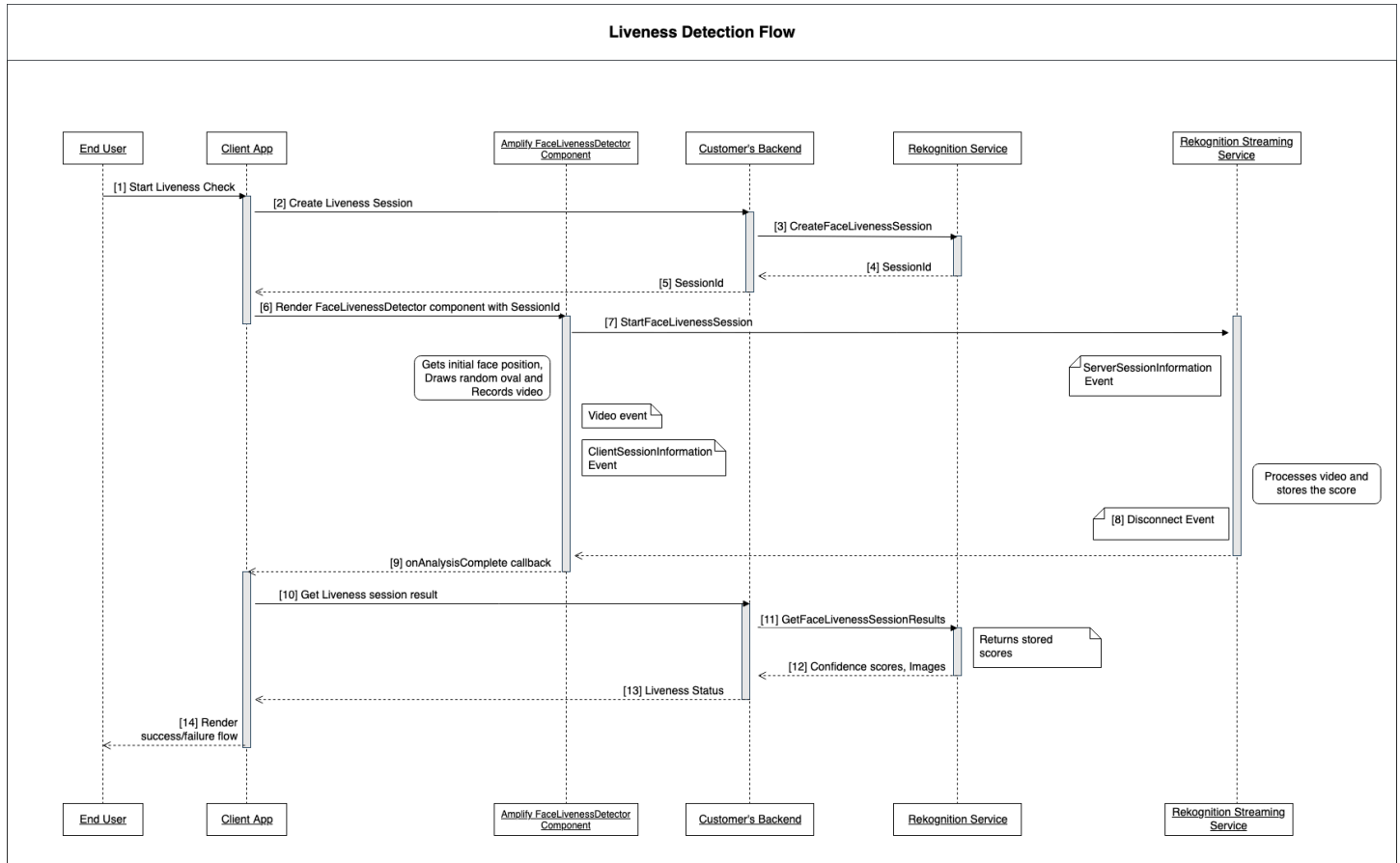
- Kamera Warna: kamera yang menghadap ke depan harus dapat merekam warna.
- Tidak ada kamera virtual atau perangkat lunak kamera.
- Kemampuan perekaman minimum: 15 frame per detik.
- Resolusi perekaman video minimum: 320x240px.
- Saat pengguna menggunakan webcam dengan desktop untuk pemeriksaan Face Liveness, penting untuk memasang webcam di atas layar yang sama di mana pemeriksaan Face Liveness dimulai.

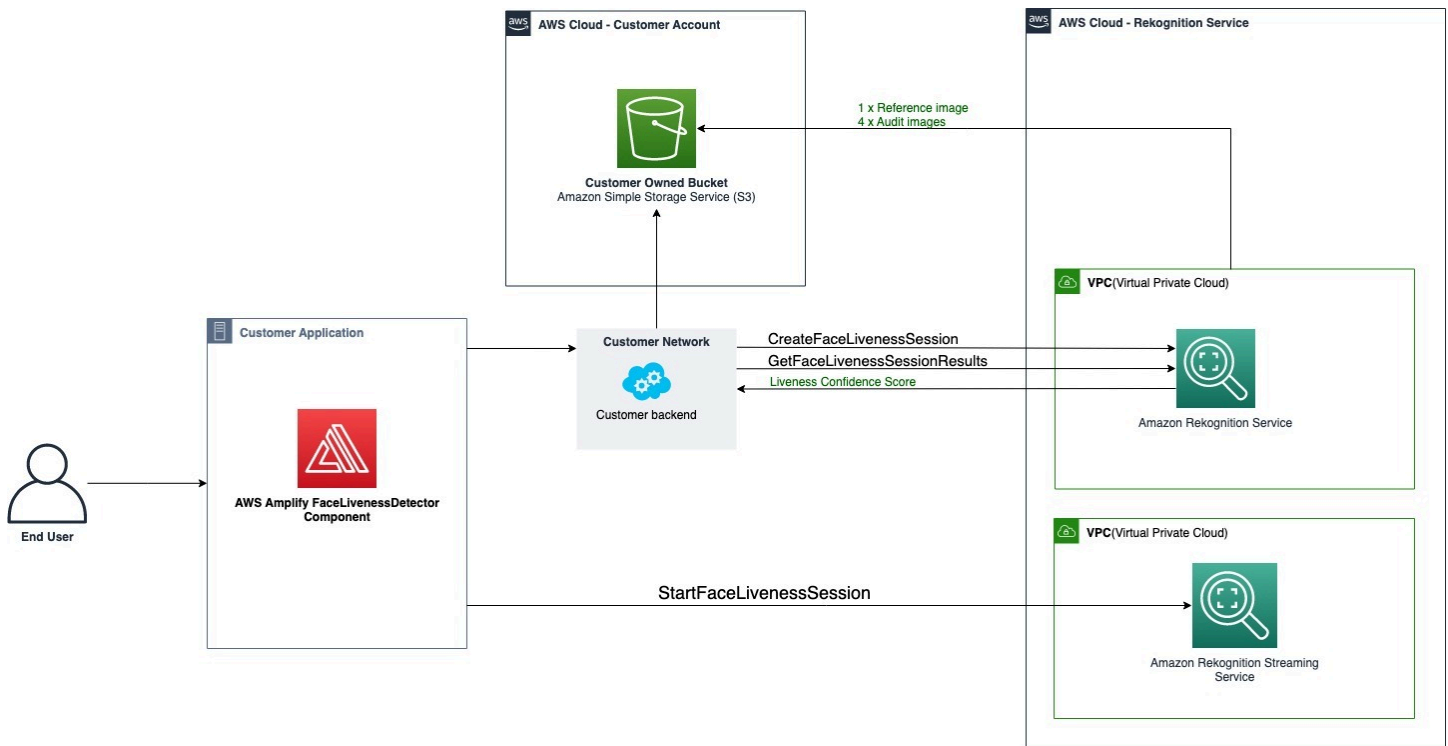
Persyaratan bandwidth minimum: 100 kbps

Browser yang didukung: Tiga versi terbaru dari browser utama, seperti Google Chrome, Mozilla Firefox, Apple Safari, dan Microsoft Edge. Untuk informasi selengkapnya mengenai dukungan browser, lihat [Browser apa yang didukung untuk digunakan dengan AWS Management Console?](#)

Arsitektur dan Diagram Urutan

Diagram berikut merinci cara Amazon Rekognition Face Liveness beroperasi terkait arsitektur fitur dan urutan operasi:





Proses pemeriksaan Face Liveness melibatkan beberapa langkah sebagaimana diuraikan sebagai berikut:

1. Pengguna memulai pemeriksaan Face Liveness di Aplikasi Klien.
2. Aplikasi Klien memanggil backend pelanggan, yang pada gilirannya memanggil layanan Amazon Rekognition. Layanan ini menciptakan Face Liveness Session dan mengembalikan yang unik SessionId. Catatan: Setelah dikirim akan kedaluwarsa dalam 3 menit, jadi hanya ada jendela 3 menit untuk menyelesaikan Langkah 3 hingga 7 di bawah ini. SessionId SessionID baru harus digunakan untuk setiap pemeriksaan Face Liveness. Jika SessionID yang diberikan digunakan untuk pemeriksaan Face Liveness berikutnya, pemeriksaan akan gagal. Selain itu, SessionId kedaluwarsa 3 menit setelah dikirim, membuat semua data Liveness yang terkait dengan sesi (misalnya, SessionID, gambar referensi, gambar audit, dll.) tidak tersedia.
3. Aplikasi Klien merender komponen FaceLivenessDetector Amplify menggunakan callback yang SessionId diperoleh dan sesuai.
4. FaceLivenessDetector Komponen ini membuat koneksi ke layanan streaming Amazon Rekognition, membuat oval di layar pengguna, dan menampilkan urutan lampu berwarna. FaceLivenessDetector merekam dan mengalirkan video secara real-time ke layanan streaming Amazon Rekognition.

5. Layanan streaming Amazon Rekognition memproses video secara real-time, menyimpan hasilnya, dan mengembalikan DisconnectEvent a ke FaceLivenessDetector komponen saat streaming selesai.
6. FaceLivenessDetector Komponen memanggil onAnalysisComplete callback untuk memberi sinyal ke Aplikasi Klien bahwa streaming telah selesai dan skor siap untuk diambil.
7. Aplikasi Klien memanggil backend pelanggan untuk mendapatkan tanda Boolean yang menunjukkan apakah pengguna aktif atau tidak. Backend pelanggan membuat permintaan ke layanan Amazon Rekognition untuk mendapatkan skor kepercayaan, referensi, dan gambar audit. Backend pelanggan menggunakan atribut ini untuk menentukan apakah pengguna aktif dan mengembalikan respons yang sesuai ke Aplikasi Klien.
8. Terakhir, Aplikasi Klien meneruskan respons ke FaceLivenessDetector komponen, yang secara tepat merender pesan sukses/kegagalan untuk menyelesaikan alur.

Prasyarat

Prasyarat untuk menggunakan Amazon Rekognition Face Liveness meliputi yang berikut:

1. Menyiapkan AWS Akun
2. Siapkan SDK Face Liveness AWS
3. Siapkan sumber daya AWS Amplify

Langkah 1: Siapkan AWS akun

Jika Anda belum memiliki AWS akun, selesaikan langkah-langkah yang terlihat di [Buat AWS Akun dan Pengguna](#) untuk membuatnya.

Langkah 2: Siapkan SDK Face Liveness AWS

Jika Anda belum melakukannya, instal dan konfigurasi SDK AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).

Ada beberapa cara untuk mengautentikasi panggilan AWS SDK. Contoh dalam panduan ini mengasumsikan bahwa Anda menggunakan profil kredensial default untuk memanggil perintah AWS CLI dan AWS operasi SDK API.

Lihat halaman [Pemberian Akses Terprogram](#) untuk informasi selengkapnya tentang pemberian akses akun pengguna ke SDK pilihan Anda. AWS Halaman ini juga menjelaskan cara menggunakan profil di komputer lokal Anda dan cara menjalankan kode sampel di AWS lingkungan.

Pastikan bahwa pengguna yang memanggil operasi Face Liveness memiliki izin yang benar untuk memanggil operasi, seperti izin `AmazonRekognitionFullAccess` dan `AmazonS3FullAccess` izin.

Langkah 3: Siapkan AWS Amplify Resources

Untuk mengintegrasikan Amazon Rekognition Face Liveness di aplikasi, Anda harus menyiapkan Amplify SDK untuk menggunakan komponen AWS Amplify. `FaceLivenessDetector`

Jika Anda belum melakukannya, ikuti petunjuk untuk menyiapkan AWS Command Line Interface (AWS CLI) Interface (AWS CLI) [saat Memulai AWS CLI](#). Setelah CLI diinstal, selesaikan langkah langkah Konfigurasi Auth yang terlihat di situs [dokumen Amplify UI untuk menyiapkan sumber daya Amplify](#) Anda. AWS

Praktik Terbaik untuk Mendeteksi Keaktifan Wajah

Kami menyarankan Anda mengikuti beberapa praktik terbaik saat menggunakan Amazon Rekognition Face Liveness. Praktik terbaik Face Liveness mencakup pedoman di mana pemeriksaan Face Liveness harus dilakukan, penggunaan gambar audit, dan memilih ambang batas skor kepercayaan.

Lihat [Rekomendasi untuk Penggunaan Face Liveness](#) daftar lengkap praktik terbaik.

Memprogram Amazon Rekognition Face Liveness API

Untuk menggunakan Amazon Rekognition Face Liveness API, Anda harus membuat backend yang melakukan langkah-langkah berikut:

1. Panggilan [CreateFaceLivenessSession](#) untuk memulai sesi Face Liveness. Ketika `CreateFaceLivenessSession` operasi selesai, UI meminta pengguna untuk mengirimkan selfie video. `FaceLivenessDetector` Komponen AWS Amplify kemudian memanggil [StartFaceLivenessSession](#) untuk melakukan deteksi Liveness.
2. Hubungi [GetFaceLivenessSessionHasil](#) untuk mengembalikan hasil deteksi yang terkait dengan sesi Face Liveness.

3. Lanjutkan untuk mengonfigurasi aplikasi React Anda untuk menggunakan FaceLivenessDetector komponen dengan mengikuti langkah-langkah di panduan [Amplify Liveness](#).

Sebelum menggunakan Face Liveness, pastikan Anda telah membuat Akun AWS, menyiapkan AWS CLI dan AWS SDK, dan menyiapkan AWS Amplify. Anda juga harus memastikan kebijakan IAM untuk API backend Anda memiliki izin yang mencakup hal-hal berikut: dan. `GetFaceLivenessSessionResults` `CreateFaceLivenessSession` Lihat bagian [Prasyarat](#) untuk informasi selengkapnya.

Langkah 1: CreateFaceLivenessSession

`CreateFaceLivenessSession` Operasi API membuat sesi Face Liveness dan mengembalikan yang `unikSessionId`.

Sebagai bagian dari input untuk operasi ini, dimungkinkan juga untuk menentukan lokasi bucket Amazon S3. Ini memungkinkan penyimpanan gambar referensi dan gambar audit yang dihasilkan selama sesi Face Liveness. Bucket Amazon S3 harus berada di akun AWS pemanggil dan di wilayah yang sama dengan titik akhir Face Liveness. Selain itu, kunci objek S3 dihasilkan oleh sistem Face Liveness.

Dimungkinkan juga untuk memberikan `AuditImagesLimit`, yang merupakan angka antara 0 dan 4. Secara default, ini diatur ke 0. Jumlah gambar yang dikembalikan adalah upaya terbaik dan berdasarkan durasi video selfie.

Contoh Permintaan

```
{
  "ClientRequestToken": "my_default_session",
  "Settings": {
    "OutputConfig": {
      "S3Bucket": "s3bucket",
      "S3KeyPrefix": "s3prefix"
    },
    "AuditImagesLimit": 1
  }
}
```

Contoh Respons

```
{
  {"SessionId": "0f959dbb-37cc-45d8-a08d-dc42cce85fa8"}
}
```

Langkah 2: StartFaceLivenessSession

Saat operasi CreateFaceLivenessSession API selesai, komponen AWS Amplify melakukan operasi API StartFaceLivenessSession . Pengguna diminta untuk merekam selfie video. Agar pemeriksaan berhasil, pengguna harus memosisikan wajah mereka di dalam oval layar sambil mempertahankan pencahayaan yang baik. Untuk informasi selengkapnya, lihat [Rekomendasi untuk Penggunaan Face Liveness](#).

Operasi API ini memerlukan video yang direkam selama sesi Face Liveness, SessionID yang diperoleh dari operasi API, CreateFaceLivenessSession dan callback. onAnalysisComplete Callback dapat digunakan untuk memberi sinyal backend untuk memanggil operasi GetFaceLivenessSessionResults API, yang mengembalikan skor kepercayaan, referensi, dan gambar audit.

Perhatikan bahwa langkah ini dilakukan oleh FaceLivenessDetector komponen AWS Amplify pada aplikasi klien. Anda tidak perlu melakukan pengaturan tambahan untuk meneleponStartFaceLivenessSession.

Langkah 3: GetFaceLivenessSessionResults

Operasi GetFaceLivenessSessionResults API mengambil hasil sesi Face Liveness tertentu. Ini membutuhkan SessionID sebagai input dan mengembalikan skor kepercayaan Face Liveness yang sesuai. Ini juga menyediakan gambar referensi yang mencakup kotak pembatas wajah, dan gambar audit yang juga berisi kotak pembatas wajah. Skor kepercayaan diri Face Liveness berkisar antara 0-100.

Contoh Permintaan

```
{"SessionId": "0f959dbb-37cc-45d8-a08d-dc42cce85fa8"}
```

Contoh Respons

```
{
  "SessionId": "0f959dbb-37cc-45d8-a08d-dc42cce85fa8",
  "Confidence": 98.9735,
  "ReferenceImage": {
    "S3Object": {
      "Bucket": "s3-bucket-name",
      "Name": "file-name",
    },
    "BoundingBox": {
      "Height": 0.4943420886993408,
      "Left": 0.8435328006744385,
      "Top": 0.8435328006744385,
      "Width": 0.9521094560623169}
  },
  "AuditImages": [{
    "S3Object": {
      "Bucket": "s3-bucket-name",
      "Name": "audit-image-name",
    },
    "BoundingBox": {
      "Width": 0.6399999856948853,
      "Height": 0.47999998927116394,
      "Left": 0.1644444465637207,
      "Top": 0.17666666209697723}
  }],
  "Status": "SUCCEEDED"
}
```

Langkah 4: Menanggapi hasil

Setelah sesi Face Liveness, bandingkan skor kepercayaan cek dengan ambang batas yang ditentukan. Jika skor lebih tinggi dari ambang batas, pengguna dapat pergi ke layar atau tugas berikutnya. Jika cek gagal, pengguna akan diberi tahu dan diminta untuk mencoba lagi.

Memanggil Face Liveness API

[Anda dapat menguji Amazon Rekognition Face Liveness dengan SDK apa pun yang AWS didukung, seperti AWS Python SDK Boto3 atau AWS SDK for Java.](#) Anda dapat memanggil `CreateFaceLivenessSession` dan `GetFaceLivenessSessionResults` API dengan SDK pilihan Anda. Bagian berikut menunjukkan cara memanggil API ini dengan Python dan Java SDK.

Untuk memanggil Face Liveness API:

- Jika Anda belum melakukannya, buat atau perbarui pengguna dengan AmazonRekognitionFullAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Mengatur akun AWS dan membuat Pengguna](#).
- Jika Anda belum melakukannya, instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Mengatur AWS CLI dan AWS SDK](#).

Python

Cuplikan berikut menunjukkan bagaimana Anda dapat memanggil API ini di aplikasi Python Anda. Perhatikan bahwa untuk menjalankan contoh ini Anda harus menggunakan setidaknya versi 1.26.110 dari Boto3 SDK, meskipun versi terbaru SDK direkomendasikan.

```
import boto3

session = boto3.Session(profile_name='default')
client = session.client('rekognition')

def create_session():

    response = client.create_face_liveness_session()

    session_id = response.get("SessionId")
    print('SessionId: ' + session_id)

    return session_id

def get_session_results(session_id):

    response = client.get_face_liveness_session_results(SessionId=session_id)

    confidence = response.get("Confidence")
    status = response.get("Status")

    print('Confidence: ' + "{:.2f}".format(confidence) + "%")
    print('Status: ' + status)

    return status
```

```
def main():
    session_id = create_session()
    print('Created a Face Liveness Session with ID: ' + session_id)

    status = get_session_results(session_id)
    print('Status of Face Liveness Session: ' + status)

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

Java

Cuplikan berikut menunjukkan bagaimana Anda dapat memanggil API ini di aplikasi Java Anda:

```
package aws.example.rekognition.liveness;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.CreateFaceLivenessSessionRequest;
import com.amazonaws.services.rekognition.model.CreateFaceLivenessSessionResult;
import
    com.amazonaws.services.rekognition.model.GetFaceLivenessSessionResultsRequest;
import
    com.amazonaws.services.rekognition.model.GetFaceLivenessSessionResultsResult;

public class DemoLivenessApplication {

    static AmazonRekognition rekognitionClient;

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

        rekognitionClient = AmazonRekognitionClientBuilder.defaultClient();

        try {
            String sessionId = createSession();
            System.out.println("Created a Face Liveness Session with ID: " +
                sessionId);
        }
    }
}
```

```
        String status = getSessionResults(sessionId);
        System.out.println("Status of Face Liveness Session: " + status);

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

private static String createSession() throws Exception {

    CreateFaceLivenessSessionRequest request = new
CreateFaceLivenessSessionRequest();
    CreateFaceLivenessSessionResult result =
rekognitionClient.createFaceLivenessSession(request);

    String sessionId = result.getSessionId();
    System.out.println("SessionId: " + sessionId);

    return sessionId;
}

private static String getSessionResults(String sessionId) throws Exception {

    GetFaceLivenessSessionResultsRequest request = new
GetFaceLivenessSessionResultsRequest().withSessionId(sessionId);
    GetFaceLivenessSessionResultsResult result =
rekognitionClient.getFaceLivenessSessionResults(request);

    Float confidence = result.getConfidence();
    String status = result.getStatus();

    System.out.println("Confidence: " + confidence);
    System.out.println("status: " + status);

    return status;
}
}
```

Java V2

Cuplikan berikut menunjukkan cara memanggil Face Liveness API dengan Java V2 SDK AWS :


```
package aws.example.rekognition.liveness;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.CreateFaceLivenessSessionRequest;
import com.amazonaws.services.rekognition.model.CreateFaceLivenessSessionResult;
import
    com.amazonaws.services.rekognition.model.GetFaceLivenessSessionResultsRequest;
import
    com.amazonaws.services.rekognition.model.GetFaceLivenessSessionResultsResult;

public class DemoLivenessApplication {

    static AmazonRekognition rekognitionClient;

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

        rekognitionClient = AmazonRekognitionClientBuilder.defaultClient();

        try {
            String sessionId = createSession();
            System.out.println("Created a Face Liveness Session with ID: " +
sessionId);

            String status = getSessionResults(sessionId);
            System.out.println("Status of Face Liveness Session: " + status);

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

    private static String createSession() throws Exception {

        CreateFaceLivenessSessionRequest request = new
CreateFaceLivenessSessionRequest();
        CreateFaceLivenessSessionResult result =
rekognitionClient.createFaceLivenessSession(request);

        String sessionId = result.getSessionId();
        System.out.println("SessionId: " + sessionId);
    }
}
```

```
        return sessionId;
    }

    private static String getSessionResults(String sessionId) throws Exception {

        GetFaceLivenessSessionResultsRequest request = new
        GetFaceLivenessSessionResultsRequest().withSessionId(sessionId);
        GetFaceLivenessSessionResultsResult result =
        rekognitionClient.getFaceLivenessSessionResults(request);

        Float confidence = result.getConfidence();
        String status = result.getStatus();

        System.out.println("Confidence: " + confidence);
        System.out.println("status: " + status);

        return status;
    }
}
```

Node.Js

Cuplikan berikut menunjukkan cara memanggil Face Liveness API dengan Node.Js SDK: AWS

```
const Rekognition = require("aws-sdk/clients/rekognition");

const rekognitionClient = new Rekognition({ region: "us-east-1" });

async function createSession() {
    const response = await rekognitionClient.createFaceLivenessSession().promise();

    const sessionId = response.SessionId;
    console.log("SessionId:", sessionId);

    return sessionId;
}

async function getSessionResults(sessionId) {
    const response = await rekognitionClient
        .getFaceLivenessSessionResults({
```

```
        SessionId: sessionId,
    })
    .promise();

    const confidence = response.Confidence;
    const status = response.Status;
    console.log("Confidence:", confidence);
    console.log("Status:", status);

    return status;
}

async function main() {
    const sessionId = await createSession();
    console.log("Created a Face Liveness Session with ID:", sessionId);

    const status = await getSessionResults(sessionId);
    console.log("Status of Face Liveness Session:", status);
}

main();
```

Node.Js (Javascript SDK v3)

Cuplikan berikut menunjukkan cara memanggil Face Liveness API dengan AWS Node.Js SDK untuk Javascript v3:

```
import { RekognitionClient, CreateFaceLivenessSessionCommand } from "@aws-sdk/
client-rekognition"; // ES Modules
import const { RekognitionClient, CreateFaceLivenessSessionCommand } =
    require("@aws-sdk/client-rekognition"); // CommonJS import
const client = new RekognitionClient(config);
const input = {
    KmsKeyId: "STRING_VALUE",
    Settings: {
        OutputConfig: { // LivenessOutputConfig
            S3Bucket: "STRING_VALUE", // required
            S3KeyPrefix: "STRING_VALUE",
        },
        AuditImagesLimit: Number("int"),
    },
    ClientRequestToken: "STRING_VALUE",
```

```
};  
const command = new CreateFaceLivenessSessionCommand(input);  
const response = await client.send(command);  
// { // CreateFaceLivenessSessionResponse  
//   SessionId: "STRING_VALUE", // required  
// };
```

Mengkonfigurasi dan Menyesuaikan Aplikasi Anda

Mengkonfigurasi Aplikasi Anda

Aplikasi Face Liveness Anda dapat beroperasi di perangkat seluler atau browser web desktop. Anda akan ingin mengonfigurasi komponen Face Liveness untuk diintegrasikan dengan solusi pilihan Anda. Anda juga harus memastikan aplikasi Anda memiliki izin untuk menggunakan kamera perangkat. [Panduan Amplify Liveness](#) memberikan instruksi terperinci mengenai cara:

- Instal dan konfigurasikan AWS Amplify
- Impor dan render FaceLivenessDetector komponen
- Dengarkan callback
- Render Amplify contoh pesan kesalahan

Sesuaikan Aplikasi Anda

Anda dapat menyesuaikan komponen tertentu dari aplikasi liveness Anda menggunakan [AWS Amplify](#).

Untuk informasi tentang terjemahan, lihat dokumentasi [Amplify Authenticator](#).

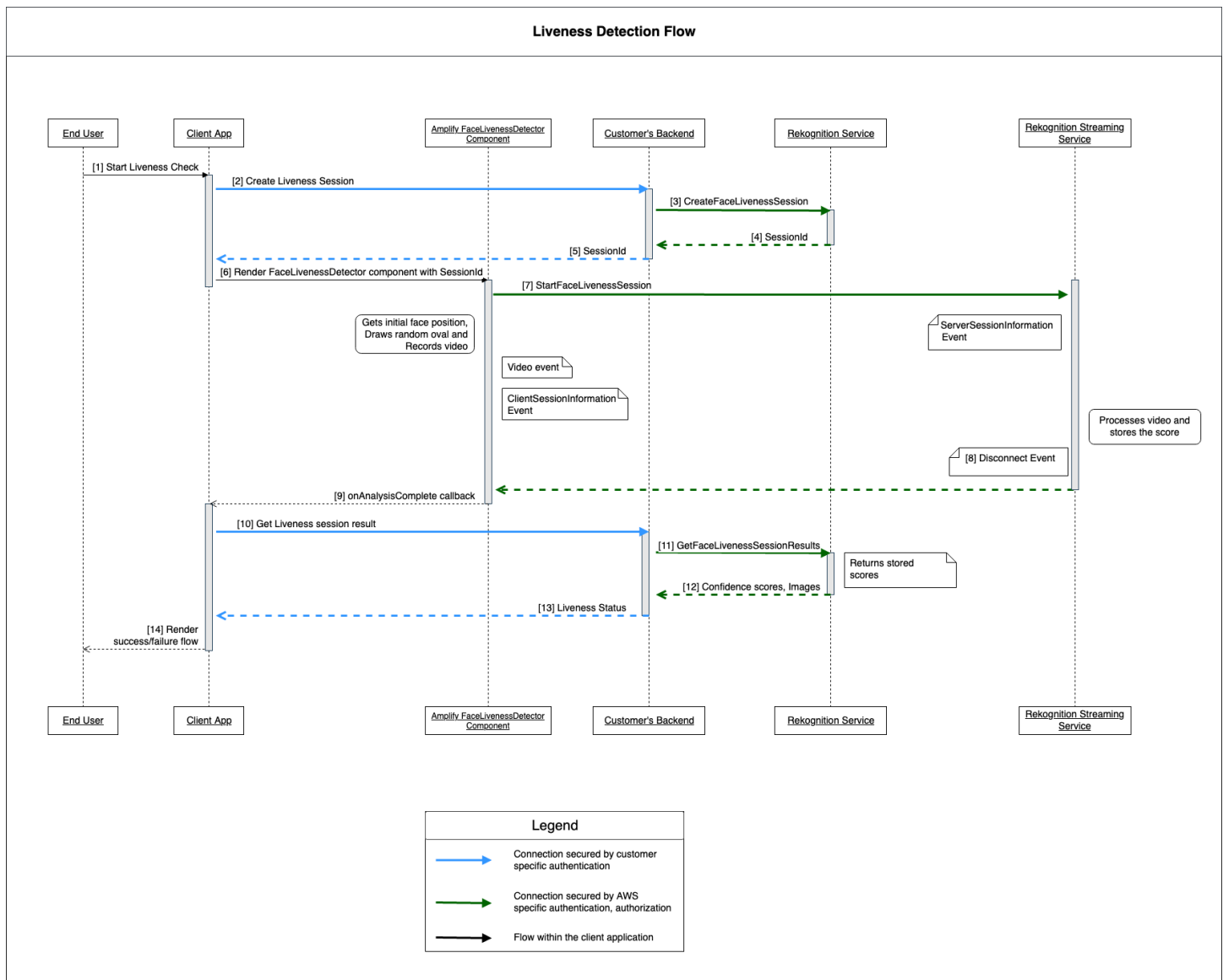
[Untuk informasi tentang menyesuaikan komponen dan tema Amplify, lihat dokumentasi Amplify mengenai tema.](#)

Model Tanggung Jawab Bersama Face Liveness

Keamanan dan Kepatuhan adalah tanggung jawab bersama antara AWS dan Anda, pelanggan. Baca lebih lanjut tentang model tanggung jawab AWS bersama [di sini](#).

1. Semua panggilan ke AWS layanan (melalui aplikasi klien atau backend pelanggan) diautentikasi dan diotorisasi dengan AWS Auth (Otentikasi). AWS Ini adalah tanggung jawab pemilik layanan Face Liveness untuk memastikan hal ini terjadi.
2. Semua panggilan ke backend pelanggan (dari aplikasi klien) diautentikasi dan diotorisasi melalui pelanggan. Tanggung jawab ini jatuh pada pelanggan. Pelanggan harus memastikan bahwa panggilan dari aplikasi klien diautentikasi dan belum dimanipulasi dengan cara apa pun.
3. Backend pelanggan harus mengidentifikasi pengguna akhir yang melakukan tantangan Face Liveness. Adalah tanggung jawab pelanggan untuk mengikat pengguna akhir ke sesi Face Liveness. Layanan Face Liveness tidak membedakan antara pengguna akhir. Itu hanya dapat mengidentifikasi AWS identitas panggilan (yang ditangani pelanggan).

Diagram alir berikut menunjukkan panggilan mana yang diautentikasi oleh layanan AWS atau oleh pelanggan:



Semua panggilan ke layanan Amazon Rekognition Face Liveness dilindungi AWS oleh Auth (menggunakan mekanisme penandatanganan). AWS Ini termasuk panggilan berikut:

- [3] Panggilan [CreateFaceLivenessSession](#) API (dari backend pelanggan)
- [7] Panggilan [StartFaceLivenessSession](#) API (dari aplikasi klien)
- [11] Panggilan [GetFaceLivenessSessionResults](#) API (dari backend pelanggan)

Semua panggilan ke backend pelanggan harus memiliki mekanisme otentikasi dan otorisasi. Pelanggan perlu memastikan bahwa kode/perpustakaan/dll pihak ketiga yang digunakan sedang dipelihara dan dikembangkan secara aktif. Pelanggan juga perlu memastikan bahwa pengguna

akhir yang benar melakukan panggilan ke sesi Face Liveness yang benar. Pelanggan harus mengautentikasi dan mengotorisasi alur berikut:

- [2] Buat sesi Face Liveness (dari aplikasi klien)
- [10] Dapatkan hasil sesi Face Liveness (dari aplikasi klien)

Pelanggan dapat mengikuti model keamanan [STRIDE](#) untuk memastikan bahwa panggilan API mereka dilindungi.

Tipe	Deskripsi	Kontrol Keamanan
Pemalsuan	Tindakan ancaman yang ditujukan untuk mengakses dan menggunakan kredensya l pengguna lain, seperti nama pengguna dan kata sandi.	Autentikasi
Merusak	Tindakan ancaman yang bermaksud mengubah atau memodifikasi data persisten secara jahat. Contohnya termasuk catatan dalam database, dan perubahan data dalam perjalanan antara dua komputer melalui jaringan terbuka, seperti internet.	Integritas
Penolakan	Tindakan ancaman yang ditujukan untuk melakukan operasi terlarang dalam sistem yang tidak memiliki kemampuan untuk melacak operasi.	Non-Penolakan
Pengungkapan informasi	Tindakan ancaman yang bermaksud untuk membaca file yang tidak diberikan akses,	Kerahasiaan

atau untuk membaca data dalam perjalanan.

Penolakan layanan	Tindakan ancaman mencoba untuk menolak akses ke pengguna yang valid, seperti dengan membuat server web sementara tidak tersedia atau tidak dapat digunakan.	Ketersediaan
Ketinggian hak istimewa	Tindakan ancaman yang bermaksud untuk mendapatkan akses istimewa ke sumber daya untuk mendapatkan akses tidak sah ke informasi atau untuk membahayakan sistem.	Otorisasi

AWS mengamankan koneksinya dengan cara berikut:

1. Menghitung tanda tangan permintaan dan kemudian memverifikasi tanda tangan di sisi layanan. Permintaan diautentikasi menggunakan tanda tangan ini.
2. AWS pelanggan diminta untuk mengatur peran IAM yang tepat untuk mengotorisasi tindakan/ operasi tertentu. Peran IAM ini diperlukan untuk melakukan panggilan ke layanan AWS.
3. Hanya permintaan HTTPS ke AWS layanan yang diizinkan. Permintaan dienkripsi di jaringan terbuka menggunakan TLS. Ini melindungi kerahasiaan permintaan dan menjaga integritas permintaan.
4. AWS layanan mencatat data yang cukup untuk mengidentifikasi panggilan yang dilakukan oleh pelanggan. Ini mencegah serangan penolakan.
5. AWS layanan memiliki menjaga ketersediaan yang memadai

Pelanggan bertanggung jawab untuk mengamankan layanan dan panggilan API mereka dengan cara berikut:

1. Pelanggan harus memastikan bahwa mereka mengikuti mekanisme otentikasi yang tepat. Ada berbagai mekanisme otentikasi yang dapat digunakan untuk mengotentikasi permintaan.

- Pelanggan dapat menjelajahi [otentikasi berbasis intisari](#), [OAuth](#), [koneksi OpenID](#), dan mekanisme lainnya.
2. Pelanggan harus memastikan bahwa layanan mereka mendukung saluran enkripsi yang tepat (seperti TLS/HTTPS) untuk melakukan panggilan API layanan.
 3. Pelanggan harus memastikan bahwa mereka mencatat data yang diperlukan untuk mengidentifikasi panggilan API dan penelepon secara unik. Mereka harus dapat mengidentifikasi klien yang memanggil API mereka dengan parameter yang ditentukan dan waktu panggilan.
 4. Pelanggan harus memastikan bahwa sistem mereka tersedia, dan bahwa mereka dilindungi dari [serangan DDoS](#). Berikut adalah beberapa contoh [teknik pertahanan](#) terhadap serangan DDoS.

Pelanggan bertanggung jawab untuk menjaga aplikasi mereka up-to-date. Untuk informasi selengkapnya, lihat [Panduan pembaruan Face Liveness](#).

Panduan pembaruan Face Liveness

AWS secara teratur memperbarui Face Liveness AWS SDK (digunakan di backend pelanggan) dan komponen FaceLivenessDetector AWS Amplify SDK (digunakan dalam aplikasi klien) untuk menyediakan fitur baru, API yang diperbarui, keamanan yang ditingkatkan, perbaikan bug, peningkatan kegunaan, dan banyak lagi. Kami menyarankan Anda menyimpan SDK up-to-date untuk memastikan fungsi fitur yang optimal. Jika Anda terus menggunakan versi SDK yang lebih lama, permintaan mungkin diblokir karena alasan pemeliharaan dan keamanan.

Face Liveness mengharuskan Anda menggunakan FaceLivenessDetector komponen, yang disertakan dalam AWS Amplify SDK (React, iOS, Android).

Pembuatan versi dan kerangka waktu

Kami membuat versi komponen kunci berikut dari fitur Face Liveness. Kami mengikuti format versi semantik. Misalnya, format versi X.Y.Z di mana X mewakili versi utama, Y mewakili versi minor, dan Z mewakili versi patch.

- Tantangan pengguna Face Liveness (Misalnya, FaceMovement AndLight tantangan tantangan) adalah bagian dari API StartFaceLivenessSession
- FaceLivenessDetector komponen yang dikirimkan melalui AWS Amplify SDK digunakan dalam aplikasi klien

Versi utama: Kami mencadangkan pembaruan versi utama untuk keamanan kritis, melanggar API, dan pembaruan kegunaan show-stopper. Aplikasi dan backend pelanggan harus diperbarui sesegera mungkin agar Anda dapat terus menggunakan fitur Face Liveness. Setelah kami merilis versi mayor baru, kami mendukung versi utama sebelumnya selama 120 hari sejak hari rilis baru. Kami dapat memblokir permintaan yang berasal dari versi utama sebelumnya setelah 120 hari.

Versi minor: Kami mencadangkan pembaruan versi minor untuk fitur dan peningkatan keamanan dan kegunaan yang penting. Kami sangat menyarankan untuk menerapkan pembaruan ini. Meskipun kami berusaha untuk memastikan pembaruan minor kompatibel ke belakang selama mungkin, kami dapat mengumumkan end-of-support untuk versi minor sebelumnya 180 hari setelah rilis versi minor baru.

Versi tambalan: Kami mencadangkan pembaruan versi tambalan untuk perbaikan dan peningkatan bug opsional. Meskipun kami menyarankan agar Anda menyimpan versi Anda up-to-date untuk keamanan dan pengalaman pengguna terbaik, kami berusaha untuk memastikan pembaruan tambalan sepenuhnya kompatibel ke belakang hingga kami merilis versi mayor atau minor baru.

Jendela waktu pembuatan versi (120 hari untuk mayor dan 180 hari untuk minor) berlaku untuk memperbarui SDK di aplikasi Anda, mengunggah aplikasi Anda ke app store atau situs web, dan pengguna mengunduh versi terbaru aplikasi.

Rilis versi dan matriks kompatibilitas

Rilis versi utama untuk FaceLivenessDetector komponen atau tantangan pengguna sering bertepatan. Untuk membantu Anda melacak dependensi versi, lihat sumber daya yang ditautkan dalam tabel berikut.

Versi SDK dan changelog:

FaceLivenessDetector untuk web SDK

FaceLivenessDetector untuk iOS SDK

FaceLivenessDetector untuk Android SDK

[Versi saat ini](#)

[Changelog](#)

[Versi saat ini/Changelog](#)

[Versi saat ini/Changelog](#)

Tantangan pengguna:

Nama Tantangan

Versi

Tanggal rilis

Tanggal pensiun

FaceMovementAndLightTantangan	v1.0.0	4/10/2023	N/A
-------------------------------	--------	-----------	-----

Komunikasi rilis baru

AWS mengkomunikasikan rilis baru melalui saluran berikut:

- Pemberitahuan email pembaruan kesehatan layanan dikirim ke email akun yang terkait dengan ID akun Face Liveness.
- Pembaruan yang dipublikasikan untuk AWS SDK dan pemberitahuan terkait di repo masing-masing GitHub .
- Pembaruan yang dipublikasikan untuk AWS Amplify SDK dan notifikasi terkait di masing-masing repo. GitHub

Kami menyarankan Anda berlangganan saluran ini untuk tinggal up-to-date.

FAQ Face Liveness

Gunakan item FAQ berikut untuk menemukan jawaban atas pertanyaan umum tentang Rekognition Face Liveness.

- Apa output dari pemeriksaan keaktifan wajah?

Rekognition Face Liveness memberikan output berikut untuk setiap pemeriksaan keaktifan:

- Skor Keyakinan: Skor numerik mulai dari 0 hingga 100 dikembalikan. Skor ini menunjukkan kemungkinan bahwa video selfie berasal dari orang sungguhan dan bukan aktor buruk menggunakan spoof.
- Gambar Berkualitas Tinggi: Satu gambar berkualitas tinggi diekstraksi dari video selfie. Bingkai ini dapat digunakan untuk berbagai tujuan seperti perbandingan wajah, estimasi usia, atau pencarian wajah.
- Gambar audit: Hingga empat gambar dikembalikan dari video selfie, yang dapat digunakan untuk tujuan jejak audit.
- Apakah Rekognition Face Liveness sesuai dengan tes iBeta Presentation Attack Detection (PAD)?

Pengujian Presentation Attack Detection (PAD) iBeta Quality Assurance dilakukan sesuai dengan ISO/IEC 30107-3. iBeta diakreditasi oleh NIST/NVLAP untuk menguji dan memberikan hasil sesuai standar PAD ini. Rekognition Face Liveness lulus pengujian kesesuaian Level 1 dan Level 2 iBeta Presentation Attack Detection (PAD) dengan skor PAD yang sempurna. [Laporan dapat ditemukan di halaman web iBeta di sini.](#)

- Bagaimana saya bisa mendapatkan bingkai berkualitas tinggi dan bingkai tambahan?

Bingkai berkualitas tinggi dan bingkai tambahan dapat dikembalikan sebagai byte mentah atau diunggah ke bucket Amazon S3 yang Anda tentukan, tergantung pada konfigurasi permintaan API Anda. [CreateFaceLivenessSession](#)

- Dapatkah saya mengubah lokasi lampu oval dan berwarna?

Tidak. Lokasi oval dan lampu berwarna adalah fitur keamanan dan karenanya tidak dapat disesuaikan.

- Dapatkah saya menyesuaikan antarmuka pengguna sesuai aplikasi kami?

Ya, Anda dapat menyesuaikan sebagian besar komponen layar seperti tema, warna, bahasa, konten teks, dan font agar selaras dengan aplikasi Anda. Detail tentang cara menyesuaikan komponen ini dapat ditemukan di dokumentasi untuk komponen [React](#), [Swift](#), dan UI [Android](#) kami.

- Dapatkah saya menyesuaikan waktu dan waktu hitung mundur agar sesuai dengan wajah dalam bentuk oval?

Tidak, waktu hitung mundur dan waktu fit wajah telah ditentukan sebelumnya berdasarkan studi internal skala besar di 1000-an pengguna, dengan tujuan memberikan keseimbangan optimal antara keamanan dan latensi. Untuk alasan ini, pengaturan waktu ini tidak dapat disesuaikan.

- Mengapa lokasi oval wajah tidak selalu terpusat?

Lokasi oval dirancang untuk berubah dengan setiap pemeriksaan sebagai tindakan pengamanan. Penentuan posisi dinamis ini meningkatkan keamanan Face Liveness.

- Mengapa oval tumpah di area tampilan dalam beberapa kasus?

Lokasi oval diubah dengan setiap pemeriksaan untuk meningkatkan keamanan. Kadang-kadang, oval dapat tumpah di atas area tampilan. Namun, komponen Face Liveness memastikan setiap tumpahan terbatas dan kemampuan pengguna untuk menyelesaikan pemeriksaan dipertahankan.

- Apakah lampu warna yang berbeda memenuhi pedoman aksesibilitas?

Ya, lampu warna yang berbeda dalam produk kami mematuhi pedoman aksesibilitas yang diuraikan dalam WCAG 2.1. Seperti yang diverifikasi dengan lebih dari 1000 pemeriksaan pengguna, pengalaman pengguna menampilkan sekitar dua warna per detik, yang sesuai dengan rekomendasi membatasi warna hingga tiga per detik. Ini mengurangi kemungkinan memicu kejang epilepsi di sebagian besar populasi.

- Apakah SDK menyesuaikan kecerahan layar untuk hasil yang optimal?

SDK seluler Face Liveness (untuk Android dan iOS) secara otomatis menyesuaikan kecerahan saat pemeriksaan dimulai. Namun, untuk SDK web ada batasan pada halaman web yang mencegah penyesuaian kecerahan otomatis. Dalam kasus seperti itu, kami berharap aplikasi web menginstruksikan pengguna akhir untuk meningkatkan kecerahan layar secara manual untuk hasil yang optimal.

- Apakah itu harus oval? Bisakah kita menggunakan bentuk serupa lainnya?

Tidak, ukuran, bentuk, dan lokasi oval tidak dapat disesuaikan. Desain oval spesifik telah dipilih dengan cermat karena efektivitasnya dalam menangkap dan menganalisis gerakan wajah secara akurat. Oleh karena itu, bentuk oval tidak dapat dimodifikasi.

- Apa itu end-to-end latensi?

Kami mengukur end-to-end latensi dari saat pengguna memulai tindakan yang diperlukan untuk menyelesaikan pemeriksaan keaktifan hingga saat pengguna mendapatkan hasilnya (lulus atau gagal). Dalam kasus terbaik, latensi adalah 5s. Dalam kasus rata-rata, kami berharap sekitar 7 detik. Dalam kasus terburuk, latensi adalah 11 detik. Kami melihat variasi end-to-end latensi karena tergantung pada: waktu pengguna untuk menyelesaikan tindakan yang diperlukan (yaitu, memindahkan wajah mereka ke oval), konektivitas jaringan, latensi aplikasi, dll.

- Bisakah saya menggunakan fitur Face Liveness tanpa Amplify SDK?

Tidak, Amplify SDK diperlukan untuk menggunakan fitur Rekognition Face Liveness.

- Di mana saya dapat menemukan status kesalahan yang terkait dengan Face Liveness?

Anda dapat melihat status kesalahan Face Liveness yang berbeda [di sini](#).

- Face Liveness tidak tersedia di wilayah my. Bagaimana saya bisa menggunakan fitur ini?

Anda dapat memilih untuk memanggil Face Liveness di salah satu wilayah yang tersedia, tergantung pada beban lalu lintas dan kedekatan Anda. Face liveness saat ini tersedia di AWS wilayah berikut:

- AS Timur (Virginia Utara)
- US West (Oregon)
- Eropa (Irlandia)
- Asia Pasifik (Tokyo, Mumbai)

Bahkan jika AWS akun Anda berada di wilayah yang berbeda, perbedaan latensi tidak diharapkan signifikan. Anda dapat memperoleh bingkai selfie berkualitas tinggi dan gambar audit melalui lokasi Amazon S3 atau sebagai byte mentah, tetapi bucket Amazon S3 Anda harus cocok dengan wilayah Face Liveness. AWS Jika berbeda, Anda harus menerima gambar sebagai byte mentah.

- Apakah Amazon Rekognition Liveness Detection menggunakan konten pelanggan untuk meningkatkan layanan?

Anda dapat memilih untuk tidak menggunakan input gambar dan video Anda untuk meningkatkan atau mengembangkan kualitas Rekognition dan teknologi pembelajaran mesin/kecerdasan buatan Amazon lainnya dengan menggunakan kebijakan opt-out Organizations. AWS Untuk informasi tentang cara memilih keluar, lihat Kebijakan [opt-out Mengelola Layanan AI](#).

Analisis massal

Amazon Rekognition Bulk Analysis memungkinkan Anda memproses banyak koleksi gambar secara asinkron dengan menggunakan file manifes dengan operasi. [StartMediaAnalysisJob](#) Output untuk setiap gambar individu cocok dengan output yang dikembalikan oleh operasi yang Anda gunakan untuk analisis.

Saat ini, Rekognition mendukung analisis dengan operasi. [DetectModerationLabels](#)

Anda akan dikenakan biaya untuk jumlah gambar yang telah berhasil diproses oleh pekerjaan. Hasil pekerjaan yang sudah selesai dikeluarkan ke bucket Amazon S3 yang ditentukan.

Perhatikan bahwa Analisis Massal tidak mendukung integrasi Amazon A2I.

API dapat mendeteksi jenis konten animasi atau bergambar, dan informasi tentang jenis konten yang terdeteksi dikembalikan sebagai bagian dari respons.

Memproses gambar dalam jumlah besar

Anda dapat memulai pekerjaan analisis massal baru dengan mengirimkan file manifes dan memanggil operasi. StartMediaAnalysisJob File manifes masukan berisi referensi ke gambar di bucket Amazon S3 dan diformat sebagai berikut:

```
{"source-ref": "s3://foo/bar/1.jpg"}
```

Untuk membuat pekerjaan analisis massal (CLI)

1. Jika belum:
 - a. Buat atau perbarui pengguna dengan AmazonRekognitionFullAccess dan AmazonS3ReadOnlyAccess izin. Untuk informasi selengkapnya, lihat [Langkah 1: Siapkan akun AWS dan buat Pengguna](#).
 - b. Instal dan konfigurasi AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).
2. Unggah gambar ke bucket S3 Anda.

Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

- Gunakan perintah berikut untuk membuat dan mengambil pekerjaan analisis massal.

CLI

Gunakan perintah berikut untuk memanggil [StartMediaAnalysisJob](#) operasi untuk analisis dengan DetectModerationLabels operasi:

```
# Requests
# Starting DetectModerationLabels job with default settings
aws rekognition start-media-analysis-job \
--operations-config "DetectModerationLabels={MinConfidence='1'}" \
--input "S3Object={Bucket=my-bucket,Name=my-input.jsonl}" \
--output-config "S3Bucket=my-output-bucket,S3KeyPrefix=my-results"
```

Anda bisa mendapatkan informasi tentang pekerjaan tertentu, seperti jalur Amazon S3 dari bucket tempat file hasil dan ringkasan disimpan, dengan menggunakan operasi [GetMediaAnalysisJob](#). Anda memberikannya ID pekerjaan yang dikembalikan oleh StartMediaAnalysisJob atau ListMediaAnalysisJob. Rincian tentang pekerjaan individu hanya dipertahankan selama satu tahun.

```
# Request
aws rekognition get-media-analysis-job \
--job-id customer-job-id
```

Anda dapat membuat daftar semua analisis massal Anda dengan menggunakan operasi [ListMediaAnalysisJobs](#) pekerjaan, yang mengembalikan halaman pekerjaan. Dengan `max-results` argumen, Anda dapat menentukan jumlah maksimum pekerjaan untuk kembali per halaman, terbatas pada nilai `max-results`. Maksimal 100 hasil dikembalikan per halaman. Rincian tentang pekerjaan individu hanya dipertahankan selama satu tahun.

```
# Request
# Specify number of jobs to return per page, limited to max-results.
aws rekognition list-media-analysis-jobs --max-results 1
```


StartMediaAnalysisJob manifestasi keluaran

Pekerjaan analisis massal menghasilkan file manifes keluaran yang berisi hasil pekerjaan, serta ringkasan manifes yang berisi statistik dan detail tentang kesalahan apa pun saat memproses entri manifes masukan.

Jika entri duplikat disertakan dalam manifes masukan, pekerjaan tidak akan mencoba memfilter input unik, dan sebaliknya akan memproses semua entri yang disediakan.

File manifes keluaran diformat sebagai berikut:

```
// Output manifest for content moderation
{"source-ref":"s3://foo/bar/1.jpg", "detect-moderation-labels":
  {"ModerationLabels":[],"ModerationModelVersion":"7.0","ContentTypes":
  [{"Confidence":72.7257,"Name":"Animated"}]}}
```

Ringkasan manifes keluaran diformat sebagai berikut:

```
{
  "version": "1.0",           # Schema version, 1.0 for GA.
  "statistics": {
    "total-json-lines": Number, # Total number json lines (images) in the input
    manifest.
    "valid-json-lines": Number, # Total number of JSON Lines (images) that contain
    references to valid images.
    "invalid-json-lines": Number # Total number of invalid JSON Lines. These lines
    were not handled.
  },
  "errors": [
    {
      "line-numer": Number, # The number of the line in the manifest where the
      error occured.
      "source-ref": "String", # Optional. Name of the file if was parsed.
      "code": "String", # Error code.
      "message": "String" # Description of the error.
    }
  ]
}
```

Jenis konten

Informasi tentang jenis konten media yang dianalisis dengan `StartMediaAnalysisJob` operasi dikembalikan oleh `GetMediaAnalysisJob` operasi. `ContentType` dapat menjadi salah satu dari dua kategori yang berbeda:

- Konten animasi, yang mencakup video game dan animasi (misalnya, kartun, komik, manga, anime).
- Konten bergambar, yang meliputi menggambar, melukis, dan sketsa.

Verifikasi prediksi dan pelatihan adaptor

Analisis Massal juga dapat dimanfaatkan melalui konsol [Rekognition](#) untuk mendapatkan prediksi untuk sekumpulan gambar, memverifikasi prediksi ini, dan kemudian membuat adaptor menggunakan prediksi yang diverifikasi. Adaptor memungkinkan Anda untuk meningkatkan akurasi operasi Rekognition yang didukung.

Saat ini, Anda dapat membuat adaptor untuk digunakan dengan fitur Moderasi Kustom Rekognition. Dengan membuat adaptor dan menyediakannya untuk [DetectModerationLabels](#) operasi, Anda dapat mencapai akurasi yang lebih baik untuk tugas moderasi konten yang terkait dengan kasus penggunaan spesifik Anda.

Untuk informasi selengkapnya tentang Moderasi Kustom, lihat [Meningkatkan akurasi dengan Custom Moderation](#). Lihat [Analisis dan verifikasi massal](#) penjelasan tentang cara memverifikasi prediksi yang dibuat dengan analisis Massal. Untuk tutorial yang membahas cara menggunakan konsol Rekognition untuk memverifikasi prediksi dan membuat adaptor, lihat [Tutorial adaptor Moderasi Kustom](#)

Tutorial

Tutorial lintas-layanan ini menunjukkan cara menggunakan operasi API Rekognition bersama lainnya AWS layanan untuk membuat aplikasi sampel dan menyelesaikan berbagai tugas. Sebagian besar tutorial ini menggunakan Amazon S3 untuk menyimpan gambar atau video. Layanan lain yang umum digunakan termasuk AWS Lambda.

Topik

- [Menyimpan Data Amazon Rekognition dengan Amazon RDS dan DynamoDB](#)
- [Menggunakan Amazon Rekognition dan Lambda untuk menandai aset di bucket Amazon S3](#)
- [Membuat AWS aplikasi penganalisis video](#)
- [Membuat fungsi Amazon Rekognition Lambda](#)
- [Menggunakan Amazon Rekognition untuk Verifikasi Identitas](#)
- [Mendeteksi Label dalam Gambar Menggunakan Lambda dan Python](#)

Menyimpan Data Amazon Rekognition dengan Amazon RDS dan DynamoDB

Saat menggunakan API Amazon Rekognition, penting untuk diingat bahwa operasi API tidak menyimpan label yang dihasilkan. Anda dapat menyimpan label ini dengan menempatkannya dalam database, bersama dengan pengidentifikasi untuk gambar masing-masing.

Tutorial ini menunjukkan mendeteksi label dan menyimpan label yang terdeteksi ke database. Contoh aplikasi yang dikembangkan dalam tutorial ini akan membaca gambar dari [Amazon](#) ember memanggil [DetectLabels](#) operasi pada gambar-gambar ini, dan menyimpan label yang dihasilkan dalam database. Aplikasi akan menyimpan data baik dalam instans database Amazon RDS atau database DynamoDB, tergantung pada jenis database yang ingin Anda gunakan.

Anda akan menggunakan [AWS SDK untuk Python](#) atau tutorial ini. Anda juga dapat melihat [AWS Contoh SDK Dokumentasi GitHub repo](#) untuk tutorial Python lainnya.

Topik

- [Prasyarat](#)
- [Mendapatkan Label untuk Gambar di Amazon S3 Bucket](#)
- [Membuat Tabel Amazon DynamoDB](#)

- [Mengunggah Data ke DynamoDB](#)
- [Membuat Database MySQL di Amazon RDS](#)
- [Mengunggah Data ke Tabel MySQL Amazon RDS](#)

Prasyarat

Sebelum Anda memulai tutorial ini, Anda akan perlu menginstal Python dan menyelesaikan langkah-langkah yang diperlukan untuk [mengatur PythonAWSSDK](#). Di luar ini, pastikan bahwa Anda memiliki:

[Membuat akun AWS dan peran IAM](#)

[Menginstal SDK Python \(Boto3\)](#)

[Dikonfigurasi dengan benarAWSkredensi akses](#)

[Bucket Amazon S3 yang dibuat mengisinya dengan gambar](#)

[Membuat contoh database RDS](#), jika menggunakan RDS untuk menyimpan data

Mendapatkan Label untuk Gambar di Amazon S3 Bucket

Mulailah dengan menulis fungsi yang akan mengambil nama gambar di bucket Amazon S3 Anda dan mengambil gambar itu. Gambar ini akan ditampilkan untuk mengkonfirmasi bahwa gambar yang benar sedang diteruskan ke panggilan ke [DetectLabels](#) yang juga dalam fungsi.

1. Temukan bucket Amazon S3 yang ingin Anda gunakan dan tuliskan namanya. Anda akan melakukan panggilan ke bucket Amazon S3 ini dan membaca gambar di dalamnya. Pastikan bucket Anda berisi beberapa gambar untuk diteruskan ke [DetectLabels](#) operasi.
2. Tulis kode untuk terhubung ke bucket Amazon S3 Anda. Anda dapat terhubung ke sumber daya Amazon S3 dengan Boto3 untuk mengambil gambar dari bucket Amazon S3. Setelah terhubung ke sumber daya Amazon S3, Anda dapat mengakses bucket Anda dengan menyediakan metode Bucket dengan nama bucket Amazon S3 Anda. Setelah terhubung ke bucket Amazon S3, Anda mengambil gambar dari bucket dengan menggunakan metode Object. Dengan memanfaatkan Matplotlib, Anda dapat menggunakan koneksi ini untuk memvisualisasikan gambar Anda saat mereka memproses. Boto3 juga digunakan untuk terhubung ke klien Rekognition.

Dalam kode berikut, berikan wilayah Anda ke parameter `region_name`. Anda akan meneruskan nama bucket Amazon S3 dan nama gambar ke [DetectLabels](#), yang mengembalikan label

untuk gambar yang sesuai. Setelah memilih hanya label dari respons, nama gambar dan label dikembalikan.

```
import boto3
from io import BytesIO
from matplotlib import pyplot as plt
from matplotlib import image as mp_img

boto3 = boto3.Session()

def read_image_from_s3(bucket_name, image_name):

    # Connect to the S3 resource with Boto 3
    # get bucket and find object matching image name
    s3 = boto3.resource('s3')
    bucket = s3.Bucket(name=bucket_name)
    Object = bucket.Object(image_name)

    # Downloading the image for display purposes, not necessary for detection of
    labels
    # You can comment this code out if you don't want to visualize the images
    file_name = Object.key
    file_stream = BytesIO()
    Object.download_fileobj(file_stream)
    img = mp_img.imread(file_stream, format="jpeg")
    plt.imshow(img)
    plt.show()

    # get the labels for the image by calling DetectLabels from Rekognition
    client = boto3.client('rekognition', region_name="region-name")
    response = client.detect_labels(Image={'S3Object': {'Bucket': bucket_name,
'Name': image_name}},
                                   MaxLabels=10)

    print('Detected labels for ' + image_name)

    full_labels = response['Labels']

    return file_name, full_labels
```

3. Simpan kode ini dalam sebuah file bernama `get_images.py`.

Membuat Tabel Amazon DynamoDB

Kode berikut menggunakan Boto3 untuk terhubung ke DynamoDB dan menggunakan `DynamoDBCreateTable` metode untuk membuat tabel bernama `Images`. Tabel ini memiliki kunci primer komposit yang terdiri dari kunci partisi yang disebut `Image` dan kunci semacam yang disebut `Label`. Kunci `Image` berisi nama gambar, sedangkan tombol `Labels` menyimpan label yang ditetapkan ke `Image` tersebut.

```
import boto3

def create_new_table(dynamodb=None):
    dynamodb = boto3.resource(
        'dynamodb',)
    # Table defination
    table = dynamodb.create_table(
        TableName='Images',
        KeySchema=[
            {
                'AttributeName': 'Image',
                'KeyType': 'HASH' # Partition key
            },
            {
                'AttributeName': 'Labels',
                'KeyType': 'RANGE' # Sort key
            }
        ],
        AttributeDefinitions=[
            {
                'AttributeName': 'Image',
                'AttributeType': 'S'
            },
            {
                'AttributeName': 'Labels',
                'AttributeType': 'S'
            }
        ],
        ProvisionedThroughput={
            'ReadCapacityUnits': 10,
            'WriteCapacityUnits': 10
        }
    )
    return table
```

```
if __name__ == '__main__':
    device_table = create_new_table()
    print("Status:", device_table.table_status)
```

Simpan kode ini di editor dan jalankan sekali untuk membuat tabel DynamoDB.

Mengunggah Data ke DynamoDB

Sekarang database DynamoDB telah dibuat dan Anda memiliki fungsi untuk mendapatkan label untuk gambar, Anda dapat menyimpan label di DynamoDB. Kode berikut mengambil semua gambar dalam bucket S3, mendapatkan label untuk mereka, dan menyimpan data di DynamoDB.

1. Anda harus menulis kode untuk mengunggah data ke DynamoDB. Sebuah fungsi yang disebut `get_image_names` digunakan untuk terhubung ke bucket Amazon S3 Anda dan mengembalikan nama semua gambar dalam bucket sebagai daftar. Anda akan melewati daftar ini ke dalam `read_image_from_S3` fungsi, yang diimpor dari `get_images.py` file yang Anda buat.

```
import boto3
import json
from get_images import read_image_from_s3

boto3 = boto3.Session()

def get_image_names(name_of_bucket):

    s3_resource = boto3.resource('s3')
    my_bucket = s3_resource.Bucket(name_of_bucket)
    file_list = []
    for file in my_bucket.objects.all():
        file_list.append(file.key)
    return file_list
```

2. Yang `read_image_from_S3` fungsi yang kita buat sebelumnya akan mengembalikan nama gambar yang sedang diproses dan kamus label yang terkait dengan gambar itu. Sebuah fungsi yang disebut `find_values` digunakan untuk mendapatkan hanya label dari respon. Nama gambar dan labelnya kemudian siap untuk diunggah ke tabel DynamoDB Anda.

```
def find_values(id, json_repr):
    results = []
```

```
def _decode_dict(a_dict):
    try:
        results.append(a_dict[id])
    except KeyError:
        pass
    return a_dict

json.loads(json_repr, object_hook=_decode_dict) # Return value ignored.
return results
```

3. Anda akan menggunakan fungsi ketiga, yang disebut `load_data`, untuk benar-benar memuat gambar dan label ke dalam tabel DynamoDB yang Anda buat.

```
def load_data(image_labels, dynamodb=None):

    if not dynamodb:
        dynamodb = boto3.resource('dynamodb')

    table = dynamodb.Table('Images')

    print("Adding image details:", image_labels)
    table.put_item(Item=image_labels)
    print("Success!!")
```

4. Di sinilah ketiga fungsi yang kita definisikan sebelumnya dipanggil, dan operasi dilakukan. Tambahkan tiga fungsi yang didefinisikan di atas, bersama dengan kode di bawah ini, ke file Python. Jalankan kode tersebut.

```
bucket = "bucket_name"
file_list = get_image_names(bucket)

for file in file_list:
    file_name = file
    print("Getting labels for " + file_name)
    image_name, image_labels = read_image_from_s3(bucket, file_name)
    image_json_string = json.dumps(image_labels, indent=4)
    labels=set(find_values("Name", image_json_string))
    print("Labels found: " + str(labels))
    labels_dict = {}
    print("Saving label data to database")
    labels_dict["Image"] = str(image_name)
    labels_dict["Labels"] = str(labels)
    print(labels_dict)
```



```
load_data(labels_dict)
print("Success!")
```

Anda baru saja menggunakan [DetectLabels](#) untuk menghasilkan label untuk gambar Anda dan menyimpan label tersebut dalam instance DynamoDB. Pastikan bahwa Anda merobohkan semua sumber daya yang Anda buat saat akan melalui tutorial ini. Itu akan mencegah Anda ditagih untuk sumber daya yang tidak Anda gunakan.

Membuat Database MySQL di Amazon RDS

Sebelum melangkah lebih jauh, pastikan Anda telah menyelesaikan [prosedur pengaturan](#) untuk Amazon RDS dan [membuat contoh MySQL DB](#) menggunakan Amazon RDS.

Kode berikut memanfaatkan [PyMySQL](#) library dan instans DB Amazon RDS Anda. Ini menciptakan tabel untuk memegang nama-nama gambar Anda dan label yang terkait dengan gambar-gambar. Amazon RDS menerima perintah untuk membuat tabel dan memasukkan data ke dalam tabel. Untuk menggunakan Amazon RDS, Anda harus terhubung ke host Amazon RDS menggunakan nama host, nama pengguna, dan kata sandi Anda. Anda akan terhubung ke Amazon RDS dengan memberikan argumen ini `PyMySQL.connect` fungsi dan menciptakan sebuah instance dari cursor.

1. Dalam kode berikut, ganti nilai host dengan titik akhir host Amazon RDS Anda dan ganti nilai pengguna dengan nama pengguna utama yang terkait dengan instans Amazon RDS Anda. Anda juga perlu mengganti kata sandi dengan kata sandi utama untuk pengguna utama Anda.

```
import pymysql

host = "host-endpoint"
user = "username"
password = "master-password"
```

2. Buat database dan tabel untuk memasukkan data gambar dan label Anda. Lakukan ini dengan menjalankan dan melakukan kueri pembuatan. Kode berikut membuat database. Jalankan kode ini hanya sekali.

```
conn = pymysql.connect(host=host, user=user, passwd=password)
print(conn)
cursor = conn.cursor()
print("Connection successful")
```

```
# run once
create_query = "create database rekogDB1"
print("Creation successful!")
cursor.execute(create_query)
cursor.connection.commit()
```

3. Setelah database telah dibuat, Anda harus membuat tabel untuk memasukkan nama gambar Anda dan label ke dalam. Untuk membuat tabel, Anda akan terlebih dahulu meneruskan perintah use SQL, bersama dengan nama database Anda, ke execute fungsi. Setelah koneksi dibuat, query untuk membuat tabel dijalankan. Kode berikut terhubung ke database dan kemudian membuat tabel dengan kedua kunci primer, yang disebut image_id, dan atribut teks menyimpan label. Gunakan impor dan variabel yang Anda tentukan sebelumnya, dan jalankan kode ini untuk membuat tabel di database Anda.

```
# connect to existing DB
cursor.execute("use rekogDB1")
cursor.execute("CREATE TABLE IF NOT EXISTS test_table(image_id VARCHAR (255)
PRIMARY KEY, image_labels TEXT)")
conn.commit()
print("Table creation - Successful creation!")
```

Mengunggah Data ke Tabel MySQL Amazon RDS

Setelah membuat database Amazon RDS dan tabel dalam database, Anda bisa mendapatkan label untuk gambar Anda dan menyimpan label tersebut di database Amazon RDS.

1. Hubungkan ke bucket Amazon S3 Anda dan ambil nama semua gambar di bucket. Nama-nama gambar ini akan diteruskan ke read_image_from_s3 fungsi yang Anda buat sebelumnya untuk mendapatkan label untuk semua gambar Anda. Kode berikut terhubung ke bucket Amazon S3 Anda dan mengembalikan daftar semua gambar di bucket Anda.

```
import pymysql
from get_images import read_image_from_s3
import json
import boto3

host = "host-endpoint"
user = "username"
password = "master-password"
```

```
conn = pymysql.connect(host=host, user=user, passwd=password)
print(conn)
cursor = conn.cursor()
print("Connection successful")

def get_image_names(name_of_bucket):

    s3_resource = boto3.resource('s3')
    my_bucket = s3_resource.Bucket(name_of_bucket)
    file_list = []
    for file in my_bucket.objects.all():
        file_list.append(file.key)
    return file_list
```

2. Respon dari [DetectLabels](#) API berisi lebih dari sekedar label, jadi tulis fungsi untuk mengekstrak hanya nilai label. Fungsi berikut mengembalikan daftar penuh hanya label.

```
def find_values(id, json_repr):
    results = []

    def _decode_dict(a_dict):
        try:
            results.append(a_dict[id])
        except KeyError:
            pass
        return a_dict

    json.loads(json_repr, object_hook=_decode_dict) # Return value ignored.
    return results
```

3. Anda akan memerlukan fungsi untuk memasukkan nama gambar dan label ke meja Anda. Fungsi berikut menjalankan query penyisipan dan menyisipkan setiap pasangan tertentu nama gambar dan label.

```
def upload_data(image_id, image_labels):

    # insert into db
    cursor.execute("use rekogDB1")
    query = "INSERT IGNORE INTO test_table(image_id, image_labels) VALUES (%s, %s)"
    values = (image_id, image_labels)
    cursor.execute(query, values)
    conn.commit()
```

```
print("Insert successful!")
```

4. Akhirnya, Anda harus menjalankan fungsi yang Anda tentukan di atas. Dalam kode berikut, nama-nama semua gambar di bucket Anda dikumpulkan dan diberikan ke fungsi yang memanggil [DetectLabels](#). Setelah itu, label dan nama gambar yang mereka terapkan diunggah ke database Amazon RDS Anda. Salin tiga fungsi yang didefinisikan di atas, bersama dengan kode di bawah ini, ke dalam file Python. Jalankan file Python.

```
bucket = "bucket-name"
file_list = get_image_names(bucket)

for file in file_list:
    file_name = file
    print("Getting labels for " + file_name)
    image_name, image_labels = read_image_from_s3(bucket, file_name)
    image_json = json.dumps(image_labels, indent=4)
    labels=set(find_values("Name", image_json))
    print("Labels found: " + str(labels))
    unique_labels=set(find_values("Name", image_json))
    print(unique_labels)
    image_name_string = str(image_name)
    labels_string = str(unique_labels)
    upload_data(image_name_string, labels_string)
    print("Success!")
```

Anda telah berhasil menggunakan `DetectLabels` untuk menghasilkan label untuk gambar Anda dan menyimpan label tersebut dalam database MySQL menggunakan Amazon RDS. Pastikan bahwa Anda merobohkan semua sumber daya yang Anda buat saat akan melalui tutorial ini. Ini akan mencegah Anda ditagih untuk sumber daya yang tidak Anda gunakan.

Untuk lebih AWS multiservice contoh, lihat [AWS Contoh SDK Dokumentasi GitHub repositori](#).

Menggunakan Amazon Rekognition dan Lambda untuk menandai aset di bucket Amazon S3

Dalam tutorial ini, Anda membuat AWS Lambda fungsi yang secara otomatis menandai aset digital yang terletak di bucket Amazon S3. Fungsi Lambda membaca semua objek dalam bucket Amazon S3 yang diberikan. Untuk setiap objek dalam bucket, itu meneruskan citra ke layanan Amazon Rekognition untuk menghasilkan serangkaian label. Setiap label digunakan untuk membuat tanda


yang diterapkan pada citra. Setelah Anda menjalankan fungsi Lambda, itu secara otomatis membuat tanda berdasarkan semua citra dalam bucket Amazon S3 yang diberikan dan menerapkannya pada citra.

Sebagai contoh, asumsikan Anda menjalankan fungsi Lambda dan Anda memiliki citra ini dalam bucket Amazon S3.



Aplikasi kemudian secara otomatis membuat tanda dan menerapkannya pada citra.

Tags (6)

Track storage cost of other criteria by tagging your objects. [Learn more](#) 

Key	Value
Nature	99.99188
Volcano	97.60948
Eruption	96.54574
Lava	79.63064
Mountain	99.99188
Outdoors	99.99188

Note

Layanan yang Anda gunakan dalam tutorial ini adalah bagian dari Tingkat AWS Gratis. Ketika Anda telah menyelesaikan tutorial, kami merekomendasikan untuk mengakhiri sumber daya apa pun yang Anda buat selama tutorial sehingga Anda tidak dikenakan biaya.

Tutorial ini menggunakan AWS SDK for Java versi 2. Lihat [GitHub repositori contoh SDK AWS Dokumentasi](#) untuk tutorial Java V2 tambahan.

Topik

- [Prasyarat](#)
- [Konfigurasi peran Lambda IAM](#)
- [Buat proyek](#)
- [Tulis kode](#)
- [Kemas proyek](#)
- [Deploy fungsi Lambda](#)
- [Uji metode Lambda](#)

Prasyarat

Sebelum memulai, Anda perlu menyelesaikan langkah-langkah dalam [Menyiapkan AWS SDK for Java](#). Kemudian pastikan bahwa Anda memiliki hal berikut ini:

- Java 1.8 JDK.
- Maven 3.6 atau lebih tinggi.
- Bucket [Amazon S3](#) dengan 5-7 citra alam di dalamnya. Citra ini dibaca oleh fungsi Lambda.

Konfigurasi peran Lambda IAM

Tutorial ini menggunakan layanan Amazon Rekognition dan Amazon S3. Konfigurasi peran lambda-support untuk memiliki kebijakan yang memungkinkannya mengaktifkan layanan ini dari fungsi Lambda.

Untuk mengonfigurasi peran

1. Masuk ke AWS Management Console dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi, pilih Peran, lalu pilih Buat Peran.
3. Pilih Layanan AWS , lalu pilih Lambda.
4. Pilih tab Izin.
5. Cari AWSLambdaBasicExecutionRole.
6. Pilih Tanda selanjutnya.
7. Pilih Tinjau.

8. Namai peran dengan lambda-support.
9. Pilih Buat peran.
10. Pilih lambda-support untuk melihat halaman gambaran umum.
11. Pilih Pasang kebijakan.
12. Pilih AmazonRekognitionFullAccessdari daftar kebijakan.
13. Pilih Lampirkan kebijakan.
14. Cari AmazonS3 FullAccess, lalu pilih Lampirkan kebijakan.

Buat proyek

Buat proyek Java baru, kemudian konfigurasi Maven pom.xml dengan pengaturan yang diperlukan dan dependensi. Pastikan file pom.xml Anda terlihat seperti berikut ini:

```
<?xml version="1.0" encoding="UTF-8"?>
  <project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>WorkflowTagAssets</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>java-basic-function</name>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.10.54</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
```

```
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-lambda-java-core</artifactId>
    <version>1.2.1</version>
  </dependency>
  <dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.6</version>
  </dependency>
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.10.0</version>
  </dependency>
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.13.0</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-slf4j18-impl</artifactId>
    <version>2.13.3</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.6.0</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>5.6.0</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>com.googlecode.json-simple</groupId>
    <artifactId>json-simple</artifactId>
    <version>1.1.1</version>
```

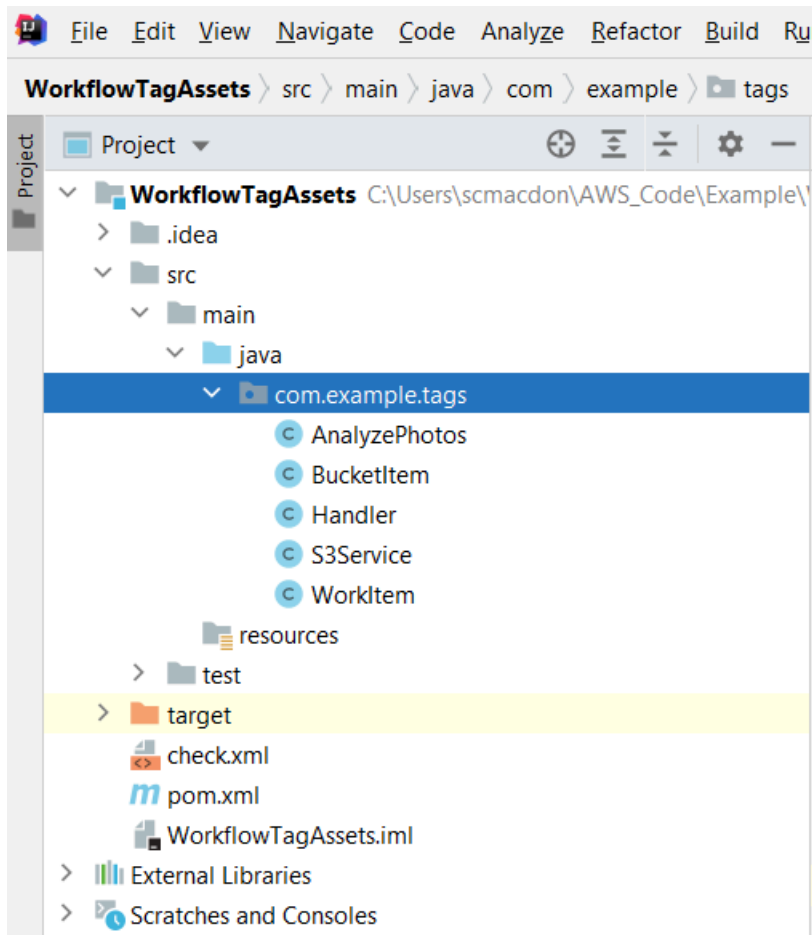


```
</dependency>
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
</dependency>
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>rekognition</artifactId>
</dependency>
</dependencies>
<build>
  <plugins>
    <plugin>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.22.2</version>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-shade-plugin</artifactId>
      <version>3.2.2</version>
      <configuration>
        <createDependencyReducedPom>>false</createDependencyReducedPom>
      </configuration>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>shade</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```

```
</project>
```

Tulis kode

Gunakan AWS Lambda runtime Java API untuk membuat kelas Java yang mendefinisikan fungsi Lambda. Dalam contoh ini, ada satu kelas Java untuk fungsi Lambda bernama Handler dan kelas tambahan yang diperlukan untuk kasus penggunaan ini. Citra berikut menunjukkan kelas Java dalam proyek. Perhatikan bahwa semua kelas Java terletak di sebuah paket bernama `com.example.tags`.



Buat kelas Java berikut untuk kode:

- Handler menggunakan Lambda Java run-time API dan melakukan kasus penggunaan yang dijelaskan dalam tutorial ini. AWS Logis aplikasi yang dijalankan terletak di metode `handleRequest`.
- S3Service menggunakan API Amazon S3 untuk melakukan operasi S3.
- AnalyzePhotos menggunakan Amazon Rekognition API untuk menganalisis gambar.
- BucketItem mendefinisikan model yang menyimpan informasi bucket Amazon S3.
- WorkItem mendefinisikan model yang menyimpan data Amazon Rekognition.

Kelas handler

Kode Java ini mewakili kelas Handler. Kelas membaca bendera yang diteruskan ke fungsi Lambda. Layanan S3. ListBucketObjectsmetode mengembalikan objek Daftar di mana setiap elemen adalah nilai string yang mewakili kunci objek. Jika nilai bendera benar, maka tanda diterapkan dengan mengiterasi melalui daftar dan menerapkan tanda ke setiap objek dengan memanggil metode s3Service.tagAssets. Jika nilai flag adalah false, maka S3service. deleteTagFromMetode objek dipanggil yang menghapus tag. Juga, perhatikan bahwa Anda dapat mencatat pesan ke CloudWatch log Amazon dengan menggunakan LambdaLoggerobjek.

Note

Pastikan Anda menetapkan nama bucket ke variabel bucketName.

```
package com.example.tags;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

public class Handler implements RequestHandler<Map<String,String>, String> {

    @Override
    public String handleRequest(Map<String, String> event, Context context) {
        LambdaLogger logger = context.getLogger();
        String delFlag = event.get("flag");
        logger.log("FLAG IS: " + delFlag);
        S3Service s3Service = new S3Service();
        AnalyzePhotos photos = new AnalyzePhotos();

        String bucketName = "<Enter your bucket name>";
        List<String> myKeys = s3Service.listBucketObjects(bucketName);
        if (delFlag.compareTo("true") == 0) {

            // Create a List to store the data.
            List<ArrayList<WorkItem>> myList = new ArrayList<>();
```

```
// loop through each element in the List and tag the assets.
for (String key : myKeys) {

    byte[] keyData = s3Service.getObjectBytes(bucketName, key);

    // Analyze the photo and return a list where each element is a WorkItem.
    ArrayList<WorkItem> item = photos.detectLabels(keyData, key);
    myList.add(item);
}

s3Service.tagAssets(myList, bucketName);
logger.log("All Assets in the bucket are tagged!");

} else {

    // Delete all object tags.
    for (String key : myKeys) {
        s3Service.deleteTagFromObject(bucketName, key);
        logger.log("All Assets in the bucket are deleted!");
    }
}
return delFlag;
}
}
```

Kelas S3Service

Kelas berikut menggunakan API Amazon S3 untuk melakukan operasi S3. Misalnya, getObjectBytes metode mengembalikan array byte yang mewakili gambar. Demikian juga, listBucketObjects metode mengembalikan objek List di mana setiap elemen adalah nilai string yang menentukan nama kunci.

```
package com.example.tags;

import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Object;
```

```
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.services.s3.model.Tagging;
import software.amazon.awssdk.services.s3.model.Tag;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectTaggingRequest;

public class S3Service {

    private S3Client getClient() {

        Region region = Region.US_WEST_2;
        return S3Client.builder()
            .region(region)
            .build();
    }

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

        S3Client s3 = getClient();

        try {

            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            // Return the byte[] from this object.
            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            return objectBytes.asByteArray();

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

    // Returns the names of all images in the given bucket.
```

```
public List<String> listBucketObjects(String bucketName) {

    S3Client s3 = getClient();
    String keyName;

    List<String> keys = new ArrayList<>();

    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();

        for (S3Object myValue: objects) {
            keyName = myValue.key();
            keys.add(keyName);
        }
        return keys;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

// Tag assets with labels in the given list.
public void tagAssets(List myList, String bucketName) {

    try {

        S3Client s3 = getClient();
        int len = myList.size();

        String assetName = "";
        String labelName = "";
        String labelValue = "";

        // Tag all the assets in the list.
        for (Object o : myList) {
```

```
        // Need to get the WorkItem from each list.
        List innerList = (List) o;
        for (Object value : innerList) {

            WorkItem workItem = (WorkItem) value;
            assetName = workItem.getKey();
            labelName = workItem.getName();
            labelValue = workItem.getConfidence();
            tagExistingObject(s3, bucketName, assetName, labelName, labelValue);
        }
    }

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

// This method tags an existing object.
private void tagExistingObject(S3Client s3, String bucketName, String key, String
label, String LabelValue) {

    try {

        // First need to get existing tag set; otherwise the existing tags are
        overwritten.
        GetObjectTaggingRequest getObjectTaggingRequest =
        GetObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        GetObjectTaggingResponse response =
        s3.getObjectTagging(getObjectTaggingRequest);

        // Get the existing immutable list - cannot modify this list.
        List<Tag> existingList = response.getTagSet();
        ArrayList<Tag> newTagList = new ArrayList(new ArrayList<>(existingList));

        // Create a new tag.
        Tag myTag = Tag.builder()
            .key(label)
            .value(LabelValue)
            .build();
```

```
// push new tag to list.
newTagList.add(myTag);
Tagging tagging = Tagging.builder()
    .tagSet(newTagList)
    .build();

PutObjectTaggingRequest taggingRequest = PutObjectTaggingRequest.builder()
    .key(key)
    .bucket(bucketName)
    .tagging(tagging)
    .build();

s3.putObjectTagging(taggingRequest);
System.out.println(key + " was tagged with " + label);

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

// Delete tags from the given object.
public void deleteTagFromObject(String bucketName, String key) {

    try {

        DeleteObjectTaggingRequest deleteObjectTaggingRequest =
DeleteObjectTaggingRequest.builder()
            .key(key)
            .bucket(bucketName)
            .build();

        S3Client s3 = getClient();
        s3.deleteObjectTagging(deleteObjectTaggingRequest);

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


AnalyzePhotos kelas

Kode Java berikut mewakili AnalyzePhotoskelas. Kelas ini menggunakan API Amazon Rekognition untuk menganalisis citra.

```
package com.example.tags;

import software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.ArrayList;
import java.util.List;

public class AnalyzePhotos {

    // Returns a list of WorkItem objects that contains labels.
    public ArrayList<WorkItem> detectLabels(byte[] bytes, String key) {

        Region region = Region.US_EAST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .region(region)
            .build();

        try {

            SdkBytes sourceBytes = SdkBytes.fromByteArray(bytes);

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

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

```
        DetectLabelsResponse labelsResponse =
rekClient.detectLabels(detectLabelsRequest);

        // Write the results to a WorkItem instance.
        List<Label> labels = labelsResponse.labels();
        ArrayList<WorkItem> list = new ArrayList<>();
        WorkItem item ;
        for (Label label: labels) {
            item = new WorkItem();
            item.setKey(key); // identifies the photo.
            item.setConfidence(label.confidence().toString());
            item.setName(label.name());
            list.add(item);
        }
        return list;

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

BucketItem kelas

Kode Java berikut mewakili BucketItem kelas yang menyimpan data objek Amazon S3.

```
package com.example.tags;

public class BucketItem {

    private String key;
    private String owner;
    private String date ;
    private String size ;

    public void setSize(String size) {
        this.size = size ;
    }
}
```

```
public String getSize() {
    return this.size ;
}

public void setDate(String date) {
    this.date = date ;
}

public String getDate() {
    return this.date ;
}

public void setOwner(String owner) {
    this.owner = owner ;
}

public String getOwner() {
    return this.owner ;
}

public void setKey(String key) {
    this.key = key ;
}

public String getKey() {
    return this.key ;
}
}
```

WorkItem kelas

Kode Java berikut mewakili WorkItemkelas.

```
package com.example.tags;

public class WorkItem {

    private String key;
    private String name;
    private String confidence ;

    public void setKey (String key) {
        this.key = key;
    }
}
```

```
}

public String getKey() {
    return this.key;
}

public void setName (String name) {
    this.name = name;
}

public String getName() {
    return this.name;
}

public void setConfidence (String confidence) {
    this.confidence = confidence;
}

public String getConfidence() {
    return this.confidence;
}
}
```

Kemas proyek

Kemas proyek ke dalam file .jar (JAR) dengan menggunakan perintah Maven berikut.

```
mvn package
```

File JAR terletak di folder target (yang merupakan folder anak dari folder proyek).

Name	Date modified	Type	Size
classes	3/31/2021 9:47 AM	File folder	
generated-sources	3/30/2021 8:36 AM	File folder	
generated-test-sources	3/30/2021 12:01 PM	File folder	
maven-archiver	3/30/2021 12:01 PM	File folder	
maven-status	3/30/2021 12:01 PM	File folder	
test-classes	3/30/2021 12:01 PM	File folder	
checkstyle-cachefile	3/31/2021 9:31 AM	File	1 KB
checkstyle-checker.xml	3/31/2021 9:31 AM	XML Document	1 KB
checkstyle-result.xml	3/31/2021 9:31 AM	XML Document	1 KB
original-WorkflowTagAssets-1.0-SNAPSHOT.jar	3/31/2021 9:47 AM	Executable Jar File	11 KB
WorkflowTagAssets-1.0-SNAPSHOT.jar	3/31/2021 9:47 AM	Executable Jar File	12,866 KB
WorkflowTagAssets-1.0-SNAPSHOT-shaded.jar	3/31/2021 9:47 AM	Executable Jar File	12,866 KB

Note

Perhatikan penggunaan maven-shade-plugin dalam file POM proyek. Plugin ini bertanggung jawab untuk membuat JAR yang berisi dependensi yang diperlukan. Jika Anda mencoba untuk mengemas proyek tanpa plugin ini, dependensi yang diperlukan tidak termasuk dalam file JAR dan Anda akan menemukan file. `ClassNotFoundException`

Deploy fungsi Lambda

1. Buka [Konsol Lambda](#).
2. Pilih Buat Fungsi.
3. Pilih Penulis dari scratch.
4. Di bagian Informasi dasar, masukkan cron sebagai nama.
5. Di Waktu aktif, pilih Java 8.
6. Pilih Gunakan peran yang sudah ada, lalu pilih lambda-support (IAM role yang telah Anda buat).
7. Pilih Buat fungsi.
8. Untuk Tipe entri kode, pilih Unggah file .zip atau .jar.
9. Pilih Unggah, lalu jelajahi ke file JAR yang telah Anda buat.
10. Untuk Handler, masukkan nama fungsi yang memenuhi syarat, misalnya, menentukan paket `com.example.tags.Handler:handleRequest` (`com.example.tags`, `Handler` adalah kelas yang diikuti oleh `::` dan nama metode).

11 Pilih Simpan.

Uji metode Lambda

Pada titik dalam tutorial ini, Anda dapat menguji fungsi Lambda.

1. Dalam konsol Lambda, klik tab Uji lalu masukkan JSON berikut.

```
{
  "flag": "true"
}
```

Code **Test** Monitor Configuration Aliases Versions

Test event Delete Format Save changes Invoke

Invoke your function with a test event. Choose a template that matches the service that triggers your function, or enter your event document in JSON.

New event

Saved event

Saved event

deleteTest

```
1 {
2   "flag": "true"
3 }
```

Note

Meneruskan tanda BETUL aset digital dan teruskan menghapus tanda SALAH.

2. Pilih tombol Panggil. Setelah fungsi Lambda dipanggil, Anda melihat pesan berhasil.

Code **Test** Monitor Configuration Aliases Versions

Execution result: succeeded (logs) X

► Details

Selamat, Anda telah membuat AWS Lambda fungsi yang secara otomatis menerapkan tag ke aset digital yang terletak di bucket Amazon S3. Sebagaimana dinyatakan di awal tutorial ini, pastikan untuk mengakhiri semua sumber daya yang telah Anda buat saat akan melalui tutorial ini untuk memastikan Anda tidak dikenakan biaya.

Untuk contoh AWS multiservice lainnya, lihat repositori contoh [SDK AWS Dokumentasi](#). GitHub

Membuat AWS aplikasi penganalisis video

Anda dapat membuat aplikasi web Java yang menganalisis video untuk deteksi label dengan menggunakan AWS SDK for Java versi 2. Aplikasi yang dibuat dalam AWS tutorial ini memungkinkan Anda mengunggah video (file MP4) ke ember Amazon S3. Kemudian aplikasi menggunakan layanan Amazon Rekognition untuk menganalisis video. Hasilnya digunakan untuk mengisi model data dan kemudian laporan dihasilkan dan dikirim melalui email ke pengguna tertentu dengan menggunakan Amazon Simple Email Service.

Ilustrasi berikut menunjukkan laporan yang dihasilkan setelah aplikasi selesai menganalisis video. Kolom pada tabel di bawah ini menunjukkan Rentang Usia, Jenggot, Kacamata mata, dan Mata terbuka, serta nilai kepercayaan untuk prediksi atribut yang berbeda.

1	<u>Age Range</u>	<u>Beard</u>	<u>Eye glasses</u>	<u>Eyes open</u>
2				
3	AgeRange(Low=38, High=56)	Beard(Value=false, Confidence=83.07253)	Eyeglasses(Value=true, Confidence=55.965977)	EyeOpen(Value=true, Confidence=94.691696)
4	AgeRange(Low=36, High=52)	Beard(Value=true, Confidence=50.721912)	Eyeglasses(Value=false, Confidence=63.886036)	EyeOpen(Value=true, Confidence=95.906364)
5	AgeRange(Low=51, High=69)	Beard(Value=true, Confidence=58.38352)	Eyeglasses(Value=false, Confidence=96.39576)	EyeOpen(Value=true, Confidence=53.580643)
6	AgeRange(Low=49, High=67)	Beard(Value=false, Confidence=81.41662)	Eyeglasses(Value=true, Confidence=65.28722)	EyeOpen(Value=true, Confidence=95.11523)
7	AgeRange(Low=51, High=69)	Beard(Value=true, Confidence=61.533833)	Eyeglasses(Value=false, Confidence=97.51163)	EyeOpen(Value=true, Confidence=82.21834)
8	AgeRange(Low=29, High=45)	Beard(Value=false, Confidence=74.22591)	Eyeglasses(Value=true, Confidence=64.906685)	EyeOpen(Value=true, Confidence=98.48175)
9	AgeRange(Low=51, High=69)	Beard(Value=true, Confidence=65.9394)	Eyeglasses(Value=false, Confidence=94.14824)	EyeOpen(Value=true, Confidence=94.857346)
10	AgeRange(Low=44, High=62)	Beard(Value=true, Confidence=78.648)	Eyeglasses(Value=true, Confidence=65.83134)	EyeOpen(Value=true, Confidence=98.538666)
11				

Dalam tutorial ini, Anda membuat aplikasi Spring Boot yang memanggil berbagai AWS layanan. API Spring Boot digunakan untuk membangun sebuah model, pandangan yang berbeda, dan pengendali. Untuk informasi selengkapnya, lihat [Musim semi boot](#).

Layanan ini menggunakan AWS layanan berikut:

- Amazon Rekognition
- [Amazon S3](#)
- [Amazon SES](#)
- [AWS Elastic Beanstalk](#)

AWS Layanan yang termasuk dalam tutorial ini termasuk dalam Tingkat AWS Gratis. Kami merekomendasikan Anda untuk mengakhiri semua sumber daya yang Anda buat dalam tutorial ketika Anda selesai dengan sumber daya untuk menghindari pembayaran biaya.

Prasyarat

Sebelum memulai, Anda perlu menyelesaikan langkah-langkah dalam [Menyiapkan AWS SDK for Java](#). Kemudian pastikan bahwa Anda memiliki hal berikut ini:

- Java 1.8 JDK.
- Maven 3.6 atau lebih baru.
- Bucket Amazon S3 bernama video[somevalue]. Pastikan untuk menggunakan nama bucket ini dalam kode Amazon S3 Java Anda. Untuk informasi selengkapnya, lihat [Membuat bucket](#).
- IAM role. Anda membutuhkan ini untuk VideoDetectFaceskelas yang akan Anda buat. Untuk informasi selengkapnya, lihat [Mengonfigurasi Amazon Rekognition Video](#).
- Topik Amazon SNS yang valid. Anda membutuhkan ini untuk VideoDetectFaceskelas yang akan Anda buat. Untuk informasi selengkapnya, lihat [Mengonfigurasi Amazon Rekognition Video](#).

Prosedur

Dalam proses tutorial, Anda melakukan hal berikut:

1. Buat proyek
2. Tambahkan dependensi POM ke proyek Anda
3. Buat kelas Java
4. Buat file HTML
5. Buat file skrip
6. Paketkan proyek ke dalam file JAR
7. Menyebarkan aplikasi ke AWS Elastic Beanstalk

Untuk melanjutkan tutorial, ikuti petunjuk rinci dalam [GitHub repositori contoh SDK AWS Dokumentasi](#).

Membuat fungsi Amazon Rekognition Lambda

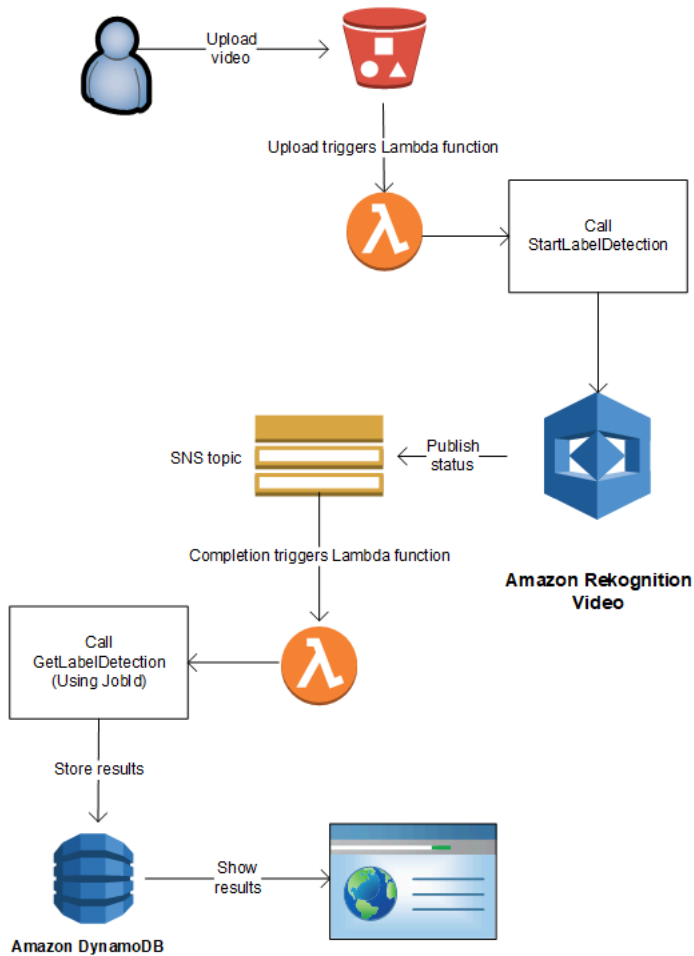
Tutorial ini menunjukkan bagaimana mendapatkan hasil dari operasi analisis video untuk pendeteksi label dengan menggunakan fungsi Java Lambda.

Note

Tutorial ini menggunakan AWS SDK for Java 1.x. Untuk tutorial menggunakan Rekognition dan SDK AWS for Java versi 2, [AWS lihat](#) contoh repositori Documentation SDK. GitHub

Anda dapat menggunakan fungsi Lambda dengan operasi Amazon Rekognition Video. Misalnya, diagram berikut menunjukkan situs web yang menggunakan fungsi Lambda untuk secara otomatis memulai analisis video saat diunggah ke bucket Amazon S3. Ketika fungsi Lambda dipicu, ia memanggil [StartLabelDetection](#) untuk mulai mendeteksi label dalam video yang diunggah. Untuk informasi tentang penggunaan Lambda untuk memproses notifikasi peristiwa dari bucket Amazon S3, lihat [Menggunakan AWS Lambda dengan Amazon S3 Events](#).

Fungsi Lambda kedua dipicu ketika status penyelesaian analisis dikirim ke topik Amazon SNS terdaftar. Fungsi Lambda kedua memanggil [GetLabelDetection](#) untuk mendapatkan hasil analisis. Hasilnya kemudian disimpan dalam basis data untuk persiapan menampilkannya pada halaman web. Fokus dari tutorial ini ada pada Fungsi lambda kedua.



Dalam tutorial ini, fungsi Lambda dipicu ketika Amazon Rekognition Video mengirimkan status penyelesaian analisis video untuk topik Amazon SNS yang terdaftar. Kemudian mengumpulkan hasil analisis video dengan menelepon [GetLabelDetection](#). Untuk tujuan demonstrasi, tutorial ini menulis hasil deteksi label ke CloudWatch log. Dalam fungsi Lambda aplikasi Anda, Anda harus menyimpan hasil analisis untuk digunakan nanti. Misalnya, Anda dapat menggunakan Amazon DynamoDB untuk menyimpan hasil analisis. Untuk informasi selengkapnya, lihat [Bekerja dengan DynamoDB](#).

Prosedur berikut menunjukkan kepada Anda cara:

- Buat topik Amazon SNS dan atur izin.
- Buat fungsi Lambda dengan menggunakan AWS Management Console dan berlangganan ke topik Amazon SNS.
- Konfigurasi fungsi Lambda dengan menggunakan AWS Management Console.
- Tambahkan kode sampel ke AWS Toolkit for Eclipse proyek dan unggah ke fungsi Lambda.
- Uji fungsi Lambda dengan menggunakan AWS CLI.

Note

Gunakan AWS Wilayah yang sama di seluruh tutorial.

Prasyarat

Tutorial ini mengasumsikan bahwa Anda sudah familier dengan AWS Toolkit for Eclipse. Untuk informasi selengkapnya, lihat [AWS Toolkit for Eclipse](#).

Buat topik SNS

Status penyelesaian operasi analisis Amazon Rekognition Video dikirim ke topik Amazon SNS. Prosedur ini menciptakan topik Amazon SNS dan peran layanan IAM yang memberikan akses Amazon Rekognition Video ke topik Amazon SNS Anda. Untuk informasi selengkapnya, lihat [Memanggil operasi Amazon Rekognition Video](#).

Untuk membuat topik Amazon SNS

1. Jika belum, buat peran layanan IAM untuk memberikan akses Amazon Rekognition Video ke topik Amazon SNS Anda. Perhatikan Amazon Resource Name (ARN). Untuk informasi selengkapnya, lihat [Memberikan akses ke beberapa topik Amazon SNS](#).
2. [Buat topik Amazon SNS](#) dengan menggunakan [Konsol Amazon SNS](#) Anda hanya perlu menentukan nama topik. Tambahkan nama topik dengan `AmazonRekognition`. Perhatikan topik ARN.

Buat fungsi Lambda

Anda membuat fungsi Lambda dengan menggunakan AWS Management Console. Kemudian Anda menggunakan proyek AWS Toolkit for Eclipse untuk mengunggah paket fungsi Lambda ke AWS Lambda. Hal ini juga memungkinkan untuk membuat fungsi Lambda dengan AWS Toolkit for Eclipse. Untuk informasi selengkapnya, lihat [Tutorial: Cara Membuat, Mengunggah, dan Memanggil Fungsi AWS Lambda](#).

Untuk membuat fungsi Lambda

1. Masuk ke Konsol Manajemen AWS dan buka konsol AWS Lambda di <https://console.aws.amazon.com/lambda/>.

2. Pilih Buat fungsi.
3. Pilih Penulis dari scratch.
4. Di Nama fungsi, ketikkan nama untuk fungsi Anda.
5. Di Waktu aktif, pilih Java 8.
6. Pilih Memilih atau membuat peran eksekusi.
7. Dalam Peran eksekusi, pilih Membuat peran baru dengan izin Lambda dasar.
8. Perhatikan nama peran baru yang ditampilkan di bagian bawah bagian Informasi dasar.
9. Pilih Buat fungsi.

Konfigurasi fungsi Lambda

Setelah Anda membuat fungsi Lambda, Anda mengonfigurasi fungsi Lambda yang dipicu oleh topik Amazon SNS yang Anda buat di [Buat topik SNS](#). Anda juga dapat menyesuaikan persyaratan memori dan batas waktu untuk fungsi Lambda.

Untuk mengonfigurasi fungsi Lambda

1. Dalam Kode fungsi, ketik `com.amazonaws.lambda.demo.JobCompletionHandler` untuk Handler.
2. Dalam Pengaturan dasar, pilih Edit. Parameter Edit pengaturan dasar akan ditampilkan.
 - a. Pilih 1024 untuk Memori.
 - b. Pilih 10 detik untuk Waktu habis.
 - c. Pilih Simpan.
3. Di bawah Desainer, pilih + Tambahkan pemicu. Tambah dialog pemicu yang ditampilkan.
4. Dalam Konfigurasi pemicu pilih SNS.

Dalam Topik SNS, pilih topik Amazon SNS yang Anda buat di [Buat topik SNS](#).

5. Pilih Aktifkan pemicu.
6. Untuk menambahkan pemicu, pilih Tambahkan.
7. Pilih Simpan untuk menyimpan fungsi Lambda.

Konfigurasi peran IAM Lambda

Untuk memanggil operasi Video Rekognition Amazon, Anda menambahkan kebijakan terkelola AmazonRekognitionFullAccessAWS ke peran Lambda IAM. Mulai operasi, seperti [StartLabelDetection](#), juga memerlukan izin peran lulus untuk peran layanan IAM yang digunakan Amazon Rekognition Video untuk mengakses topik Amazon SNS.

Untuk mengonfigurasi peran

1. Masuk ke AWS Management Console dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi, pilih Peran.
3. Dalam daftar, pilih nama peran eksekusi yang Anda buat di [Buat fungsi Lambda](#).
4. Pilih tab Izin.
5. Pilih Pasang kebijakan.
6. Pilih AmazonRekognitionFullAccessdari daftar kebijakan.
7. Pilih Lampirkan kebijakan.
8. Sekali lagi, pilih peran eksekusi.
9. Pilih Tambahkan kebijakan inline.
10. Pilih tab JSON.
11. Ganti kebijakan yang sudah ada dengan kebijakan berikut. Ganti `servicerole` dengan peran layanan IAM yang Anda buat di [Buat topik SNS](#).

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

12. Pilih Tinjau kebijakan.
13. Pada Nama*, ketik nama kebijakan.

14. Pilih Buat kebijakan.

Buat proyek AWS Toolkit for Eclipse Lambda

Ketika fungsi Lambda dipicu, kode berikut mendapatkan status penyelesaian dari topik Amazon SNS, dan [GetLabelDetection](#) panggilan untuk mendapatkan hasil analisis. Hitungan label yang terdeteksi, dan daftar label yang terdeteksi ditulis ke CloudWatch log. Fungsi Lambda Anda perlu menyimpan hasil analisis video untuk digunakan nanti.

Untuk membuat proyek AWS Toolkit for Eclipse Lambda

1. [Buat proyek AWS Toolkit for EclipseAWS Lambda.](#)

- Untuk Nama proyek:, ketik nama proyek yang Anda pilih.
- Untuk Nama Kelas:, masukkan JobCompletionHandler.
- Untuk tipe input:, pilih Peristiwa SNS.
- Tidak perlu mengubah bidang lain.

2. Di Eclipse Project explorer, buka metode handler Lambda yang dihasilkan (JobCompletionHandler.java) dan ganti isinya dengan yang berikut:

```
//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. Namespace Rekognition belum terselesaikan. Untuk mengoreksi ini:
 - Hentikan sejenak mouse Anda di atas bagian garis `import com.amazonaws.services.rekognition.AmazonRekognition;` yang digaris bawahi.
 - Pilih Perbaiki pengaturan proyek....
 - Pilih versi terbaru dari arsip Amazon Rekognition.
 - Pilih OKE untuk menambahkan arsip ke proyek.
4. Simpan file.
5. Klik kanan di jendela kode Eclipse Anda, pilih AWS Lambda, lalu pilih Fungsi unggah ke AWS Lambda.
6. Pada halaman Pilih Target Fungsi Lambda, pilih Wilayah AWS untuk digunakan.
7. Pilih Pilih fungsi lambda yang sudah ada, dan pilih fungsi Lambda yang Anda buat di [Buat fungsi Lambda](#).
8. Pilih Selanjutnya. Parameter kotak dialog Konfigurasi fungsi akan ditampilkan.
9. Dalam IAM Role Pilih IAM role yang Anda buat di [Buat fungsi Lambda](#).
10. Pilih Selesai, dan fungsi Lambda diunggah ke AWS.

Uji fungsi Lambda

Gunakan AWS CLI perintah berikut untuk menguji fungsi Lambda dengan memulai analisis deteksi label video. Setelah analisis selesai, fungsi Lambda dipicu. Konfirmasikan bahwa analisis berhasil dengan memeriksa CloudWatch log Log.

Untuk menguji fungsi Lambda

1. Unggah file video format MOV atau MPEG-4 ke bucket S3 Anda. Untuk pengujian, unggah video yang panjangnya tidak lebih dari 30 detik.

Untuk petunjuk, lihat [Mengunggah Objek ke Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

2. Jalankan AWS CLI perintah berikut untuk mulai mendeteksi label dalam video.

```
aws rekognition start-label-detection --video
  "S3Object={Bucket="bucketname",Name="videofile"}" \
--notification-channel "SNSTopicArn=TopicARN,RoleArn=RoleARN" \
--region Region
```

Perbarui nilai berikut:

- Ubah *bucketname* dan *videofile* ke nama bucket Amazon S3 dan nama file video yang Anda ingin labelnya dideteksi.
 - Ubah *TopicARN* untuk ARN dari topik Amazon SNS yang Anda buat di [Buat topik SNS](#).
 - Ubah *RoleARN* untuk ARN dari IAM role yang Anda buat di [Buat topik SNS](#).
 - Ubah *Region* ke AWS Wilayah yang Anda gunakan.
3. Perhatikan nilai dari *JobId* dalam respons. Respons tersebut serupa dengan contoh JSON berikut ini.

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

4. Buka konsol <https://console.aws.amazon.com/cloudwatch/>.
5. Ketika analisis selesai, entri log untuk fungsi Lambda muncul di Grup log.
6. Pilih fungsi Lambda untuk melihat aliran log.
7. Pilih aliran log terbaru untuk melihat entri log yang dibuat oleh fungsi Lambda. Jika operasi berhasil, tampilannya mirip dengan output berikut, yang menunjukkan detail operasi pengenalan video, termasuk ID pekerjaan, jenis operasi "StartLabelDetection", dan daftar kategori label yang terdeteksi seperti Botol, Pakaian, Kerumunan, dan Makanan:

Time (UTC +00:00)	Message
2018-02-28	
19:48:01	START RequestId: 47cb1472-1cc0-11e8-860a-4d00aa2ff96b Version: \$LATEST
19:48:02	Rekognition Video Operation:=====
19:48:02	Job id: "9c7c3b1403a375a044c6dbe793d5c78d06014ee16f5efde083ad654b06f6c59a"
19:48:02	Status: "SUCCEEDED"
19:48:02	Job tag : null
19:48:02	Operation : "StartLabelDetection"
19:48:09	Total number of labels : 29
19:48:09	labels : Audience / Badge / Bottle / Clothing / Coat / Crowd / Electric Guitar / Flora / Food / Guitar / Hu
19:48:09	Result: {}
19:48:09	END RequestId: 47cb1472-1cc0-11e8-860a-4d00aa2ff96b
19:48:09	REPORT RequestId: 47cb1472-1cc0-11e8-860a-4d00aa2ff96b Duration: 8036.70 ms Billed Duration:

Nilai dari ID tugas harus sesuai dengan nilai JobId yang Anda catat di langkah 3.

Menggunakan Amazon Rekognition untuk Verifikasi Identitas

Amazon Rekognition memberi pengguna beberapa operasi yang memungkinkan pembuatan sistem verifikasi identitas secara sederhana. Amazon Rekognition memungkinkan pengguna mendeteksi wajah dalam gambar dan kemudian membandingkan wajah yang terdeteksi dengan wajah lain dengan membandingkan data wajah. Data wajah ini disimpan dalam wadah sisi server yang disebut Koleksi. Dengan memanfaatkan operasi deteksi wajah, perbandingan wajah, dan manajemen pengumpulan Amazon Rekognition, Anda dapat membuat aplikasi dengan solusi verifikasi identitas.

Tutorial ini akan menunjukkan dua alur kerja umum untuk pembuatan aplikasi yang membutuhkan verifikasi identitas.

Alur kerja pertama melibatkan pendaftaran pengguna baru dalam koleksi. Alur kerja kedua melibatkan pencarian koleksi yang ada untuk tujuan login pengguna kembali.

Anda akan menggunakan [AWSSDK untuk Python](#) untuk tutorial ini. Anda juga dapat melihat [AWS Contoh SDK Dokumentasi GitHub repo](#) untuk tutorial Python lainnya.

Topik

- [Prasyarat](#)
- [Membuat Koleksi](#)
- [Registrasi Pengguna Baru](#)
- [Login Pengguna yang Ada](#)

Prasyarat

Sebelum Anda memulai tutorial ini, Anda akan perlu menginstal Python dan menyelesaikan langkah-langkah yang diperlukan untuk [mengatur PythonAWSSDK](#). Di luar ini, pastikan bahwa Anda memiliki:

- [Menciptakan sebuahAWSakun dan peran IAM](#)
- [Menginstal SDK Python \(Boto3\)](#)
- [Dikonfigurasi dengan benarAWSkredensi akses](#)
- [Membuat bucket Amazon Simple Storage Service](#) dan mengunggah gambar yang ingin Anda gunakan sebagai ID untuk keperluan Verifikasi Identitas.
- Telah memilih gambar kedua untuk berfungsi sebagai gambar target untuk verifikasi identitas.

Membuat Koleksi

Sebelum Anda dapat mendaftarkan pengguna baru di Koleksi atau mencari Koleksi untuk pengguna, Anda harus memiliki Koleksi untuk bekerja dengan. Amazon Rekognition Collection adalah wadah sisi server yang digunakan untuk menyimpan informasi tentang wajah yang terdeteksi.

Buat Koleksi

Anda akan mulai dengan menulis fungsi yang membuat Collection untuk digunakan oleh aplikasi Anda. Amazon Rekognition menyimpan informasi tentang wajah yang telah terdeteksi dalam wadah sisi server yang disebut Collections. Anda dapat mencari informasi wajah yang disimpan dalam Koleksi untuk wajah yang dikenal. Untuk menyimpan informasi wajah, Anda harus terlebih dahulu membuat Koleksi menggunakan `CreateCollection` operasi.

1. Pilih nama untuk Koleksi yang ingin Anda buat. Dalam kode berikut, ganti nilai `collection_id` dengan nama Koleksi yang ingin Anda buat, dan ganti nilai `region` dengan nama wilayah yang ditentukan dalam kredensi pengguna Anda. Anda dapat menggunakan `Tags` argumen untuk menerapkan tag apa pun yang Anda inginkan ke koleksi, meskipun ini tidak diperlukan. Yang `CreateCollection` operasi akan mengembalikan informasi tentang koleksi yang telah Anda buat, termasuk Arn koleksi. Catat Arn yang Anda terima sebagai hasil dari menjalankan kode.

```
import boto3

def create_collection(collection_id, region):
```

```
client = boto3.client('rekognition', region_name=region)

# Create a collection
print('Creating collection:' + collection_id)
response = client.create_collection(CollectionId=collection_id,
Tags={"SampleKey1":"SampleValue1"})
print('Collection ARN: ' + response['CollectionArn'])
print('Status code: ' + str(response['StatusCode']))
print('Done...')

collection_id = 'collection-id-name'
region = "region-name"
create_collection(collection_id, region)
```

2. Simpan dan jalankan kode. Salin koleksi Arn.

Sekarang Koleksi Rekognition telah dibuat, Anda dapat menyimpan informasi wajah dan pengidentifikasi dalam Koleksi itu. Anda juga dapat membandingkan wajah dengan informasi yang disimpan untuk verifikasi.

Registrasi Pengguna Baru

Anda akan ingin dapat mendaftarkan pengguna baru dan menambahkan info mereka ke Koleksi. Proses mendaftarkan pengguna baru biasanya melibatkan langkah-langkah berikut:

Panggil **DetectFaces** Operasi

Tulis kode untuk memeriksa kualitas gambar wajah melalui **DetectFaces** operasi. Anda akan menggunakan **DetectFaces** operasi untuk menentukan apakah gambar yang diambil oleh kamera cocok untuk diproses oleh **SearchFacesByImage** operasi. Gambar harus berisi hanya satu wajah. Anda akan memberikan file gambar input lokal ke **DetectFaces** operasi dan menerima rincian untuk wajah terdeteksi dalam gambar. Kode contoh berikut menyediakan gambar masukan untuk **DetectFaces** dan kemudian memeriksa untuk melihat apakah hanya satu wajah yang terdeteksi dalam gambar.

1. Dalam contoh kode berikut, ganti `photodengan` nama gambar target di mana Anda ingin mendeteksi wajah. Anda juga harus mengganti nilai `region` dengan nama wilayah yang terkait dengan akun Anda.

```
import boto3
```

```
import json

def detect_faces(target_file, region):

    client=boto3.client('rekognition', region_name=region)

    imageTarget = open(target_file, 'rb')

    response = client.detect_faces(Image={'Bytes': imageTarget.read()},
    Attributes=['ALL'])

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

        print('Here are the other attributes:')
        print(json.dumps(faceDetail, indent=4, sort_keys=True))

        # Access predictions for individual face details and print them
        print("Gender: " + str(faceDetail['Gender']))
        print("Smile: " + str(faceDetail['Smile']))
        print("Eyeglasses: " + str(faceDetail['Eyeglasses']))
        print("Emotions: " + str(faceDetail['Emotions'][0]))

    return len(response['FaceDetails'])

photo = 'photo-name'
region = 'region-name'
face_count=detect_faces(photo, region)
print("Faces detected: " + str(face_count))

if face_count == 1:
    print("Image suitable for use in collection.")
else:
    print("Please submit an image with only one face.")
```

2. Simpan dan jalankan kode persidangan.

Panggil **CompareFaces** Operasi

Aplikasi Anda harus dapat mendaftarkan pengguna baru dalam Koleksi dan mengonfirmasi identitas pengguna yang kembali. Anda akan membuat fungsi yang digunakan untuk mendaftarkan

pengguna baru terlebih dahulu. Anda akan mulai dengan menggunakan `CompareFaces` operasi untuk membandingkan gambar input/target lokal pengguna dan gambar ID/disimpan. Jika ada kecocokan antara wajah yang terdeteksi di kedua gambar, Anda dapat mencari melalui Koleksi untuk melihat apakah pengguna telah terdaftar di dalamnya.

Mulailah dengan menulis fungsi yang membandingkan gambar input dengan gambar ID yang telah Anda simpan di bucket Amazon S3 Anda. Dalam contoh kode berikut, Anda perlu memberikan gambar masukan sendiri, yang harus ditangkap setelah menggunakan beberapa bentuk detektor liveness. Anda juga harus meneruskan nama gambar yang disimpan di bucket Amazon S3 Anda.

1. Ganti nilai `bucket` dengan nama bucket Amazon S3 yang berisi file sumber Anda. Anda juga perlu mengganti nilai `source_file` dengan nama gambar sumber yang Anda gunakan. Ganti nilai `target_file` dengan nama file target yang telah Anda berikan. Ganti nilai `region` dengan nama `region` yang didefinisikan dalam kredensi pengguna Anda.

Anda juga ingin menentukan tingkat kepercayaan minimum dalam pertandingan yang dikembalikan dalam respons, menggunakan `similarityThreshold` argumen. Wajah yang terdeteksi hanya akan dikembalikan di `FaceMatches` array jika kepercayaan berada di atas ambang batas ini. Yang Anda pilih `similarityThreshold` harus mencerminkan sifat kasus penggunaan spesifik Anda. Setiap kasus penggunaan yang melibatkan aplikasi keamanan kritis harus menggunakan 99 sebagai ambang batas yang dipilih.

```
import boto3

def compare_faces(bucket, sourceFile, targetFile, region):
    client = boto3.client('rekognition', region_name=region)

    imageTarget = open(targetFile, 'rb')

    response = client.compare_faces(SimilarityThreshold=99,
                                    SourceImage={'S3Object':
{'Bucket':bucket, 'Name':sourceFile}},
                                    TargetImage={'Bytes': imageTarget.read()})

    for faceMatch in response['FaceMatches']:
        position = faceMatch['Face']['BoundingBox']
        similarity = str(faceMatch['Similarity'])
        print('The face at ' +
              str(position['Left']) + ' ' +
              str(position['Top']) +
              ' matches with ' + similarity + '% confidence')
```

```
imageTarget.close()
return len(response['FaceMatches'])

bucket = 'bucket-name'
source_file = 'source-file-name'
target_file = 'target-file-name'
region = "region-name"
face_matches = compare_faces(bucket, source_file, target_file, region)
print("Face matches: " + str(face_matches))

if str(face_matches) == "1":
    print("Face match found.")
else:
    print("No face match found.")
```

2. Simpan dan jalankan kode persidangan.

Anda akan dikembalikan objek respons yang berisi informasi tentang wajah yang cocok dan tingkat kepercayaan.

Panggil **SearchFacesByImage** Operasi

Jika tingkat kepercayaan `CompareFaces` operasi di atas pilihan Anda `SimilarityThreshold`, Anda akan ingin mencari `Collection` Anda untuk wajah yang mungkin cocok dengan gambar input. Jika kecocokan ditemukan di koleksi Anda, itu berarti pengguna kemungkinan sudah terdaftar di Koleksi dan tidak perlu mendaftarkan pengguna baru di Koleksi Anda. Jika tidak ada kecocokan, Anda dapat mendaftarkan pengguna baru di Koleksi Anda.

1. Mulailah dengan menulis kode yang akan memanggil `SearchFacesByImage` operasi. Operasi akan mengambil file gambar lokal sebagai argumen dan kemudian mencari `Collection` untuk wajah yang cocok dengan wajah terdeteksi terbesar dalam gambar yang disediakan.

Dalam contoh kode berikut, mengubah nilai `collectionId` ke Koleksi yang ingin Anda cari. Ganti nilai `region` dengan nama wilayah yang terkait dengan akun Anda. Anda juga harus mengganti nilai `photo` dengan nama file input Anda. Anda juga akan ingin menentukan ambang kesamaan dengan mengganti nilai `threshold` dengan persentil yang dipilih.

```
import boto3
```



```
collectionId = 'collection-id-name'
region = "region-name"
photo = 'photo-name'
threshold = 99
maxFaces = 1
client = boto3.client('rekognition', region_name=region)

# input image should be local file here, not s3 file
with open(photo, 'rb') as image:
    response = client.search_faces_by_image(CollectionId=collectionId,
        Image={'Bytes': image.read()},
        FaceMatchThreshold=threshold, MaxFaces=maxFaces)

faceMatches = response['FaceMatches']
print(faceMatches)

for match in faceMatches:
    print('FaceId:' + match['Face']['FaceId'])
    print('ImageId:' + match['Face']['ImageId'])
    print('Similarity: ' + "{:.2f}".format(match['Similarity']) + "%")
    print('Confidence: ' + str(match['Face']['Confidence']))
```

2. Simpan dan jalankan kode persidangan. Jika ada kecocokan, itu berarti orang yang dikenali dalam gambar sudah menjadi bagian dari Koleksi dan tidak perlu melanjutkan ke langkah berikutnya. Dalam hal ini, Anda hanya dapat mengizinkan akses pengguna ke aplikasi.

Panggil **IndexFaces** Operasi

Dengan asumsi bahwa tidak ada kecocokan ditemukan dalam Koleksi yang Anda cari, Anda akan ingin menambahkan wajah pengguna ke koleksi Anda. Anda melakukan ini dengan menelepon `IndexFaces` operasi. Saat Anda menelepon `IndexFaces`, Amazon Rekognition mengekstrak fitur wajah wajah yang diidentifikasi dalam gambar masukan Anda, menyimpan data dalam koleksi yang ditentukan.

1. Mulailah dengan menulis kode untuk menelepon `IndexFaces`. Ganti nilai `image` dengan nama file lokal yang ingin Anda gunakan sebagai gambar masukan untuk `IndexFaces` operasi. Anda juga perlu mengganti nilai `photo_name` dengan nama yang diinginkan untuk gambar masukan Anda. Pastikan untuk mengganti nilai `collection_id` dengan ID koleksi yang Anda buat sebelumnya. Selanjutnya, ganti nilai `region` dengan nama wilayah yang terkait dengan akun Anda. Anda juga akan ingin menentukan nilai untuk `MaxFaces` parameter input, yang

mendefinisikan jumlah maksimum wajah dalam gambar yang harus diindeks. Nilai default untuk parameter ini adalah 1.

```
import boto3

def add_faces_to_collection(target_file, photo, collection_id, region):
    client = boto3.client('rekognition', region_name=region)

    imageTarget = open(target_file, 'rb')

    response = client.index_faces(CollectionId=collection_id,
                                  Image={'Bytes': imageTarget.read()},
                                  ExternalImageId=photo,
                                  MaxFaces=1,
                                  QualityFilter="AUTO",
                                  DetectionAttributes=['ALL'])

    print(response)

    print('Results for ' + photo)
    print('Faces indexed:')
    for faceRecord in response['FaceRecords']:
        print('  Face ID: ' + faceRecord['Face']['FaceId'])
        print('  Location: {}'.format(faceRecord['Face']['BoundingBox']))
        print('  Image ID: {}'.format(faceRecord['Face']['ImageId']))
        print('  External Image ID: {}'.format(faceRecord['Face']
        ['ExternalImageId']))
        print('  Confidence: {}'.format(faceRecord['Face']['Confidence']))

    print('Faces not indexed:')
    for unindexedFace in response['UnindexedFaces']:
        print('  Location: {}'.format(unindexedFace['FaceDetail']['BoundingBox']))
        print('  Reasons:')
        for reason in unindexedFace['Reasons']:
            print('    ' + reason)
    return len(response['FaceRecords'])

image = 'image-file-name'
collection_id = 'collection-id-name'
photo_name = 'desired-image-name'
region = "region-name"

indexed_faces_count = add_faces_to_collection(image, photo_name, collection_id,
region)
```

```
print("Faces indexed count: " + str(indexed_faces_count))
```

2. Simpan dan jalankan kode persidangan. Tentukan apakah Anda ingin menyimpan salah satu data yang dikembalikan oleh `indexFaces` operasi, seperti `FaceID` ditugaskan untuk orang dalam gambar. Bagian selanjutnya akan memeriksa cara menyimpan data ini. Salin kembali `FaceId`, `ImageId`, dan `Confidence` nilai sebelum melanjutkan.

Simpan Data Gambar dan FaceID di Amazon S3 dan Amazon DynamoDB

Setelah ID Wajah untuk gambar input diperoleh, data gambar dapat disimpan di Amazon S3, sedangkan data wajah dan URL gambar dapat dimasukkan ke dalam database seperti DynamoDB.

1. Tuliskan kode untuk mengunggah gambar input ke database Amazon S3 Anda. Dalam contoh kode berikut, ganti nilai `bucket` dengan nama bucket yang ingin Anda unggah file, lalu ganti nilai `file_name` dengan nama file lokal yang ingin Anda simpan di bucket Amazon S3 Anda. Berikan nama kunci yang akan mengidentifikasi file di bucket Amazon S3 dengan mengganti nilai `key_name` dengan nama yang ingin Anda berikan file gambar. File yang ingin Anda unggah adalah file yang sama dengan yang didefinisikan dalam contoh kode sebelumnya, yang merupakan file input yang Anda gunakan `indexFaces`. Akhirnya, ganti nilai `region` dengan nama wilayah yang terkait dengan akun Anda.

```
import boto3
import logging
from botocore.exceptions import ClientError

# store local file in S3 bucket
bucket = "bucket-name"
file_name = "file-name"
key_name = "key-name"
region = "region-name"
s3 = boto3.client('s3', region_name=region)
# Upload the file
try:
    response = s3.upload_file(file_name, bucket, key_name)
    print("File upload successful!")
except ClientError as e:
    logging.error(e)
```

2. Simpan dan jalankan sampel kode persidangan untuk mengunggah gambar masukan Anda ke Amazon Amazon S3.

3. Anda juga ingin menyimpan ID Wajah yang dikembalikan ke database. Hal ini dapat dilakukan dengan membuat tabel database DynamoDB dan kemudian meng-upload ID Wajah ke tabel itu. Contoh kode berikut membuat tabel DynamoDB. Perhatikan bahwa Anda hanya perlu menjalankan kode yang membuat tabel ini sekali. Dalam kode berikut, ganti nilai `region` dengan nilai wilayah yang terkait dengan akun Anda. Anda juga perlu mengganti nilai `database_name` dengan nama yang Anda ingin memberikan tabel DynamoDB.

```
import boto3

# Create DynamoDB database with image URL and face data, face ID

def create_dynamodb_table(table_name, region):
    dynamodb = boto3.client("dynamodb", region_name=region)

    table = dynamodb.create_table(
        TableName=table_name,
        KeySchema=[{
            'AttributeName': 'FaceID', 'KeyType': 'HASH' # Partition key
        },],
        AttributeDefinitions=[
            {
                'AttributeName': 'FaceID', 'AttributeType': 'S' }, ],
        ProvisionedThroughput={
            'ReadCapacityUnits': 10, 'WriteCapacityUnits': 10 }
    )
    print(table)
    return table

region = "region-name"
database_name = 'database-name'
dynamodb_table = create_dynamodb_table(database_name, region)
print("Table status:", dynamodb_table)
```

4. Simpan dan jalankan kode persidangan untuk membuat tabel Anda.
5. Setelah membuat tabel, Anda dapat meng-upload kembali `FaceId` untuk itu. Untuk melakukan ini, Anda akan membuat koneksi ke tabel dengan fungsi `Table` dan kemudian menggunakan `put_item` fungsi untuk mengunggah data.

Dalam contoh kode berikut, ganti nilai `bucket` dengan nama bucket yang berisi gambar input yang Anda unggah ke Amazon S3. Anda juga harus mengganti nilai `file_name` dengan nama file input yang Anda unggah ke bucket Amazon S3 Anda dan nilai `key_name` dengan

kunci yang sebelumnya Anda gunakan untuk mengidentifikasi file input. Akhirnya, ganti nilai `region` dengan nama wilayah yang terkait dengan akun Anda. Nilai-nilai ini harus sesuai dengan yang disediakan pada langkah 1.

`AddDBEntry` menyimpan `FaceId`, `ImageId`, dan nilai Keyakinan ditugaskan ke wajah dalam koleksi. Berikan fungsi di bawah ini dengan nilai-nilai yang Anda simpan selama Langkah 2 dari proses `IndexFaces` bagian.

```
import boto3
from pprint import pprint
from decimal import Decimal
import json

# The local file that was stored in S3 bucket
bucket = "s3-bucket-name"
file_name = "file-name"
key_name = "key-name"
region = "region-name"
# Get URL of file
file_url = "https://s3.amazonaws.com/{}/{}".format(bucket, key_name)
print(file_url)

# upload face-id, face info, and image url
def AddDBEntry(file_name, file_url, face_id, image_id, confidence):
    dynamodb = boto3.resource('dynamodb', region_name=region)
    table = dynamodb.Table('FacesDB-4')
    response = table.put_item(
        Item={
            'ExternalImageID': file_name,
            'ImageURL': file_url,
            'FaceID': face_id,
            'ImageID': image_id,
            'Confidence': json.loads(json.dumps(confidence), parse_float=Decimal)
        }
    )
    return response

# Mock values for face ID, image ID, and confidence - replace them with actual
# values from your collection results
dynamodb_resp = AddDBEntry(file_name, file_url, "FACE-ID-HERE",
    "IMAGE-ID-HERE", confidence-here)
print("Database entry successful.")
```

```
pprint(dynamodb_resp, sort_dicts=False)
```

6. Simpan dan jalankan sampel kode persidangan untuk menyimpan data ID Wajah yang dikembalikan dalam tabel.

Login Pengguna yang Ada

Setelah pengguna telah terdaftar dalam Koleksi, mereka dapat diautentikasi setelah mereka kembali dengan menggunakan `SearchFacesByImage` operasi. Anda akan perlu untuk mendapatkan gambar input dan kemudian memeriksa kualitas gambar input menggunakan `DetectFaces`. Ini menentukan apakah gambar yang cocok telah digunakan sebelum menjalankan `SearchFacesByImage` operasi.

Panggil `DetectFaces` Operasi

1. Anda akan menggunakan `DetectFaces` operasi untuk memeriksa kualitas gambar wajah dan menentukan apakah gambar yang diambil oleh kamera cocok untuk diproses oleh `SearchFacesByImage` operasi. Gambar input harus berisi hanya satu wajah. Contoh kode berikut mengambil gambar masukan dan menyediakannya ke `DetectFaces` operasi.

Dalam contoh kode berikut, ganti nilai `photo` dengan nama gambar target lokal dan ganti nilai `region` dengan nama wilayah yang terkait dengan akun Anda.

```
import boto3
import json

def detect_faces(target_file, region):

    client=boto3.client('rekognition', region_name=region)

    imageTarget = open(target_file, 'rb')

    response = client.detect_faces(Image={'Bytes': imageTarget.read()},
    Attributes=['ALL'])

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

    print('Here are the other attributes:')
    print(json.dumps(faceDetail, indent=4, sort_keys=True))
```

```
# Access predictions for individual face details and print them
print("Gender: " + str(faceDetail['Gender']))
print("Smile: " + str(faceDetail['Smile']))
print("Eyeglasses: " + str(faceDetail['Eyeglasses']))
print("Emotions: " + str(faceDetail['Emotions'][0]))

return len(response['FaceDetails'])

photo = 'photo-name'
region = 'region-name'
face_count=detect_faces(photo, region)
print("Faces detected: " + str(face_count))

if face_count == 1:
    print("Image suitable for use in collection.")
else:
    print("Please submit an image with only one face.")
```

2. Simpan dan jalankan kode.

PanggilSearchFacesByImageOperasi

1. Tulis kode untuk membandingkan wajah yang terdeteksi dengan wajah di Koleksi denganSearchFacesByImage. Anda akan menggunakan kode yang ditampilkan di bagian Registrasi Pengguna Baru yang melanjutkan dan memberikan gambar masukan keSearchFacesByImageoperasi.

Dalam contoh kode berikut, mengubah nilaicollectionIdke koleksi yang ingin Anda cari. Anda juga akan mengubah nilaibucketdengan nama bucket Amazon S3 dan nilaifileNameke file gambar di bucket itu. Ganti nilairegiondengan nama wilayah yang terkait dengan akun Anda. Anda juga akan ingin menentukan ambang kesamaan dengan mengganti nilaithresholddengan persentil yang dipilih.

```
import boto3

bucket = 'bucket-name'
collectionId = 'collection-id-name'
region = "region-name"
fileName = 'file-name'
threshold = 70
```

```
maxFaces = 1
client = boto3.client('rekognition', region_name=region)

# input image should be local file here, not s3 file
with open(fileName, 'rb') as image:
    response = client.search_faces_by_image(CollectionId=collectionId,
        Image={'Bytes': image.read()},
        FaceMatchThreshold=threshold, MaxFaces=maxFaces)
```

2. Simpan dan jalankan kode.

Periksa FaceID yang Dikembalikan dan Tingkat Keyakinan

Sekarang Anda dapat memeriksa informasi tentang yang cocokFaceId dengan mencetak elemen respon sepertiFaceId, Kesamaan, dan atribut Keyakinan.

```
faceMatches = response['FaceMatches']
print(faceMatches)

for match in faceMatches:
    print('FaceId:' + match['Face']['FaceId'])
    print('ImageId:' + match['Face']['ImageId'])
    print('Similarity: ' + "{:.2f}".format(match['Similarity']) + "%")
    print('Confidence: ' + str(match['Face']['Confidence']))
```

Mendeteksi Label dalam Gambar Menggunakan Lambda dan Python

AWS Lambda adalah layanan komputasi yang dapat Anda gunakan untuk menjalankan kode tanpa perlu menyediakan atau mengelola server. Anda dapat memanggil operasi API Rekognition dari dalam fungsi Lambda. Petunjuk berikut menunjukkan cara membuat fungsi Lambda dengan Python yang memanggilDetectLabels.

Fungsi Lambda memanggilDetectLabelsdan mengembalikan array label terdeteksi dalam gambar, serta tingkat kepercayaan yang dengannya mereka terdeteksi.

Instruksi termasuk contoh kode Python yang menunjukkan kepada Anda cara memanggil fungsi Lambda dan menyediakannya dengan gambar yang bersumber dari bucket Amazon S3 atau komputer lokal Anda.

Pastikan gambar yang Anda pilih memenuhi batas Rekognition. Lihat [Pedoman dan kuota](#) di Rekognition dan [DetectLabelsReferensi API](#) untuk informasi tentang jenis file gambar dan batas ukuran.

Buat fungsi Lambda (konsol)

Pada langkah ini, Anda membuat fungsi Lambda kosong dan peran eksekusi IAM yang memungkinkan fungsi Lambda Anda memanggil `DetectLabels` operasi. Pada langkah selanjutnya, Anda menambahkan kode sumber dan secara opsional menambahkan layer ke fungsi Lambda.

Jika Anda menggunakan dokumen yang disimpan dalam bucket Amazon S3, langkah ini juga menunjukkan cara memberikan akses ke bucket yang menyimpan dokumen Anda.

Untuk membuat AWS Lambda fungsi (konsol)

1. Masuk ke AWS Management Console dan buka konsol AWS Lambda di <https://console.aws.amazon.com/lambda/>.
2. Pilih Buat fungsi. Untuk informasi lebih lanjut, lihat [Membuat Fungsi Lambda dengan Konsol](#).
3. Pilih opsi berikut:
 - Pilih Penulis dari scratch.
 - Masukkan nilai untuk Nama fungsi.
 - Untuk Runtime, pilih versi terbaru dari Python.
 - Untuk Arsitektur, pilih x86_64.
4. Pilih Buat fungsi untuk membuat AWS Lambda fungsi.
5. Pada halaman fungsi, pilih Konfigurasi tab.
6. Pada izin panel, di bawah Peran eksekusi, pilih nama peran untuk membuka peran di konsol IAM.
7. Di dalam Izin tab, pilih Tambahkan izinkan kemudian Buat kebijakan inline.
8. Pilih JSON tab dan ganti kebijakan dengan kebijakan berikut:

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

```

        "Sid": "DetectLabels"
      }
    ]
  }

```

9. Pilih Tinjau kebijakan.
10. Masukkan nama untuk kebijakan, misalnya DetectLabels-akses.
11. Pilih Buat kebijakan.
12. Jika Anda menyimpan dokumen untuk analisis dalam bucket Amazon S3, Anda harus menambahkan kebijakan akses Amazon S3. Untuk melakukan ini, ulangi langkah 7 hingga 11 di AWS Lambda konsol dan membuat perubahan berikut.
 - a. Untuk langkah 8, gunakan kebijakan berikut. Ganti *jalur ember/folder* dengan bucket Amazon S3 dan jalur folder ke dokumen yang ingin Anda analisis.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3Access",
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::bucket/folder path/*"
    }
  ]
}

```

- b. Untuk langkah 10, pilih nama kebijakan yang berbeda, seperti S3Bucket-akses.

(Opsional) Buat layer (konsol)

Anda tidak perlu melakukan langkah ini untuk menggunakan fungsi dan panggilan `LambdaDetectLabels`.

Yang `DetectLabels` operasi disertakan dalam lingkungan default Lambda Python sebagai bagian dari AWS SDK untuk Python (Boto3).

Jika bagian lain dari fungsi Lambda Anda memerlukan terbaru AWS pembaruan layanan yang tidak ada di lingkungan Lambda Python default, maka Anda dapat melakukan langkah ini untuk menambahkan rilis Boto3 SDK terbaru sebagai lapisan ke fungsi Anda.

Untuk menambahkan SDK sebagai layer, pertama-tama Anda membuat arsip file zip yang berisi Boto3 SDK. Kemudian, Anda membuat layer dan menambahkan arsip file zip ke layer. Untuk informasi lebih lanjut, lihat [Menggunakan layer dengan fungsi Lambda Anda](#).

Untuk membuat dan menambahkan layer (konsol)

1. Buka prompt perintah dan masukkan perintah berikut untuk membuat paket penyebaran dengan versi terbaru AWSSDK.

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

2. Perhatikan nama file zip (boto3-layer.zip), yang Anda gunakan pada langkah 8 dari prosedur ini.
3. Buka konsol AWS Lambda tersebut di <https://console.aws.amazon.com/lambda/>.
4. Di panel navigasi, pilih Layers (Lapisan).
5. Pilih Buat lapisan.
6. Masukkan nilai untuk Nama dan Deskripsi.
7. Untuk Jenis entri kode, pilih Unggah file.zip dan pilihlah Unggah.
8. Di kotak dialog, pilih arsip file zip (boto3-layer.zip) yang Anda buat pada langkah 1 dari prosedur ini.
9. Untuk Runtime yang kompatibel, pilih versi terbaru dari Python.
10. Pilih Buat untuk membuat layer.
11. Pilih ikon menu panel navigasi.
12. Di panel navigasi, pilih Fungsi.
13. Dalam daftar sumber daya, pilih fungsi yang Anda buat sebelumnya [???](#).
14. Pilih Kode tab.
15. Di dalam Lapisan bagian, pilih Tambahkan layer.
16. Pilih Lapisan kustom.
17. Dalam Lapisan kustom, pilih nama layer yang Anda masukkan pada langkah 6.
18. Dalam Versi pilih versi layer, yang seharusnya 1.
19. Pilih Tambahkan.

Tambahkan kode Python (konsol)

Pada langkah ini, Anda menambahkan kode Python Anda ke fungsi Lambda Anda melalui editor kode konsol Lambda. Kode mendeteksi label dalam gambar menggunakan `DetectLabels` operasi. Ia mengembalikan array label terdeteksi dalam gambar, serta tingkat kepercayaan pada label terdeteksi.

Dokumen yang Anda berikan kepada `DetectLabels` operasi dapat ditemukan di bucket Amazon S3 atau komputer lokal.

Untuk menambahkan kode Python (konsol)

1. Arahkan ke `Kode` tab.
2. Di editor kode, ganti kode di `lambda_function.py` dengan kode berikut:

```
import boto3
import logging
from botocore.exceptions import ClientError
import json
import base64

# Instantiate logger
logger = logging.getLogger(__name__)

# connect to the Rekognition client
rekognition = boto3.client('rekognition')

def lambda_handler(event, context):

    try:
        image = None
        if 'S3Bucket' in event and 'S3Object' in event:
            s3 = boto3.resource('s3')
            s3_object = s3.Object(event['S3Bucket'], event['S3Object'])
            image = s3_object.get()['Body'].read()

        elif 'image' in event:
            image_bytes = event['image'].encode('utf-8')
            img_b64decoded = base64.b64decode(image_bytes)
            image = img_b64decoded
```

```
elif image is None:
    raise ValueError('Missing image, check image or bucket path.')

else:
    raise ValueError("Only base 64 encoded image bytes or S3Object are
supported.")

response = rekognition.detect_labels(Image={'Bytes': image})
lambda_response = {
    "statusCode": 200,
    "body": json.dumps(response)
}
labels = [label['Name'] for label in response['Labels']]
print("Labels found:")
print(labels)

except ClientError as client_err:

    error_message = "Couldn't analyze image: " + client_err.response['Error']
['Message']

    lambda_response = {
        'statusCode': 400,
        'body': {
            "Error": client_err.response['Error']['Code'],
            "ErrorMessage": error_message
        }
    }
    logger.error("Error function %s: %s",
                context.invoked_function_arn, error_message)

except ValueError as val_error:

    lambda_response = {
        'statusCode': 400,
        'body': {
            "Error": "ValueError",
            "ErrorMessage": format(val_error)
        }
    }
    logger.error("Error function %s: %s",
                context.invoked_function_arn, format(val_error))
```

```
return lambda_response
```

3. Pilih `Menyebarkan` untuk menyebarkan fungsi Lambda Anda.

Untuk menambahkan kode Python (konsol)

Sekarang setelah Anda membuat fungsi Lambda, Anda dapat memanggilnya untuk mendeteksi label dalam gambar.

Pada langkah ini, Anda menjalankan kode Python di komputer Anda, yang meneruskan gambar lokal atau gambar dalam bucket Amazon S3, ke fungsi Lambda Anda.

Pastikan Anda menjalankan kode yang sama AWS Wilayah di mana Anda membuat fungsi Lambda. Anda dapat melihat AWS Wilayah untuk fungsi Lambda Anda di bilah navigasi halaman detail fungsi di konsol Lambda.

Jika fungsi Lambda mengembalikan kesalahan batas waktu, perpanjang periode batas waktu untuk fungsi Lambda. Untuk informasi lebih lanjut, lihat [Mengkonfigurasi batas waktu fungsi \(konsol\)](#).

Untuk informasi selengkapnya tentang menjalankan fungsi Lambda dari kode Anda, lihat [Meminta Fungsi AWS Lambda](#).

Untuk mencoba fungsi Lambda

1. Jika Anda belum melakukannya, lakukan hal berikut:
 - a. Pastikan bahwa pengguna memiliki `lambda:InvokeFunction` izin. Anda dapat menggunakan kebijakan berikut:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "InvokeLambda",
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "ARN for lambda function"
    }
  ]
}
```

Anda bisa mendapatkan ARN untuk fungsi Lambda Anda dari ikhtisar fungsi di [Konsol Lambda](#).

Untuk menyediakan akses, tambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

Buat set izin. Ikuti instruksi di [Buat set izin](#) di dalam [AWS IAM Identity Center Panduan Pengguna](#).

- Pengguna yang dikelola dalam IAM melalui penyedia identitas:

Buat peran untuk federasi identitas. Ikuti instruksi di [Membuat peran untuk penyedia identitas pihak ketiga \(federasi\)](#) di dalam [Panduan Pengguna IAM](#).

- Pengguna IAM:

- Buat peran yang dapat diasumsikan pengguna Anda. Ikuti instruksi di [Membuat peran untuk pengguna IAM](#) di dalam [Panduan Pengguna IAM](#).

- (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti instruksi di [Menambahkan izin ke pengguna \(konsol\)](#) di dalam [Panduan Pengguna IAM](#).

- b. Instal dan konfigurasi AWS SDK untuk Python. Untuk informasi selengkapnya, lihat [Langkah 2: Siapkan AWS CLI dan AWS SDK](#).

2. Simpan kode berikut ke file bernama `client.py`:

```
import boto3
import json
import base64
import pprint

# Replace with the name of your S3 bucket and image object key
bucket_name = "name of bucket"
object_key = "name of file in s3 bucket"
# If using a local file, supply the file name as the value of image_path below
image_path = ""

# Create session and establish connection to client['
session = boto3.Session(profile_name='developer-role')
s3 = session.client('s3', region_name="us-east-1")
lambda_client = session.client('lambda', region_name="us-east-1")
```

```
# Replace with the name of your Lambda function
function_name = 'RekDetectLabels'

def analyze_image_local(img_path):

    print("Analyzing local image:")

    with open(img_path, 'rb') as image_file:
        image_bytes = image_file.read()
        data = base64.b64encode(image_bytes).decode("utf8")

    lambda_payload = {"image": data}

    # Invoke the Lambda function with the event payload
    response = lambda_client.invoke(
        FunctionName=function_name,
        Payload=(json.dumps(lambda_payload))
    )

    decoded = json.loads(response['Payload'].read().decode())
    pprint.pprint(decoded)

def analyze_image_s3(bucket_name, object_key):

    print("Analyzing image in S3 bucket:")

    # Load the image data from S3 into memory
    response = s3.get_object(Bucket=bucket_name, Key=object_key)
    image_data = response['Body'].read()
    image_data = base64.b64encode(image_data).decode("utf8")

    # Create the Lambda event payload
    event = {
        'S3Bucket': bucket_name,
        'S3Object': object_key,
        'ImageBytes': image_data
    }

    # Invoke the Lambda function with the event payload
    response = lambda_client.invoke(
        FunctionName=function_name,
        InvocationType='RequestResponse',
        Payload=json.dumps(event),
    )
```



```
decoded = json.loads(response['Payload'].read().decode())
pprint.pprint(decoded)

def main(path_to_image, name_s3_bucket, obj_key):

    if str(path_to_image) != "":
        analyze_image_local(path_to_image)
    else:
        analyze_image_s3(name_s3_bucket, obj_key)

if __name__ == "__main__":
    main(image_path, bucket_name, object_key)
```

3. Jalankan kode tersebut. Jika dokumen berada dalam bucket Amazon S3, pastikan bahwa itu adalah bucket yang sama dengan yang Anda tentukan sebelumnya di langkah [12???](#).

Jika berhasil, kode Anda mengembalikan respons JSON sebagian untuk setiap jenis Blok yang terdeteksi dalam dokumen.

Contoh kode untuk Amazon AWS Rekognition menggunakan SDK

Contoh kode berikut menunjukkan cara menggunakan Amazon Rekognition dengan perangkat pengembangan perangkat lunak (AWS SDK). Contoh kode dalam Bab ini dimaksudkan untuk melengkapi contoh kode yang ditemukan di seluruh sisa panduan ini.

Tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks pada skenario terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

Contoh lintas layanan adalah contoh aplikasi yang bekerja di beberapa Layanan AWS.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Memulai

Halo Amazon Rekognition

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon Rekognition.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

Kode untuk file CMake MakeLists C.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)
```

```
# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rekognition)

# Set this project's name.
project("hello_rekognition")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
  libraries for the AWS SDK.
  string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
    "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
  list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
  # Copy relevant AWS SDK for C++ libraries into the current binary directory
  for running and debugging.

  # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
  may need to uncomment this
  # and set the proper subdirectory to the
  executables' location.

  AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
  hello_rekognition.cpp)

target_link_libraries(${PROJECT_NAME}
  ${AWSSDK_LINK_LIBRARIES})
```

Kode untuk file sumber hello_rekognition.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rekognition/RekognitionClient.h>
#include <aws/rekognition/model/ListCollectionsRequest.h>
#include <iostream>

/*
 * A "Hello Rekognition" starter application which initializes an Amazon
 Rekognition client and
 * lists the Amazon Rekognition collections in the current account and region.
 *
 * main function
 *
 * Usage: 'hello_rekognition'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optional: change the log level for debugging.
    // options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Rekognition::RekognitionClient rekognitionClient(clientConfig);
        Aws::Rekognition::Model::ListCollectionsRequest request;
        Aws::Rekognition::Model::ListCollectionsOutcome outcome =
            rekognitionClient.ListCollections(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String>& collectionsIds =
outcome.GetResult().GetCollectionIds();
            if (!collectionsIds.empty()) {
                std::cout << "collectionsIds: " << std::endl;
                for (auto &collectionId : collectionsIds) {
                    std::cout << "- " << collectionId << std::endl;
                }
            } else {
                std::cout << "No collections found" << std::endl;
            }
        }
    }
}
```

```
    } else {
        std::cerr << "Error with ListCollections: " << outcome.GetError()
                << std::endl;
    }
}

Aws::ShutdownAPI(options); // Should only be called once.
return 0;
}
```

- Untuk detail API, lihat [ListCollections](#) di Referensi AWS SDK for C++ API.

Contoh kode

- [Tindakan untuk Amazon AWS Rekognition menggunakan SDK](#)
 - [Gunakan CompareFaces dengan AWS SDK atau CLI](#)
 - [Gunakan CreateCollection dengan AWS SDK atau CLI](#)
 - [Gunakan DeleteCollection dengan AWS SDK atau CLI](#)
 - [Gunakan DeleteFaces dengan AWS SDK atau CLI](#)
 - [Gunakan DescribeCollection dengan AWS SDK atau CLI](#)
 - [Gunakan DetectFaces dengan AWS SDK atau CLI](#)
 - [Gunakan DetectLabels dengan AWS SDK atau CLI](#)
 - [Gunakan DetectModerationLabels dengan AWS SDK atau CLI](#)
 - [Gunakan DetectText dengan AWS SDK atau CLI](#)
 - [Gunakan DisassociateFaces dengan AWS SDK atau CLI](#)
 - [Gunakan GetCelebrityInfo dengan AWS SDK atau CLI](#)
 - [Gunakan IndexFaces dengan AWS SDK atau CLI](#)
 - [Gunakan ListCollections dengan AWS SDK atau CLI](#)
 - [Gunakan ListFaces dengan AWS SDK atau CLI](#)
 - [Gunakan RecognizeCelebrities dengan AWS SDK atau CLI](#)
 - [Gunakan SearchFaces dengan AWS SDK atau CLI](#)
 - [Gunakan SearchFacesByImage dengan AWS SDK atau CLI](#)
- [Skenario untuk Amazon AWS Rekognition menggunakan SDK](#)

- [Buat koleksi Amazon Rekognition dan temukan wajah di dalamnya menggunakan SDK AWS](#)
- [Mendeteksi dan menampilkan elemen dalam gambar dengan Amazon Rekognition menggunakan SDK AWS](#)
- [Mendeteksi informasi dalam video menggunakan Amazon Rekognition dan SDK AWS](#)
- [Contoh lintas layanan untuk Amazon Rekognition menggunakan SDK AWS](#)
 - [Membuat aplikasi manajemen aset foto yang memungkinkan pengguna mengelola foto menggunakan label](#)
 - [Mendeteksi APD dalam gambar dengan Amazon AWS Rekognition menggunakan SDK](#)
 - [Mendeteksi wajah dalam gambar menggunakan AWS SDK](#)
 - [Mendeteksi objek dalam gambar dengan Amazon Rekognition menggunakan SDK AWS](#)
 - [Mendeteksi orang dan objek dalam video dengan Amazon Rekognition menggunakan SDK AWS](#)
 - [Simpan EXIF dan informasi gambar lainnya menggunakan SDK AWS](#)

Tindakan untuk Amazon AWS Rekognition menggunakan SDK

Contoh kode berikut menunjukkan cara melakukan tindakan Amazon Rekognition individual dengan SDK. AWS Kutipan ini menyebut Amazon Rekognition API dan merupakan kutipan kode dari program yang lebih besar yang harus dijalankan dalam konteks. Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan instruksi untuk mengatur dan menjalankan kode.

Contoh berikut hanya mencakup tindakan yang paling umum digunakan. Untuk daftar lengkapnya, lihat Referensi [API Amazon Rekognition](#).

Contoh

- [Gunakan CompareFaces dengan AWS SDK atau CLI](#)
- [Gunakan CreateCollection dengan AWS SDK atau CLI](#)
- [Gunakan DeleteCollection dengan AWS SDK atau CLI](#)
- [Gunakan DeleteFaces dengan AWS SDK atau CLI](#)
- [Gunakan DescribeCollection dengan AWS SDK atau CLI](#)
- [Gunakan DetectFaces dengan AWS SDK atau CLI](#)
- [Gunakan DetectLabels dengan AWS SDK atau CLI](#)
- [Gunakan DetectModerationLabels dengan AWS SDK atau CLI](#)
- [Gunakan DetectText dengan AWS SDK atau CLI](#)

- [Gunakan DisassociateFaces dengan AWS SDK atau CLI](#)
- [Gunakan GetCelebrityInfo dengan AWS SDK atau CLI](#)
- [Gunakan IndexFaces dengan AWS SDK atau CLI](#)
- [Gunakan ListCollections dengan AWS SDK atau CLI](#)
- [Gunakan ListFaces dengan AWS SDK atau CLI](#)
- [Gunakan RecognizeCelebrities dengan AWS SDK atau CLI](#)
- [Gunakan SearchFaces dengan AWS SDK atau CLI](#)
- [Gunakan SearchFacesByImage dengan AWS SDK atau CLI](#)

Gunakan **CompareFaces** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan CompareFaces.

Untuk informasi selengkapnya, lihat [Membandingkan wajah dalam gambar](#).

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

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

/// <summary>
/// Uses the Amazon Rekognition Service to compare faces in two images.
/// </summary>
public class CompareFaces
{
    public static async Task Main()
    {
```

```
float similarityThreshold = 70F;
string sourceImage = "source.jpg";
string targetImage = "target.jpg";

var rekognitionClient = new AmazonRekognitionClient();

Amazon.Rekognition.Model.Image imageSource = new
Amazon.Rekognition.Model.Image();

    try
    {
        using FileStream fs = new FileStream(sourceImage, FileMode.Open,
FileAccess.Read);
        byte[] data = new byte[fs.Length];
        fs.Read(data, 0, (int)fs.Length);
        imageSource.Bytes = new MemoryStream(data);
    }
    catch (Exception)
    {
        Console.WriteLine($"Failed to load source image: {sourceImage}");
        return;
    }

    Amazon.Rekognition.Model.Image imageTarget = new
Amazon.Rekognition.Model.Image();

    try
    {
        using FileStream fs = new FileStream(targetImage, FileMode.Open,
FileAccess.Read);
        byte[] data = new byte[fs.Length];
        data = new byte[fs.Length];
        fs.Read(data, 0, (int)fs.Length);
        imageTarget.Bytes = new MemoryStream(data);
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Failed to load target image: {targetImage}");
        Console.WriteLine(ex.Message);
        return;
    }

var compareFacesRequest = new CompareFacesRequest
{
```



```
        SourceImage = imageSource,
        TargetImage = imageTarget,
        SimilarityThreshold = similarityThreshold,
    };

    // Call operation
    var compareFacesResponse = await
rekognitionClient.CompareFacesAsync(compareFacesRequest);

    // Display results
    compareFacesResponse.FaceMatches.ForEach(match =>
    {
        ComparedFace face = match.Face;
        BoundingBox position = face.BoundingBox;
        Console.WriteLine($"Face at {position.Left} {position.Top}
matches with {match.Similarity}% confidence.");
    });

    Console.WriteLine($"Found {compareFacesResponse.UnmatchedFaces.Count}
face(s) that did not match.");
    }
}
```

- Untuk detail API, lihat [CompareFaces](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk membandingkan wajah dalam dua gambar

`compare-faces` Perintah berikut membandingkan wajah dalam dua gambar yang disimpan dalam bucket Amazon S3.

```
aws rekognition compare-faces \
  --source-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"source.jpg"}}' \
  --target-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"target.jpg"}}'
```

Output:

```
{
  "UnmatchedFaces": [],
  "FaceMatches": [
    {
      "Face": {
        "BoundingBox": {
          "Width": 0.12368916720151901,
          "Top": 0.16007372736930847,
          "Left": 0.5901257991790771,
          "Height": 0.25140416622161865
        },
        "Confidence": 100.0,
        "Pose": {
          "Yaw": -3.7351467609405518,
          "Roll": -0.10309021919965744,
          "Pitch": 0.8637830018997192
        },
        "Quality": {
          "Sharpness": 95.51618957519531,
          "Brightness": 65.29893493652344
        },
        "Landmarks": [
          {
            "Y": 0.26721030473709106,
            "X": 0.6204193830490112,
            "Type": "eyeLeft"
          },
          {
            "Y": 0.26831310987472534,
            "X": 0.6776827573776245,
            "Type": "eyeRight"
          },
          {
            "Y": 0.3514654338359833,
            "X": 0.6241428852081299,
            "Type": "mouthLeft"
          },
          {
            "Y": 0.35258132219314575,
            "X": 0.6713621020317078,
            "Type": "mouthRight"
          }
        ]
      }
    }
  ]
}
```

```
        {
            "Y": 0.3140771687030792,
            "X": 0.6428444981575012,
            "Type": "nose"
        }
    ],
    "Similarity": 100.0
}
},
"SourceImageFace": {
    "BoundingBox": {
        "Width": 0.12368916720151901,
        "Top": 0.16007372736930847,
        "Left": 0.5901257991790771,
        "Height": 0.25140416622161865
    },
    "Confidence": 100.0
}
}
```

Untuk informasi selengkapnya, lihat [Membandingkan Wajah dalam Gambar](#) di Panduan Pengembang Rekognition Amazon.

- Untuk detail API, lihat [CompareFaces](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.CompareFacesRequest;
import software.amazon.awssdk.services.rekognition.model.CompareFacesResponse;
```

```
import software.amazon.awssdk.services.rekognition.model.CompareFacesMatch;
import software.amazon.awssdk.services.rekognition.model.ComparedFace;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <pathSource> <pathTarget>

            Where:
                pathSource - The path to the source image (for example, C:\
\AWS\pic1.png).\s
                pathTarget - The path to the target image (for example, C:\
\AWS\pic2.png).\s
            """;

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

        Float similarityThreshold = 70F;
        String sourceImage = args[0];
        String targetImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();
    }
}
```

```
        compareTwoFaces(rekClient, similarityThreshold, sourceImage,
targetImage);
        rekClient.close();
    }

    public static void compareTwoFaces(RekognitionClient rekClient, Float
similarityThreshold, String sourceImage,
        String targetImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        InputStream tarStream = new FileInputStream(targetImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        SdkBytes targetBytes = SdkBytes.fromInputStream(tarStream);

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

        Image tarImage = Image.builder()
            .bytes(targetBytes)
            .build();

        CompareFacesRequest facesRequest = CompareFacesRequest.builder()
            .sourceImage(souImage)
            .targetImage(tarImage)
            .similarityThreshold(similarityThreshold)
            .build();

        // Compare the two images.
        CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
        List<CompareFacesMatch> faceDetails =
compareFacesResult.faceMatches();
        for (CompareFacesMatch match : faceDetails) {
            ComparedFace face = match.face();
            BoundingBox position = face.boundingBox();
            System.out.println("Face at " + position.left().toString()
                + " " + position.top()
                + " matches with " + face.confidence().toString()
                + "% confidence.");
        }
        List<ComparedFace> uncompered = compareFacesResult.unmatchedFaces();
    }
}
```

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

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println("Failed to load source image " + sourceImage);
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [CompareFaces](#) di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun compareTwoFaces(
    similarityThresholdVal: Float,
    sourceImageVal: String,
    targetImageVal: String,
) {
    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage =
        Image {
            bytes = sourceBytes
        }

    val tarImage =
```

```
Image {
    bytes = targetBytes
}

val facesRequest =
    CompareFacesRequest {
        sourceImage = souImage
        targetImage = tarImage
        similarityThreshold = similarityThresholdVal
    }

RekognitionClient { region = "us-east-1" }.use { rekClient ->

    val compareFacesResult = rekClient.compareFaces(facesRequest)
    val faceDetails = compareFacesResult.faceMatches

    if (faceDetails != null) {
        for (match: CompareFacesMatch in faceDetails) {
            val face = match.face
            val position = face?.boundingBox
            if (position != null) {
                println("Face at ${position.left} ${position.top} matches
with ${face.confidence} % confidence.")
            }
        }
    }

    val uncompered = compareFacesResult.unmatchedFaces
    if (uncompered != null) {
        println("There was ${uncompered.size} face(s) that did not match")
    }

    println("Source image rotation:
${compareFacesResult.sourceImageOrientationCorrection}")
    println("target image rotation:
${compareFacesResult.targetImageOrientationCorrection}")
}
}
```

- Untuk detail API, lihat [CompareFaces](#) di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
            an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client

    def compare_faces(self, target_image, similarity):
        """
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value
        greater
            than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
            reference image. The second element is the list of faces that
        have
            a similarity value below the specified threshold.
```



```
"""
try:
    response = self.rekognition_client.compare_faces(
        SourceImage=self.image,
        TargetImage=target_image.image,
        SimilarityThreshold=similarity,
    )
    matches = [
        RekognitionFace(match["Face"]) for match in
response["FaceMatches"]
    ]
    unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
    logger.info(
        "Found %s matched faces and %s unmatched faces.",
        len(matches),
        len(unmatches),
    )
except ClientError:
    logger.exception(
        "Couldn't match faces from %s to %s.",
        self.image_name,
        target_image.image_name,
    )
    raise
else:
    return matches, unmatches
```

- Untuk detail API, lihat [CompareFaces](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **CreateCollection** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `CreateCollection`.

Untuk informasi selengkapnya, lihat [Membuat koleksi](#).

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses Amazon Rekognition to create a collection to which you can add
/// faces using the IndexFaces operation.
/// </summary>
public class CreateCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine("Creating collection: " + collectionId);

        var createCollectionRequest = new CreateCollectionRequest
        {
            CollectionId = collectionId,
        };

        CreateCollectionResponse createCollectionResponse = await
rekognitionClient.CreateCollectionAsync(createCollectionRequest);
        Console.WriteLine($"CollectionArn :
{createCollectionResponse.CollectionArn}");
        Console.WriteLine($"Status code :
{createCollectionResponse.StatusCode}");
    }
}
```

- Untuk detail API, lihat [CreateCollection](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk membuat koleksi

`create-collection` Perintah berikut membuat koleksi dengan nama yang ditentukan.

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

Output:

```
{  
  "CollectionArn": "aws:rekognition:us-west-2:123456789012:collection/  
MyCollection",  
  "FaceModelVersion": "4.0",  
  "StatusCode": 200  
}
```

Untuk informasi selengkapnya, lihat [Membuat Koleksi](#) di Panduan Pengembang Rekognition Amazon.

- Untuk detail API, lihat [CreateCollection](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionName>\s

                Where:
                    collectionName - The name of the collection.\s
                """;

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

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

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

```

        CreateCollectionRequest collectionRequest =
CreateCollectionRequest.builder()
            .collectionId(collectionId)
            .build();

        CreateCollectionResponse collectionResponse =
rekClient.createCollection(collectionRequest);
        System.out.println("CollectionArn: " +
collectionResponse.collectionArn());
        System.out.println("Status code: " +
collectionResponse.statusCode().toString());

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

```

- Untuk detail API, lihat [CreateCollection](#) di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```

suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}

```

```
}  
}
```

- Untuk detail API, lihat [CreateCollection](#) di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
class RekognitionCollectionManager:  
    """  
    Encapsulates Amazon Rekognition collection management functions.  
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition  
    API.  
    """  
  
    def __init__(self, rekognition_client):  
        """  
        Initializes the collection manager object.  
  
        :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.rekognition_client = rekognition_client  
  
    def create_collection(self, collection_id):  
        """  
        Creates an empty collection.  
  
        :param collection_id: Text that identifies the collection.  
        :return: The newly created collection.  
        """  
        try:  
            response = self.rekognition_client.create_collection(  
                CollectionId=collection_id
```

```
    )
    response["CollectionId"] = collection_id
    collection = RekognitionCollection(response, self.rekognition_client)
    logger.info("Created collection %s.", collection_id)
except ClientError:
    logger.exception("Couldn't create collection %s.", collection_id)
    raise
else:
    return collection
```

- Untuk detail API, lihat [CreateCollection](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteCollection** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DeleteCollection`.

Untuk informasi selengkapnya, lihat [Menghapus koleksi](#).

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh selengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to delete an existing collection.
```

```
/// </summary>
public class DeleteCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine("Deleting collection: " + collectionId);

        var deleteCollectionRequest = new DeleteCollectionRequest()
        {
            CollectionId = collectionId,
        };

        var deleteCollectionResponse = await
rekognitionClient.DeleteCollectionAsync(deleteCollectionRequest);
        Console.WriteLine($"{collectionId}:
{deleteCollectionResponse.StatusCode}");
    }
}
```

- Untuk detail API, lihat [DeleteCollection](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk menghapus koleksi

`delete-collection`Perintah berikut menghapus koleksi yang ditentukan.

```
aws rekognition delete-collection \
--collection-id MyCollection
```

Output:

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


Untuk informasi selengkapnya, lihat [Menghapus Koleksi di Panduan](#) Pengembang Rekognition Amazon.

- Untuk detail API, lihat [DeleteCollection](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class DeleteCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionId>\s

            Where:
                collectionId - The id of the collection to delete.\s
            """;

        if (args.length != 1) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Deleting collection: " + collectionId);
    deleteMyCollection(rekClient, collectionId);
    rekClient.close();
}

public static void deleteMyCollection(RekognitionClient rekClient, String
collectionId) {
    try {
        DeleteCollectionRequest deleteCollectionRequest =
DeleteCollectionRequest.builder()
            .collectionId(collectionId)
            .build();

        DeleteCollectionResponse deleteCollectionResponse =
rekClient.deleteCollection(deleteCollectionRequest);
        System.out.println(collectionId + ": " +
deleteCollectionResponse.statusCode().toString());

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

- Untuk detail API, lihat [DeleteCollection](#) di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- Untuk detail API, lihat [DeleteCollection](#) di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
```

```
def __init__(self, collection, rekognition_client):
    """
    Initializes a collection object.

    :param collection: Collection data in the format returned by a call to
        create_collection.
    :param rekognition_client: A Boto3 Rekognition client.
    """
    self.collection_id = collection["CollectionId"]
    self.collection_arn, self.face_count, self.created =
self._unpack_collection(
    collection
)
    self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )

    def delete_collection(self):
        """
        Deletes the collection.
        """
        try:

self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
            logger.exception("Couldn't delete collection %s.",
self.collection_id)
```

```
raise
```

- Untuk detail API, lihat [DeleteCollection](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteFaces** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DeleteFaces`.

Untuk informasi selengkapnya, lihat [Menghapus wajah dari koleksi](#).

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh selengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to delete one or more faces from
/// a Rekognition collection.
/// </summary>
public class DeleteFaces
{
    public static async Task Main()
    {
```

```

        string collectionId = "MyCollection";
        var faces = new List<string> { "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxxxx" };

        var rekognitionClient = new AmazonRekognitionClient();

        var deleteFacesRequest = new DeleteFacesRequest()
        {
            CollectionId = collectionId,
            FaceIds = faces,
        };

        DeleteFacesResponse deleteFacesResponse = await
        rekognitionClient.DeleteFacesAsync(deleteFacesRequest);
        deleteFacesResponse.DeletedFaces.ForEach(face =>
        {
            Console.WriteLine($"FaceID: {face}");
        });
    }
}

```

- Untuk detail API, lihat [DeleteFaces](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk menghapus wajah dari koleksi

`delete-faces` Perintah berikut menghapus wajah yang ditentukan dari koleksi.

```

aws rekognition delete-faces \
  --collection-id MyCollection
  --face-ids '["0040279c-0178-436e-b70a-e61b074e96b0"]'

```

Output:

```

{
  "DeletedFaces": [
    "0040279c-0178-436e-b70a-e61b074e96b0"
  ]
}

```

```
]
}
```

Untuk informasi selengkapnya, lihat [Menghapus Wajah dari Koleksi](#) di Panduan Pengembang Rekognition Amazon.

- Untuk detail API, lihat [DeleteFaces](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class DeleteFacesFromCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <faceId>\s

                Where:
                    collectionId - The id of the collection from which faces are
                    deleted.\s
    }
}
```

```
        faceId - The id of the face to delete.\s
        """);

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

    String collectionId = args[0];
    String faceId = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Deleting collection: " + collectionId);
    deleteFacesCollection(rekClient, collectionId, faceId);
    rekClient.close();
}

public static void deleteFacesCollection(RekognitionClient rekClient,
    String collectionId,
    String faceId) {

    try {
        DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
            .collectionId(collectionId)
            .faceIds(faceId)
            .build();

        rekClient.deleteFaces(deleteFacesRequest);
        System.out.println("The face was deleted from the collection.");

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

- Untuk detail API, lihat [DeleteFaces](#) di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
        println("$faceIdVal was deleted from the collection")
    }
}
```

- Untuk detail API, lihat [DeleteFaces](#) di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
class RekognitionCollection:
```

```
"""
Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
around parts of the Boto3 Amazon Rekognition API.
"""

def __init__(self, collection, rekognition_client):
    """
    Initializes a collection object.

    :param collection: Collection data in the format returned by a call to
        create_collection.
    :param rekognition_client: A Boto3 Rekognition client.
    """
    self.collection_id = collection["CollectionId"]
    self.collection_arn, self.face_count, self.created =
self._unpack_collection(
    collection
)
    self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )

    def delete_faces(self, face_ids):
        """
        Deletes faces from the collection.

        :param face_ids: The list of IDs of faces to delete.
        :return: The list of IDs of faces that were deleted.
        """
        try:
```

```
        response = self.rekognition_client.delete_faces(
            CollectionId=self.collection_id, FaceIds=face_ids
        )
        deleted_ids = response["DeletedFaces"]
        logger.info(
            "Deleted %s faces from %s.", len(deleted_ids), self.collection_id
        )
    except ClientError:
        logger.exception("Couldn't delete faces from %s.",
            self.collection_id)
        raise
    else:
        return deleted_ids
```

- Untuk detail API, lihat [DeleteFaces](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DescribeCollection** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DescribeCollection`.

Untuk informasi selengkapnya, lihat [Menjelaskan koleksi](#).

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh selengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
using System;
using System.Threading.Tasks;
```

```
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to describe the contents of a
/// collection.
/// </summary>
public class DescribeCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine($"Describing collection: {collectionId}");

        var describeCollectionRequest = new DescribeCollectionRequest()
        {
            CollectionId = collectionId,
        };

        var describeCollectionResponse = await
rekognitionClient.DescribeCollectionAsync(describeCollectionRequest);
        Console.WriteLine($"Collection ARN:
{describeCollectionResponse.CollectionARN}");
        Console.WriteLine($"Face count:
{describeCollectionResponse.FaceCount}");
        Console.WriteLine($"Face model version:
{describeCollectionResponse.FaceModelVersion}");
        Console.WriteLine($"Created:
{describeCollectionResponse.CreationTimestamp}");
    }
}
```

- Untuk detail API, lihat [DescribeCollection](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk menggambarkan koleksi

`describe-collection` Contoh berikut menampilkan rincian tentang koleksi yang ditentukan.

```
aws rekognition describe-collection \  
  --collection-id MyCollection
```

Output:

```
{  
  "FaceCount": 200,  
  "CreationTimestamp": 1569444828.274,  
  "CollectionARN": "arn:aws:rekognition:us-west-2:123456789012:collection/  
MyCollection",  
  "FaceModelVersion": "4.0"  
}
```

Untuk informasi selengkapnya, lihat [Menjelaskan Koleksi di Panduan](#) Pengembang Rekognition Amazon.

- Untuk detail API, lihat [DescribeCollection](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.rekognition.RekognitionClient;  
import  
  software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;  
import  
  software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;  
import software.amazon.awssdk.services.rekognition.model.RekognitionException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials. */
```

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

            Usage:    <collectionName>

            Where:
                collectionName - The name of the Amazon Rekognition
collection.\s
            """;

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

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    public static void describeColl(RekognitionClient rekClient, String
collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
DescribeCollectionRequest.builder()
                .collectionId(collectionName)
                .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
                .describeCollection(describeCollectionRequest);
            System.out.println("Collection Arn : " +
describeCollectionResponse.collectionARN());
        }
    }
}
```

```
        System.out.println("Created : " +
describeCollectionResponse.creationTimestamp().toString());

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

- Untuk detail API, lihat [DescribeCollection](#) di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

- Untuk detail API, lihat [DescribeCollection](#) di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
            create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
    collection
)
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```



```
        collection.get("FaceCount", 0),
        collection.get("CreationTimestamp"),
    )

def describe_collection(self):
    """
    Gets data about the collection from the Amazon Rekognition service.

    :return: The collection rendered as a dict.
    """
    try:
        response = self.rekognition_client.describe_collection(
            CollectionId=self.collection_id
        )
        # Work around capitalization of Arn vs. ARN
        response["CollectionArn"] = response.get("CollectionARN")
        (
            self.collection_arn,
            self.face_count,
            self.created,
        ) = self._unpack_collection(response)
        logger.info("Got data for collection %s.", self.collection_id)
    except ClientError:
        logger.exception("Couldn't get data for collection %s.",
            self.collection_id)
        raise
    else:
        return self.to_dict()
```

- Untuk detail API, lihat [DescribeCollection](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.


Gunakan **DetectFaces** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DetectFaces`.

Untuk informasi selengkapnya, lihat [Mendeteksi wajah dalam gambar](#).

.NET

AWS SDK for .NET

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DetectFaces
{
    public static async Task Main()
    {
        string photo = "input.jpg";
        string bucket = "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectFacesRequest = new DetectFacesRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },
        },
    },
}
```

```
        // Attributes can be "ALL" or "DEFAULT".
        // "DEFAULT": BoundingBox, Confidence, Landmarks, Pose, and
Quality.
        // "ALL": See https://docs.aws.amazon.com/sdkfornet/v3/apidocs/
items/Rekognition/TFaceDetail.html
        Attributes = new List<string>() { "ALL" },
    };

    try
    {
        DetectFacesResponse detectFacesResponse = await
rekognitionClient.DetectFacesAsync(detectFacesRequest);
        bool hasAll = detectFacesRequest.Attributes.Contains("ALL");
        foreach (FaceDetail face in detectFacesResponse.FaceDetails)
        {
            Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
left={face.BoundingBox.Top} width={face.BoundingBox.Width}
height={face.BoundingBox.Height}");
            Console.WriteLine($"Confidence: {face.Confidence}");
            Console.WriteLine($"Landmarks: {face.Landmarks.Count}");
            Console.WriteLine($"Pose: pitch={face.Pose.Pitch}
roll={face.Pose.Roll} yaw={face.Pose.Yaw}");
            Console.WriteLine($"Brightness:
{face.Quality.Brightness}\tSharpness: {face.Quality.Sharpness}");

            if (hasAll)
            {
                Console.WriteLine($"Estimated age is between
{face.AgeRange.Low} and {face.AgeRange.High} years old.");
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

Menampilkan informasi kotak pembatas untuk semua wajah dalam gambar.

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to display the details of the
/// bounding boxes around the faces detected in an image.
/// </summary>
public class ImageOrientationBoundingBox
{
    public static async Task Main()
    {
        string photo = @"D:\Development\AWS-Examples\Rekognition
\target.jpg"; // "photo.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

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

        int height;
        int width;

        // Used to extract original photo width/height
        using (var imageBitmap = new Bitmap(photo))
        {
```

```
        height = imageBitmap.Height;
        width = imageBitmap.Width;
    }

    Console.WriteLine("Image Information:");
    Console.WriteLine(photo);
    Console.WriteLine("Image Height: " + height);
    Console.WriteLine("Image Width: " + width);

    try
    {
        var detectFacesRequest = new DetectFacesRequest()
        {
            Image = image,
            Attributes = new List<string>() { "ALL" },
        };

        DetectFacesResponse detectFacesResponse = await
rekognitionClient.DetectFacesAsync(detectFacesRequest);
        detectFacesResponse.FaceDetails.ForEach(face =>
        {
            Console.WriteLine("Face:");
            ShowBoundingBoxPositions(
                height,
                width,
                face.BoundingBox,
                detectFacesResponse.OrientationCorrection);

            Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
left={face.BoundingBox.Top} width={face.BoundingBox.Width}
height={face.BoundingBox.Height}");
            Console.WriteLine($"The detected face is estimated to be
between {face.AgeRange.Low} and {face.AgeRange.High} years old.\n");
        });
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}

/// <summary>
/// Display the bounding box information for an image.
/// </summary>
```

```
    /// <param name="imageHeight">The height of the image.</param>
    /// <param name="imageWidth">The width of the image.</param>
    /// <param name="box">The bounding box for a face found within the
image.</param>
    /// <param name="rotation">The rotation of the face's bounding box.</
param>
    public static void ShowBoundingBoxPositions(int imageHeight, int
imageWidth, BoundingBox box, string rotation)
    {
        float left;
        float top;

        if (rotation == null)
        {
            Console.WriteLine("No estimated orientation. Check Exif data.");
            return;
        }

        // Calculate face position based on image orientation.
        switch (rotation)
        {
            case "ROTATE_0":
                left = imageWidth * box.Left;
                top = imageHeight * box.Top;
                break;
            case "ROTATE_90":
                left = imageHeight * (1 - (box.Top + box.Height));
                top = imageWidth * box.Left;
                break;
            case "ROTATE_180":
                left = imageWidth - (imageWidth * (box.Left + box.Width));
                top = imageHeight * (1 - (box.Top + box.Height));
                break;
            case "ROTATE_270":
                left = imageHeight * box.Top;
                top = imageWidth * (1 - box.Left - box.Width);
                break;
            default:
                Console.WriteLine("No estimated orientation information.
Check Exif data.");
                return;
        }

        // Display face location information.
```

```
        Console.WriteLine($"Left: {left}");
        Console.WriteLine($"Top: {top}");
        Console.WriteLine($"Face Width: {imageWidth * box.Width}");
        Console.WriteLine($"Face Height: {imageHeight * box.Height}");
    }
}
```

- Untuk detail API, lihat [DetectFaces](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk mendeteksi wajah dalam gambar

`detect-faces` Perintah berikut mendeteksi wajah dalam gambar tertentu yang disimpan dalam bucket Amazon S3.

```
aws rekognition detect-faces \
  --image '{"S3Object":{"Bucket":"MyImageS3Bucket","Name":"MyFriend.jpg"}}' \
  --attributes "ALL"
```

Output:

```
{
  "FaceDetails": [
    {
      "Confidence": 100.0,
      "Eyeglasses": {
        "Confidence": 98.91107940673828,
        "Value": false
      },
      "Sunglasses": {
        "Confidence": 99.7966537475586,
        "Value": false
      },
      "Gender": {
        "Confidence": 99.56611633300781,
        "Value": "Male"
      },
    },
  ],
}
```

```
"Landmarks": [  
  {  
    "Y": 0.26721030473709106,  
    "X": 0.6204193830490112,  
    "Type": "eyeLeft"  
  },  
  {  
    "Y": 0.26831310987472534,  
    "X": 0.6776827573776245,  
    "Type": "eyeRight"  
  },  
  {  
    "Y": 0.3514654338359833,  
    "X": 0.6241428852081299,  
    "Type": "mouthLeft"  
  },  
  {  
    "Y": 0.35258132219314575,  
    "X": 0.6713621020317078,  
    "Type": "mouthRight"  
  },  
  {  
    "Y": 0.3140771687030792,  
    "X": 0.6428444981575012,  
    "Type": "nose"  
  },  
  {  
    "Y": 0.24662546813488007,  
    "X": 0.6001564860343933,  
    "Type": "leftEyeBrowLeft"  
  },  
  {  
    "Y": 0.24326619505882263,  
    "X": 0.6303644776344299,  
    "Type": "leftEyeBrowRight"  
  },  
  {  
    "Y": 0.23818562924861908,  
    "X": 0.6146903038024902,  
    "Type": "leftEyeBrowUp"  
  },  
  {  
    "Y": 0.24373626708984375,  
    "X": 0.6640064716339111,  
    "Type": "leftEyeBrowDown"  
  }  
]
```



```
    "Type": "rightEyeBrowLeft"
  },
  {
    "Y": 0.24877218902111053,
    "X": 0.7025929093360901,
    "Type": "rightEyeBrowRight"
  },
  {
    "Y": 0.23938551545143127,
    "X": 0.6823262572288513,
    "Type": "rightEyeBrowUp"
  },
  {
    "Y": 0.265746533870697,
    "X": 0.6112898588180542,
    "Type": "leftEyeLeft"
  },
  {
    "Y": 0.2676128149032593,
    "X": 0.6317071914672852,
    "Type": "leftEyeRight"
  },
  {
    "Y": 0.262735515832901,
    "X": 0.6201658248901367,
    "Type": "leftEyeUp"
  },
  {
    "Y": 0.27025148272514343,
    "X": 0.6206279993057251,
    "Type": "leftEyeDown"
  },
  {
    "Y": 0.268223375082016,
    "X": 0.6658390760421753,
    "Type": "rightEyeLeft"
  },
  {
    "Y": 0.2672517001628876,
    "X": 0.687832236289978,
    "Type": "rightEyeRight"
  },
  {
    "Y": 0.26383838057518005,
```

```
        "X": 0.6769183874130249,
        "Type": "rightEyeUp"
    },
    {
        "Y": 0.27138751745224,
        "X": 0.676596462726593,
        "Type": "rightEyeDown"
    },
    {
        "Y": 0.32283174991607666,
        "X": 0.6350004076957703,
        "Type": "noseLeft"
    },
    {
        "Y": 0.3219289481639862,
        "X": 0.6567046642303467,
        "Type": "noseRight"
    },
    {
        "Y": 0.3420318365097046,
        "X": 0.6450609564781189,
        "Type": "mouthUp"
    },
    {
        "Y": 0.3664324879646301,
        "X": 0.6455618143081665,
        "Type": "mouthDown"
    },
    {
        "Y": 0.26721030473709106,
        "X": 0.6204193830490112,
        "Type": "leftPupil"
    },
    {
        "Y": 0.26831310987472534,
        "X": 0.6776827573776245,
        "Type": "rightPupil"
    },
    {
        "Y": 0.26343393325805664,
        "X": 0.5946047306060791,
        "Type": "upperJawlineLeft"
    },
    {
```

```
        "Y": 0.3543180525302887,  
        "X": 0.6044883728027344,  
        "Type": "midJawlineLeft"  
    },  
    {  
        "Y": 0.4084877669811249,  
        "X": 0.6477024555206299,  
        "Type": "chinBottom"  
    },  
    {  
        "Y": 0.3562754988670349,  
        "X": 0.707981526851654,  
        "Type": "midJawlineRight"  
    },  
    {  
        "Y": 0.26580461859703064,  
        "X": 0.7234612107276917,  
        "Type": "upperJawlineRight"  
    }  
],  
"Pose": {  
    "Yaw": -3.7351467609405518,  
    "Roll": -0.10309021919965744,  
    "Pitch": 0.8637830018997192  
},  
"Emotions": [  
    {  
        "Confidence": 8.74203109741211,  
        "Type": "SURPRISED"  
    },  
    {  
        "Confidence": 2.501944065093994,  
        "Type": "ANGRY"  
    },  
    {  
        "Confidence": 0.7378743290901184,  
        "Type": "DISGUSTED"  
    },  
    {  
        "Confidence": 3.5296201705932617,  
        "Type": "HAPPY"  
    },  
    {  
        "Confidence": 1.7162904739379883,
```

```
        "Type": "SAD"
      },
      {
        "Confidence": 9.518536567687988,
        "Type": "CONFUSED"
      },
      {
        "Confidence": 0.45474427938461304,
        "Type": "FEAR"
      },
      {
        "Confidence": 72.79895782470703,
        "Type": "CALM"
      }
    ],
    "AgeRange": {
      "High": 48,
      "Low": 32
    },
    "EyesOpen": {
      "Confidence": 98.93987274169922,
      "Value": true
    },
    "BoundingBox": {
      "Width": 0.12368916720151901,
      "Top": 0.16007372736930847,
      "Left": 0.5901257991790771,
      "Height": 0.25140416622161865
    },
    "Smile": {
      "Confidence": 93.4493179321289,
      "Value": false
    },
    "MouthOpen": {
      "Confidence": 90.53053283691406,
      "Value": false
    },
    "Quality": {
      "Sharpness": 95.51618957519531,
      "Brightness": 65.29893493652344
    },
    "Mustache": {
      "Confidence": 89.85221099853516,
      "Value": false
    }
  }
}
```

```
    },
    "Beard": {
      "Confidence": 86.1991195678711,
      "Value": true
    }
  }
]
```

Untuk informasi selengkapnya, lihat [Mendeteksi Wajah dalam Gambar](#) di Panduan Pengembang Rekognition Amazon.

- Untuk detail API, lihat [DetectFaces](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.DetectFacesRequest;
import software.amazon.awssdk.services.rekognition.model.DetectFacesResponse;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceDetail;
import software.amazon.awssdk.services.rekognition.model.AgeRange;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

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

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

            Usage:    <sourceImage>

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

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

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectFacesinImage(rekClient, sourceImage);
        rekClient.close();
    }

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

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

            DetectFacesRequest facesRequest = DetectFacesRequest.builder()
```

```

        .attributes(Attribute.ALL)
        .image(souImage)
        .build();

    DetectFacesResponse facesResponse =
rekClient.detectFaces(facesRequest);
    List<FaceDetail> faceDetails = facesResponse.faceDetails();
    for (FaceDetail face : faceDetails) {
        AgeRange ageRange = face.ageRange();
        System.out.println("The detected face is estimated to be between
"
            + ageRange.low().toString() + " and " +
ageRange.high().toString()
            + " years old.");

        System.out.println("There is a smile : " +
face.smile().value().toString());
    }

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

```

- Untuk detail API, lihat [DetectFaces](#) di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```

suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
        Image {

```

```

        bytes = (File(sourceImage).readBytes())
    }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between
                ${ageRange?.low} and ${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}

```

- Untuk detail API, lihat [DetectFaces](#) di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```

class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

```



```
:param image: Data that defines the image, either the image bytes or
               an Amazon S3 bucket and object key.
:param image_name: The name of the image.
:param rekognition_client: A Boto3 Rekognition client.
"""
self.image = image
self.image_name = image_name
self.rekognition_client = rekognition_client

def detect_faces(self):
    """
    Detects faces in the image.

    :return: The list of faces found in the image.
    """
    try:
        response = self.rekognition_client.detect_faces(
            Image=self.image, Attributes=["ALL"]
        )
        faces = [RekognitionFace(face) for face in response["FaceDetails"]]
        logger.info("Detected %s faces.", len(faces))
    except ClientError:
        logger.exception("Couldn't detect faces in %s.", self.image_name)
        raise
    else:
        return faces
```

- Untuk detail API, lihat [DetectFaces](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DetectLabels** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DetectLabels`.

Untuk informasi selengkapnya, lihat [Mendeteksi label dalam gambar](#).

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect labels within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DetectLabels
{
    public static async Task Main()
    {
        string photo = "del_river_02092020_01.jpg"; // "input.jpg";
        string bucket = "igsmiths3photos"; // "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

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

```
        try
        {
            DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectLabelsRequest);
            Console.WriteLine("Detected labels for " + photo);
            foreach (Label label in detectLabelsResponse.Labels)
            {
                Console.WriteLine($"Name: {label.Name} Confidence:
{label.Confidence}");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}
```

Mendeteksi label dalam file gambar yang disimpan di komputer Anda.

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

/// <summary>
/// Uses the Amazon Rekognition Service to detect labels within an image
/// stored locally.
/// </summary>
public class DetectLabelsLocalFile
{
    public static async Task Main()
    {
        string photo = "input.jpg";

        var image = new Amazon.Rekognition.Model.Image();
        try
        {
            using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
```

```
        byte[] data = null;
        data = new byte[fs.Length];
        fs.Read(data, 0, (int)fs.Length);
        image.Bytes = new MemoryStream(data);
    }
    catch (Exception)
    {
        Console.WriteLine("Failed to load file " + photo);
        return;
    }

    var rekognitionClient = new AmazonRekognitionClient();


    var detectLabelsRequest = new DetectLabelsRequest
    {
        Image = image,
        MaxLabels = 10,
        MinConfidence = 77F,
    };

    try
    {
        DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectLabelsRequest);
        Console.WriteLine($"Detected labels for {photo}");
        foreach (Label label in detectLabelsResponse.Labels)
        {
            Console.WriteLine($"{label.Name}: {label.Confidence}");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

- Untuk detail API, lihat [DetectLabels](#) di Referensi AWS SDK for .NET API.

C++

SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
#!/ Detect instances of real-world entities within an image by using Amazon
Rekognition
/*!
 \param imageBucket: The Amazon Simple Storage Service (Amazon S3) bucket
containing an image.
 \param imageKey: The Amazon S3 key of an image object.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::Rekognition::detectLabels(const Aws::String &imageBucket,
                                       const Aws::String &imageKey,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::Rekognition::RekognitionClient rekognitionClient(clientConfiguration);

    Aws::Rekognition::Model::DetectLabelsRequest request;
    Aws::Rekognition::Model::S3Object s3object;
    s3object.SetBucket(imageBucket);
    s3object.SetName(imageKey);

    Aws::Rekognition::Model::Image image;
    image.SetS3Object(s3object);

    request.SetImage(image);

    const Aws::Rekognition::Model::DetectLabelsOutcome outcome =
rekognitionClient.DetectLabels(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::Rekognition::Model::Label> &labels =
outcome.GetResult().GetLabels();
        if (labels.empty()) {
```

```

        std::cout << "No labels detected" << std::endl;
    } else {
        for (const Aws::Rekognition::Model::Label &label: labels) {
            std::cout << label.GetName() << ": " << label.GetConfidence() <<
std::endl;
        }
    }
} else {
    std::cerr << "Error while detecting labels: '"
        << outcome.GetError().GetMessage()
        << "'" << std::endl;
}

return outcome.IsSuccess();
}

```

- Untuk detail API, lihat [DetectLabels](#) di Referensi AWS SDK for C++ API.

CLI

AWS CLI

Untuk mendeteksi label dalam gambar

`detect-labels` Contoh berikut mendeteksi adegan dan objek dalam gambar yang disimpan dalam bucket Amazon S3.

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

Output:

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

```
    },
    {
      "Name": "Transportation"
    }
  ],
  "Name": "Automobile"
},
{
  "Instances": [],
  "Confidence": 99.15271759033203,
  "Parents": [
    {
      "Name": "Transportation"
    }
  ],
  "Name": "Vehicle"
},
{
  "Instances": [],
  "Confidence": 99.15271759033203,
  "Parents": [],
  "Name": "Transportation"
},
{
  "Instances": [
    {
      "BoundingBox": {
        "Width": 0.10616336017847061,
        "Top": 0.5039216876029968,
        "Left": 0.0037978808395564556,
        "Height": 0.18528179824352264
      },
      "Confidence": 99.15271759033203
    },
    {
      "BoundingBox": {
        "Width": 0.2429988533258438,
        "Top": 0.5251884460449219,
        "Left": 0.7309805154800415,
        "Height": 0.21577216684818268
      },
      "Confidence": 99.1286392211914
    }
  ],
  "Name": "Transportation"
}
```

```
    "BoundingBox": {
      "Width": 0.14233611524105072,
      "Top": 0.5333095788955688,
      "Left": 0.6494812965393066,
      "Height": 0.15528248250484467
    },
    "Confidence": 98.48368072509766
  },
  {
    "BoundingBox": {
      "Width": 0.11086395382881165,
      "Top": 0.5354844927787781,
      "Left": 0.10355594009160995,
      "Height": 0.10271988064050674
    },
    "Confidence": 96.45606231689453
  },
  {
    "BoundingBox": {
      "Width": 0.06254628300666809,
      "Top": 0.5573825240135193,
      "Left": 0.46083059906959534,
      "Height": 0.053911514580249786
    },
    "Confidence": 93.65448760986328
  },
  {
    "BoundingBox": {
      "Width": 0.10105438530445099,
      "Top": 0.534368634223938,
      "Left": 0.5743985772132874,
      "Height": 0.12226245552301407
    },
    "Confidence": 93.06217193603516
  },
  {
    "BoundingBox": {
      "Width": 0.056389667093753815,
      "Top": 0.5235804319381714,
      "Left": 0.9427769780158997,
      "Height": 0.17163699865341187
    },
    "Confidence": 92.6864013671875
  },
}
```



```
{
  "BoundingBox": {
    "Width": 0.06003860384225845,
    "Top": 0.5441341400146484,
    "Left": 0.22409997880458832,
    "Height": 0.06737709045410156
  },
  "Confidence": 90.4227066040039
},
{
  "BoundingBox": {
    "Width": 0.02848697081208229,
    "Top": 0.5107086896896362,
    "Left": 0,
    "Height": 0.19150497019290924
  },
  "Confidence": 86.65286254882812
},
{
  "BoundingBox": {
    "Width": 0.04067881405353546,
    "Top": 0.5566273927688599,
    "Left": 0.316415935754776,
    "Height": 0.03428703173995018
  },
  "Confidence": 85.36471557617188
},
{
  "BoundingBox": {
    "Width": 0.043411049991846085,
    "Top": 0.5394920110702515,
    "Left": 0.18293385207653046,
    "Height": 0.0893595889210701
  },
  "Confidence": 82.21705627441406
},
{
  "BoundingBox": {
    "Width": 0.031183116137981415,
    "Top": 0.5579366683959961,
    "Left": 0.2853088080883026,
    "Height": 0.03989990055561066
  },
  "Confidence": 81.0157470703125
}
```

```
    },
    {
      "BoundingBox": {
        "Width": 0.031113790348172188,
        "Top": 0.5504819750785828,
        "Left": 0.2580395042896271,
        "Height": 0.056484755128622055
      },
      "Confidence": 56.13441467285156
    },
    {
      "BoundingBox": {
        "Width": 0.08586374670267105,
        "Top": 0.5438792705535889,
        "Left": 0.5128012895584106,
        "Height": 0.08550430089235306
      },
      "Confidence": 52.37760925292969
    }
  ],
  "Confidence": 99.15271759033203,
  "Parents": [
    {
      "Name": "Vehicle"
    },
    {
      "Name": "Transportation"
    }
  ],
  "Name": "Car"
},
{
  "Instances": [],
  "Confidence": 98.9914321899414,
  "Parents": [],
  "Name": "Human"
},
{
  "Instances": [
    {
      "BoundingBox": {
        "Width": 0.19360728561878204,
        "Top": 0.35072067379951477,
        "Left": 0.43734854459762573,
```

```
        "Height": 0.2742200493812561
      },
      "Confidence": 98.9914321899414
    },
    {
      "BoundingBox": {
        "Width": 0.03801717236638069,
        "Top": 0.5010883808135986,
        "Left": 0.9155802130699158,
        "Height": 0.06597328186035156
      },
      "Confidence": 85.02790832519531
    }
  ],
  "Confidence": 98.9914321899414,
  "Parents": [],
  "Name": "Person"
},
{
  "Instances": [],
  "Confidence": 93.24951934814453,
  "Parents": [],
  "Name": "Machine"
},
{
  "Instances": [
    {
      "BoundingBox": {
        "Width": 0.03561960905790329,
        "Top": 0.6468243598937988,
        "Left": 0.7850857377052307,
        "Height": 0.08878646790981293
      },
      "Confidence": 93.24951934814453
    },
    {
      "BoundingBox": {
        "Width": 0.02217046171426773,
        "Top": 0.6149078607559204,
        "Left": 0.04757237061858177,
        "Height": 0.07136218994855881
      },
      "Confidence": 91.5025863647461
    }
  ],
}
```

```
{
  "BoundingBox": {
    "Width": 0.016197510063648224,
    "Top": 0.6274210214614868,
    "Left": 0.6472989320755005,
    "Height": 0.04955997318029404
  },
  "Confidence": 85.14686584472656
},
{
  "BoundingBox": {
    "Width": 0.020207518711686134,
    "Top": 0.6348286867141724,
    "Left": 0.7295016646385193,
    "Height": 0.07059963047504425
  },
  "Confidence": 83.34547424316406
},
{
  "BoundingBox": {
    "Width": 0.020280985161662102,
    "Top": 0.6171894669532776,
    "Left": 0.08744934946298599,
    "Height": 0.05297485366463661
  },
  "Confidence": 79.9981460571289
},
{
  "BoundingBox": {
    "Width": 0.018318990245461464,
    "Top": 0.623889148235321,
    "Left": 0.6836880445480347,
    "Height": 0.06730121374130249
  },
  "Confidence": 78.87144470214844
},
{
  "BoundingBox": {
    "Width": 0.021310249343514442,
    "Top": 0.6167286038398743,
    "Left": 0.004064912907779217,
    "Height": 0.08317798376083374
  },
  "Confidence": 75.89361572265625
}
```

```
    },
    {
      "BoundingBox": {
        "Width": 0.03604431077837944,
        "Top": 0.7030032277107239,
        "Left": 0.9254803657531738,
        "Height": 0.04569442570209503
      },
      "Confidence": 64.402587890625
    },
    {
      "BoundingBox": {
        "Width": 0.009834849275648594,
        "Top": 0.5821820497512817,
        "Left": 0.28094568848609924,
        "Height": 0.01964157074689865
      },
      "Confidence": 62.79907989501953
    },
    {
      "BoundingBox": {
        "Width": 0.01475677452981472,
        "Top": 0.6137543320655823,
        "Left": 0.5950819253921509,
        "Height": 0.039063986390829086
      },
      "Confidence": 59.40483474731445
    }
  ],
  "Confidence": 93.24951934814453,
  "Parents": [
    {
      "Name": "Machine"
    }
  ],
  "Name": "Wheel"
},
{
  "Instances": [],
  "Confidence": 92.61514282226562,
  "Parents": [],
  "Name": "Road"
},
{
```

```
    "Instances": [],
    "Confidence": 92.37877655029297,
    "Parents": [
      {
        "Name": "Person"
      }
    ],
    "Name": "Sport"
  },
  {
    "Instances": [],
    "Confidence": 92.37877655029297,
    "Parents": [
      {
        "Name": "Person"
      }
    ],
    "Name": "Sports"
  },
  {
    "Instances": [
      {
        "BoundingBox": {
          "Width": 0.12326609343290329,
          "Top": 0.6332163214683533,
          "Left": 0.44815489649772644,
          "Height": 0.058117982000112534
        },
        "Confidence": 92.37877655029297
      }
    ],
    "Confidence": 92.37877655029297,
    "Parents": [
      {
        "Name": "Person"
      },
      {
        "Name": "Sport"
      }
    ],
    "Name": "Skateboard"
  },
  {
    "Instances": [],
```

```
    "Confidence": 90.62931060791016,
    "Parents": [
      {
        "Name": "Person"
      }
    ],
    "Name": "Pedestrian"
  },
  {
    "Instances": [],
    "Confidence": 88.81334686279297,
    "Parents": [],
    "Name": "Asphalt"
  },
  {
    "Instances": [],
    "Confidence": 88.81334686279297,
    "Parents": [],
    "Name": "Tarmac"
  },
  {
    "Instances": [],
    "Confidence": 88.23201751708984,
    "Parents": [],
    "Name": "Path"
  },
  {
    "Instances": [],
    "Confidence": 80.26520538330078,
    "Parents": [],
    "Name": "Urban"
  },
  {
    "Instances": [],
    "Confidence": 80.26520538330078,
    "Parents": [
      {
        "Name": "Building"
      },
      {
        "Name": "Urban"
      }
    ],
    "Name": "Town"
  }
]
```

```
  },
  {
    "Instances": [],
    "Confidence": 80.26520538330078,
    "Parents": [],
    "Name": "Building"
  },
  {
    "Instances": [],
    "Confidence": 80.26520538330078,
    "Parents": [
      {
        "Name": "Building"
      },
      {
        "Name": "Urban"
      }
    ],
    "Name": "City"
  },
  {
    "Instances": [],
    "Confidence": 78.37934875488281,
    "Parents": [
      {
        "Name": "Car"
      },
      {
        "Name": "Vehicle"
      },
      {
        "Name": "Transportation"
      }
    ],
    "Name": "Parking Lot"
  },
  {
    "Instances": [],
    "Confidence": 78.37934875488281,
    "Parents": [
      {
        "Name": "Car"
      },
      {
```



```
        "Name": "Vehicle"
      },
      {
        "Name": "Transportation"
      }
    ],
    "Name": "Parking"
  },
  {
    "Instances": [],
    "Confidence": 74.37590026855469,
    "Parents": [
      {
        "Name": "Building"
      },
      {
        "Name": "Urban"
      },
      {
        "Name": "City"
      }
    ],
    "Name": "Downtown"
  },
  {
    "Instances": [],
    "Confidence": 69.84622955322266,
    "Parents": [
      {
        "Name": "Road"
      }
    ],
    "Name": "Intersection"
  },
  {
    "Instances": [],
    "Confidence": 57.68518829345703,
    "Parents": [
      {
        "Name": "Sports Car"
      },
      {
        "Name": "Car"
      }
    ],
  },
```

```
        {
            "Name": "Vehicle"
        },
        {
            "Name": "Transportation"
        }
    ],
    "Name": "Coupe"
},
{
    "Instances": [],
    "Confidence": 57.68518829345703,
    "Parents": [
        {
            "Name": "Car"
        },
        {
            "Name": "Vehicle"
        },
        {
            "Name": "Transportation"
        }
    ],
    "Name": "Sports Car"
},
{
    "Instances": [],
    "Confidence": 56.59492111206055,
    "Parents": [
        {
            "Name": "Path"
        }
    ],
    "Name": "Sidewalk"
},
{
    "Instances": [],
    "Confidence": 56.59492111206055,
    "Parents": [
        {
            "Name": "Path"
        }
    ],
    "Name": "Pavement"
}
```

```
    },
    {
      "Instances": [],
      "Confidence": 55.58770751953125,
      "Parents": [
        {
          "Name": "Building"
        },
        {
          "Name": "Urban"
        }
      ],
      "Name": "Neighborhood"
    }
  ],
  "LabelModelVersion": "2.0"
}
```

Untuk informasi selengkapnya, lihat [Mendeteksi Label dalam Gambar](#) di Panduan Pengembang Rekognition Amazon.

- Untuk detail API, lihat [DetectLabels](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
```

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectLabels {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

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

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

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

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

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

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

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

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

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

- Untuk detail API, lihat [DetectLabels](#) di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun detectImageLabels(sourceImage: String) {
```

```

val souImage =
    Image {
        bytes = (File(sourceImage).readBytes())
    }
val request =
    DetectLabelsRequest {
        image = souImage
        maxLabels = 10
    }

RekognitionClient { region = "us-east-1" }.use { rekClient ->
    val response = rekClient.detectLabels(request)
    response.labels?.forEach { label ->
        println("${label.name} : ${label.confidence}")
    }
}
}

```

- Untuk detail API, lihat [DetectLabels](#) di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```

class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or

```

```
        an Amazon S3 bucket and object key.
:param image_name: The name of the image.
:param rekognition_client: A Boto3 Rekognition client.
"""
self.image = image
self.image_name = image_name
self.rekognition_client = rekognition_client

def detect_labels(self, max_labels):
    """
    Detects labels in the image. Labels are objects and people.

    :param max_labels: The maximum number of labels to return.
    :return: The list of labels detected in the image.
    """
    try:
        response = self.rekognition_client.detect_labels(
            Image=self.image, MaxLabels=max_labels
        )
        labels = [RekognitionLabel(label) for label in response["Labels"]]
        logger.info("Found %s labels in %s.", len(labels), self.image_name)
    except ClientError:
        logger.info("Couldn't detect labels in %s.", self.image_name)
        raise
    else:
        return labels
```

- Untuk detail API, lihat [DetectLabels](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DetectModerationLabels** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DetectModerationLabels`.

Untuk informasi selengkapnya, lihat [Mendeteksi gambar yang tidak pantas](#).

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect unsafe content in a
/// JPEG or PNG format image.
/// </summary>
public class DetectModerationLabels
{
    public static async Task Main(string[] args)
    {
        string photo = "input.jpg";
        string bucket = "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectModerationLabelsRequest = new
DetectModerationLabelsRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },
            MinConfidence = 60F,
        };
    }
}
```



```
        try
        {
            var detectModerationLabelsResponse = await
rekognitionClient.DetectModerationLabelsAsync(detectModerationLabelsRequest);
            Console.WriteLine("Detected labels for " + photo);
            foreach (ModerationLabel label in
detectModerationLabelsResponse.ModerationLabels)
            {
                Console.WriteLine($"Label: {label.Name}");
                Console.WriteLine($"Confidence: {label.Confidence}");
                Console.WriteLine($"Parent: {label.ParentName}");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}
```

- Untuk detail API, lihat [DetectModerationLabel](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk mendeteksi konten yang tidak aman dalam gambar

`detect-moderation-labels`Perintah berikut mendeteksi konten yang tidak aman dalam gambar tertentu yang disimpan dalam bucket Amazon S3.

```
aws rekognition detect-moderation-labels \  
  --image "S3object={Bucket=MyImageS3Bucket,Name=gun.jpg}"
```

Output:

```
{  
  "ModerationModelVersion": "3.0",  
  "ModerationLabels": [  
    {
```

```
        "Confidence": 97.29618072509766,  
        "ParentName": "Violence",  
        "Name": "Weapon Violence"  
    },  
    {  
        "Confidence": 97.29618072509766,  
        "ParentName": "",  
        "Name": "Violence"  
    }  
]  
}
```

Untuk informasi selengkapnya, lihat [Mendeteksi Gambar Tidak Aman di Panduan Pengembang Rekognition Amazon](#).

- Untuk detail API, lihat [DetectModerationLabel](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.core.SdkBytes;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.rekognition.RekognitionClient;  
import software.amazon.awssdk.services.rekognition.model.RekognitionException;  
import software.amazon.awssdk.services.rekognition.model.Image;  
import  
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsRequest;  
import  
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsResponse;  
import software.amazon.awssdk.services.rekognition.model.ModerationLabel;  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.io.InputStream;  
import java.util.List;
```

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

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

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

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

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectModLabels(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectModLabels(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();
        }
    }
}
```

```

        DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
        .image(souImage)
        .minConfidence(60F)
        .build();

        DetectModerationLabelsResponse moderationLabelsResponse = rekClient
        .detectModerationLabels(moderationLabelsRequest);
        List<ModerationLabel> labels =
moderationLabelsResponse.moderationLabels();
        System.out.println("Detected labels for image");
        for (ModerationLabel label : labels) {
            System.out.println("Label: " + label.name()
                + "\n Confidence: " + label.confidence().toString() + "%"
                + "\n Parent:" + label.parentName());
        }
    } catch (RekognitionException | FileNotFoundException e) {
        e.printStackTrace();
        System.exit(1);
    }
}
}
}

```

- Untuk detail API, lihat [DetectModerationLabel](#) di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```

suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }
}

```

```

    }

    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} %
Parent: ${label.parentName}")
        }
    }
}

```

- Untuk detail API, lihat [DetectModerationLabel](#) di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```

class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
            an Amazon S3 bucket and object key.

```

```
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client

    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify
content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image
            )
            labels = [
                RekognitionModerationLabel(label)
                for label in response["ModerationLabels"]
            ]
            logger.info(
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels
```

- Untuk detail API, lihat [DetectModerationLabel](#) dalam AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DetectText** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DetectText`.

Untuk informasi selengkapnya, lihat [Mendeteksi teks dalam gambar](#).

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh selengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect text in an image. The
/// example was created using the AWS SDK for .NET version 3.7 and .NET
/// Core 5.0.
/// </summary>
public class DetectText
{
    public static async Task Main()
    {
        string photo = "Dad_photographer.jpg"; // "input.jpg";
        string bucket = "igsmiths3photos"; // "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectTextRequest = new DetectTextRequest()
        {
            Image = new Image()
```

```
        {
            S3Object = new S3Object()
            {
                Name = photo,
                Bucket = bucket,
            },
        },
    };

    try
    {
        DetectTextResponse detectTextResponse = await
rekognitionClient.DetectTextAsync(detectTextRequest);
        Console.WriteLine($"Detected lines and words for {photo}");
        detectTextResponse.TextDetections.ForEach(text =>
        {
            Console.WriteLine($"Detected: {text.DetectedText}");
            Console.WriteLine($"Confidence: {text.Confidence}");
            Console.WriteLine($"Id : {text.Id}");
            Console.WriteLine($"Parent Id: {text.ParentId}");
            Console.WriteLine($"Type: {text.Type}");
        });
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }
}
```

- Untuk detail API, lihat [DetectText](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk mendeteksi teks dalam gambar

`detect-text` Perintah berikut mendeteksi teks dalam gambar yang ditentukan.

```
aws rekognition detect-text \
```



```
--image '{"S3Object":  
{"Bucket":"MyImageS3Bucket","Name":"ExamplePicture.jpg"}}'
```

Output:

```
{  
  "TextDetections": [  
    {  
      "Geometry": {  
        "BoundingBox": {  
          "Width": 0.24624845385551453,  
          "Top": 0.28288066387176514,  
          "Left": 0.391388863325119,  
          "Height": 0.022687450051307678  
        },  
        "Polygon": [  
          {  
            "Y": 0.28288066387176514,  
            "X": 0.391388863325119  
          },  
          {  
            "Y": 0.2826388478279114,  
            "X": 0.6376373171806335  
          },  
          {  
            "Y": 0.30532628297805786,  
            "X": 0.637677013874054  
          },  
          {  
            "Y": 0.305568128824234,  
            "X": 0.39142853021621704  
          }  
        ]  
      },  
      "Confidence": 94.35709381103516,  
      "DetectedText": "ESTD 1882",  
      "Type": "LINE",  
      "Id": 0  
    },  
    {  
      "Geometry": {  
        "BoundingBox": {  
          "Width": 0.33933889865875244,
```

```
        "Top": 0.32603850960731506,
        "Left": 0.34534579515457153,
        "Height": 0.07126858830451965
    },
    "Polygon": [
        {
            "Y": 0.32603850960731506,
            "X": 0.34534579515457153
        },
        {
            "Y": 0.32633158564567566,
            "X": 0.684684693813324
        },
        {
            "Y": 0.3976001739501953,
            "X": 0.684575080871582
        },
        {
            "Y": 0.3973070979118347,
            "X": 0.345236212015152
        }
    ]
},
"Confidence": 99.95779418945312,
"DetectedText": "BRAINS",
"Type": "LINE",
"Id": 1
},
{
    "Confidence": 97.22098541259766,
    "Geometry": {
        "BoundingBox": {
            "Width": 0.061079490929841995,
            "Top": 0.2843210697174072,
            "Left": 0.391391396522522,
            "Height": 0.021029088646173477
        },
        "Polygon": [
            {
                "Y": 0.2843210697174072,
                "X": 0.391391396522522
            },
            {
                "Y": 0.2828207015991211,
```

```
        "X": 0.4524524509906769
      },
      {
        "Y": 0.3038259446620941,
        "X": 0.4534534513950348
      },
      {
        "Y": 0.30532634258270264,
        "X": 0.3923923969268799
      }
    ]
  },
  "DetectedText": "ESTD",
  "ParentId": 0,
  "Type": "WORD",
  "Id": 2
},
{
  "Confidence": 91.49320983886719,
  "Geometry": {
    "BoundingBox": {
      "Width": 0.07007007300853729,
      "Top": 0.2828207015991211,
      "Left": 0.5675675868988037,
      "Height": 0.02250562608242035
    },
    "Polygon": [
      {
        "Y": 0.2828207015991211,
        "X": 0.5675675868988037
      },
      {
        "Y": 0.2828207015991211,
        "X": 0.6376376152038574
      },
      {
        "Y": 0.30532634258270264,
        "X": 0.6376376152038574
      },
      {
        "Y": 0.30532634258270264,
        "X": 0.5675675868988037
      }
    ]
  }
}
```

```


    },
    "DetectedText": "1882",
    "ParentId": 0,
    "Type": "WORD",
    "Id": 3
  },
  {
    "Confidence": 99.95779418945312,
    "Geometry": {
      "BoundingBox": {
        "Width": 0.33933934569358826,
        "Top": 0.32633158564567566,
        "Left": 0.3453453481197357,
        "Height": 0.07127484679222107
      },
      "Polygon": [
        {
          "Y": 0.32633158564567566,
          "X": 0.3453453481197357
        },
        {
          "Y": 0.32633158564567566,
          "X": 0.684684693813324
        },
        {
          "Y": 0.39759939908981323,
          "X": 0.6836836934089661
        },
        {
          "Y": 0.39684921503067017,
          "X": 0.3453453481197357
        }
      ]
    }
  },
  "DetectedText": "BRAINS",
  "ParentId": 1,
  "Type": "WORD",
  "Id": 4
}
]
}

```

- Untuk detail API, lihat [DetectText](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DetectTextRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectTextResponse;
import software.amazon.awssdk.services.rekognition.model.TextDetection;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <sourceImage>

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

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

    String sourceImage = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    detectTextLabels(rekClient, sourceImage);
    rekClient.close();
}

public static void detectTextLabels(RekognitionClient rekClient, String
sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectTextRequest textRequest = DetectTextRequest.builder()
            .image(souImage)
            .build();

        DetectTextResponse textResponse = rekClient.detectText(textRequest);
        List<TextDetection> textCollection = textResponse.textDetections();
        System.out.println("Detected lines and words");
        for (TextDetection text : textCollection) {
            System.out.println("Detected: " + text.detectedText());
            System.out.println("Confidence: " +
text.confidence().toString());
            System.out.println("Id : " + text.id());
            System.out.println("Parent Id: " + text.parentId());
            System.out.println("Type: " + text.type());
            System.out.println();
        }
    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
    }
}
```

```
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat [DetectText](#) di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id: ${text.parentId}")
            println("Type: ${text.type}")
        }
    }
}
```

- Untuk detail API, lihat [DetectText](#) di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
            an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client

    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in
                response["TextDetections"]]
```



```
        logger.info("Found %s texts in %s.", len(texts), self.image_name)
    except ClientError:
        logger.exception("Couldn't detect text in %s.", self.image_name)
        raise
    else:
        return texts
```

- Untuk detail API, lihat [DetectText](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DisassociateFaces** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DisassociateFaces`.

CLI

AWS CLI

```
aws rekognition disassociate-faces --face-ids list-of-face-ids
--user-id user-id --collection-id collection-name --region region-name
```

- Untuk detail API, lihat [DisassociateFaces](#) di Referensi AWS CLI Perintah.

Python

SDK untuk Python (Boto3)

```
from botocore.exceptions import ClientError
import boto3
import logging

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')
```

```
def disassociate_faces(collection_id, user_id, face_ids):
    """
    Disassociate stored faces within collection to the given user

    :param collection_id: The ID of the collection where user and faces are
    stored.
    :param user_id: The ID of the user that we want to disassociate faces from
    :param face_ids: The list of face IDs to be disassociated from the given user

    :return: response of AssociateFaces API
    """
    logger.info(f'Disassociating faces from user: {user_id}, {face_ids}')
    try:
        response = client.disassociate_faces(
            CollectionId=collection_id,
            UserId=user_id,
            FaceIds=face_ids
        )
        print(f'- disassociated {len(response["DisassociatedFaces"])} faces')
    except ClientError:
        logger.exception("Failed to disassociate faces from the given user")
        raise
    else:
        print(response)
        return response

def main():
    face_ids = ["faceId1", "faceId2"]
    collection_id = "collection-id"
    user_id = "user-id"
    disassociate_faces(collection_id, user_id, face_ids)

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

- Untuk detail API, lihat [DisassociateFaces](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetCelebrityInfo** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetCelebrityInfo`.

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to retrieve information about the
/// celebrity identified by the supplied celebrity Id.
/// </summary>
public class CelebrityInfo
{
    public static async Task Main()
    {
        string celebId = "nnnnnnnn";

        var rekognitionClient = new AmazonRekognitionClient();

        var celebrityInfoRequest = new GetCelebrityInfoRequest
        {
            Id = celebId,
        };

        Console.WriteLine($"Getting information for celebrity: {celebId}");

        var celebrityInfoResponse = await
rekognitionClient.GetCelebrityInfoAsync(celebrityInfoRequest);

        // Display celebrity information.
    }
}
```

```
        Console.WriteLine($"celebrity name: {celebrityInfoResponse.Name}");
        Console.WriteLine("Further information (if available):");
        celebrityInfoResponse.Urls.ForEach(url =>
        {
            Console.WriteLine(url);
        });
    }
}
```

- Untuk detail API, lihat [GetCelebrityInfo](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk mendapatkan informasi tentang selebriti

`get-celebrity-info`Perintah berikut menampilkan informasi tentang selebriti yang ditentukan. `id`Parameter berasal dari panggilan sebelumnya `kerrecognize-celebrities`.

```
aws rekognition get-celebrity-info --id nnnnnnn
```

Output:

```
{
  "Name": "Celeb A",
  "Urls": [
    "www.imdb.com/name/aaaaaaaa"
  ]
}
```

Untuk informasi selengkapnya, lihat [Mendapatkan Informasi Tentang Selebriti](#) di Panduan Pengembang Rekognition Amazon.

- Untuk detail API, lihat [GetCelebrityInfo](#) di Referensi AWS CLI Perintah.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **IndexFaces** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `IndexFaces`.

Untuk informasi selengkapnya, lihat [Menambahkan wajah ke koleksi](#).

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces in an image
/// that has been uploaded to an Amazon Simple Storage Service (Amazon S3)
/// bucket and then adds the information to a collection.
/// </summary>
public class AddFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection2";
        string bucket = "doc-example-bucket";
        string photo = "input.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        var image = new Image
        {
            S3Object = new S3Object
            {
                Bucket = bucket,
```

```

        Name = photo,
    },
};

var indexFacesRequest = new IndexFacesRequest
{
    Image = image,
    CollectionId = collectionId,
    ExternalImageId = photo,
    DetectionAttributes = new List<string>() { "ALL" },
};

IndexFacesResponse indexFacesResponse = await
rekognitionClient.IndexFacesAsync(indexFacesRequest);

Console.WriteLine($"{photo} added");
foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
{
    Console.WriteLine($"Face detected: Faceid is
{faceRecord.Face.FaceId}");
}
}
}

```

- Untuk detail API, lihat [IndexFaces](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk menambahkan wajah ke koleksi

`index-faces` Perintah berikut menambahkan wajah yang ditemukan dalam gambar ke koleksi yang ditentukan.

```

aws rekognition index-faces \
  --image '{"S3Object":{"Bucket":"MyVideoS3Bucket","Name":"MyPicture.jpg"}}' \
  --collection-id MyCollection \
  --max-faces 1 \
  --quality-filter "AUTO" \
  --detection-attributes "ALL" \

```

```
--external-image-id "MyPicture.jpg"
```

Output:

```
{
  "FaceRecords": [
    {
      "FaceDetail": {
        "Confidence": 99.993408203125,
        "Eyeglasses": {
          "Confidence": 99.11750030517578,
          "Value": false
        },
        "Sunglasses": {
          "Confidence": 99.98249053955078,
          "Value": false
        },
        "Gender": {
          "Confidence": 99.92769622802734,
          "Value": "Male"
        },
        "Landmarks": [
          {
            "Y": 0.26750367879867554,
            "X": 0.6202793717384338,
            "Type": "eyeLeft"
          },
          {
            "Y": 0.26642778515815735,
            "X": 0.6787431836128235,
            "Type": "eyeRight"
          },
          {
            "Y": 0.31361380219459534,
            "X": 0.6421601176261902,
            "Type": "nose"
          },
          {
            "Y": 0.3495299220085144,
            "X": 0.6216195225715637,
            "Type": "mouthLeft"
          },
          {

```

```
        "Y": 0.35194727778434753,  
        "X": 0.669899046421051,  
        "Type": "mouthRight"  
    },  
    {  
        "Y": 0.26844894886016846,  
        "X": 0.6210268139839172,  
        "Type": "leftPupil"  
    },  
    {  
        "Y": 0.26707562804222107,  
        "X": 0.6817160844802856,  
        "Type": "rightPupil"  
    },  
    {  
        "Y": 0.24834522604942322,  
        "X": 0.6018546223640442,  
        "Type": "leftEyeBrowLeft"  
    },  
    {  
        "Y": 0.24397172033786774,  
        "X": 0.6172008514404297,  
        "Type": "leftEyeBrowUp"  
    },  
    {  
        "Y": 0.24677404761314392,  
        "X": 0.6339119076728821,  
        "Type": "leftEyeBrowRight"  
    },  
    {  
        "Y": 0.24582654237747192,  
        "X": 0.6619398593902588,  
        "Type": "rightEyeBrowLeft"  
    },  
    {  
        "Y": 0.23973053693771362,  
        "X": 0.6804757118225098,  
        "Type": "rightEyeBrowUp"  
    },  
    {  
        "Y": 0.24441994726657867,  
        "X": 0.6978968977928162,  
        "Type": "rightEyeBrowRight"  
    },  
    },
```



```
{
  "Y": 0.2695908546447754,
  "X": 0.6085202693939209,
  "Type": "leftEyeLeft"
},
{
  "Y": 0.26716896891593933,
  "X": 0.6315826177597046,
  "Type": "leftEyeRight"
},
{
  "Y": 0.26289820671081543,
  "X": 0.6202316880226135,
  "Type": "leftEyeUp"
},
{
  "Y": 0.27123287320137024,
  "X": 0.6205548048019409,
  "Type": "leftEyeDown"
},
{
  "Y": 0.2668408751487732,
  "X": 0.6663622260093689,
  "Type": "rightEyeLeft"
},
{
  "Y": 0.26741549372673035,
  "X": 0.6910083889961243,
  "Type": "rightEyeRight"
},
{
  "Y": 0.2614026665687561,
  "X": 0.6785826086997986,
  "Type": "rightEyeUp"
},
{
  "Y": 0.27075251936912537,
  "X": 0.6789616942405701,
  "Type": "rightEyeDown"
},
{
  "Y": 0.3211299479007721,
  "X": 0.6324167847633362,
  "Type": "noseLeft"
}
```

```
    },
    {
      "Y": 0.32276326417922974,
      "X": 0.6558475494384766,
      "Type": "noseRight"
    },
    {
      "Y": 0.34385165572166443,
      "X": 0.6444970965385437,
      "Type": "mouthUp"
    },
    {
      "Y": 0.3671635091304779,
      "X": 0.6459195017814636,
      "Type": "mouthDown"
    }
  ],
  "Pose": {
    "Yaw": -9.54541015625,
    "Roll": -0.5709401965141296,
    "Pitch": 0.6045494675636292
  },
  "Emotions": [
    {
      "Confidence": 39.90074157714844,
      "Type": "HAPPY"
    },
    {
      "Confidence": 23.38753890991211,
      "Type": "CALM"
    },
    {
      "Confidence": 5.840933322906494,
      "Type": "CONFUSED"
    }
  ],
  "AgeRange": {
    "High": 63,
    "Low": 45
  },
  "EyesOpen": {
    "Confidence": 99.80887603759766,
    "Value": true
  },
}
```

```
    "BoundingBox": {
      "Width": 0.18562500178813934,
      "Top": 0.1618015021085739,
      "Left": 0.5575000047683716,
      "Height": 0.24770642817020416
    },
    "Smile": {
      "Confidence": 99.69740295410156,
      "Value": false
    },
    "MouthOpen": {
      "Confidence": 99.97393798828125,
      "Value": false
    },
    "Quality": {
      "Sharpness": 95.54405975341797,
      "Brightness": 63.867706298828125
    },
    "Mustache": {
      "Confidence": 97.05007934570312,
      "Value": false
    },
    "Beard": {
      "Confidence": 87.34505462646484,
      "Value": false
    }
  },
  "Face": {
    "BoundingBox": {
      "Width": 0.18562500178813934,
      "Top": 0.1618015021085739,
      "Left": 0.5575000047683716,
      "Height": 0.24770642817020416
    },
    "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
    "ExternalImageId": "example-image.jpg",
    "Confidence": 99.993408203125,
    "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
  }
},
"UnindexedFaces": [],
"FaceModelVersion": "3.0",
"OrientationCorrection": "ROTATE_0"
```

```
}
```

Untuk informasi selengkapnya, lihat [Menambahkan Wajah ke Koleksi](#) di Panduan Pengembang Rekognition Amazon.

- Untuk detail API, lihat [IndexFaces](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.IndexFacesResponse;
import software.amazon.awssdk.services.rekognition.model.IndexFacesRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.QualityFilter;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceRecord;
import software.amazon.awssdk.services.rekognition.model.UnindexedFace;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Reason;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

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

```
*/
public class AddFacesToCollection {
    public static void main(String[] args) {

        final String usage = ""

            Usage:      <collectionId> <sourceImage>

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

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

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        addToCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }

    public static void addToCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                .collectionId(collectionId)
                .image(souImage)
                .maxFaces(1)
                .qualityFilter(QualityFilter.AUTO)
        }
    }
}
```

```
        .detectionAttributes(Attribute.DEFAULT)
        .build();

    IndexFacesResponse facesResponse =
rekClient.indexFaces(facesRequest);
    System.out.println("Results for the image");
    System.out.println("\n Faces indexed:");
    List<FaceRecord> faceRecords = facesResponse.faceRecords();
    for (FaceRecord faceRecord : faceRecords) {
        System.out.println(" Face ID: " + faceRecord.face().faceId());
        System.out.println(" Location:" +
faceRecord.faceDetail().boundingBox().toString());
    }


    List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
    System.out.println("Faces not indexed:");
    for (UnindexedFace unindexedFace : unindexedFaces) {
        System.out.println(" Location:" +
unindexedFace.faceDetail().boundingBox().toString());
        System.out.println(" Reasons:");
        for (Reason reason : unindexedFace.reasons()) {
            System.out.println("Reason: " + reason);
        }
    }

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

- Untuk detail API, lihat [IndexFaces](#) di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun addToCollection(
    collectionIdVal: String?,
    sourceImage: String,
) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        IndexFacesRequest {
            collectionId = collectionIdVal
            image = souImage
            maxFaces = 1
            qualityFilter = QualityFilter.Auto
            detectionAttributes = listOf(Attribute.Default)
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)

        // Display the results.
        println("Results for the image")
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
            println("Face ID: ${faceRecord.face?.faceId}")
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
        }
    }
}
```

```

        println("Reasons:")

        unindexedFace.reasons?.forEach { reason ->
            println("Reason: $reason")
        }
    }
}
}
}

```

- Untuk detail API, lihat [IndexFaces](#) di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```

class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
            create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
    collection
)
        self.rekognition_client = rekognition_client

```



```
@staticmethod
def _unpack_collection(collection):
    """
    Unpacks optional parts of a collection that can be returned by
    describe_collection.

    :param collection: The collection data.
    :return: A tuple of the data in the collection.
    """
    return (
        collection.get("CollectionArn"),
        collection.get("FaceCount", 0),
        collection.get("CreationTimestamp"),
    )

def index_faces(self, image, max_faces):
    """
    Finds faces in the specified image, indexes them, and stores them in the
    collection.

    :param image: The image to index.
    :param max_faces: The maximum number of faces to index.
    :return: A tuple. The first element is a list of indexed faces.
             The second element is a list of faces that couldn't be indexed.
    """
    try:
        response = self.rekognition_client.index_faces(
            CollectionId=self.collection_id,
            Image=image.image,
            ExternalImageId=image.image_name,
            MaxFaces=max_faces,
            DetectionAttributes=["ALL"],
        )
        indexed_faces = [
            RekognitionFace(**face["Face"], **face["FaceDetail"])
            for face in response["FaceRecords"]
        ]
        unindexed_faces = [
            RekognitionFace(face["FaceDetail"])
            for face in response["UnindexedFaces"]
        ]
        logger.info(
```

```
        "Indexed %s faces in %s. Could not index %s faces.",
        len(indexed_faces),
        image.image_name,
        len(unindexed_faces),
    )
except ClientError:
    logger.exception("Couldn't index faces in image %s.",
image.image_name)
    raise
else:
    return indexed_faces, unindexed_faces
```

- Untuk detail API, lihat [IndexFaces](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **ListCollections** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `ListCollections`.

Untuk informasi selengkapnya, lihat [Daftar koleksi](#).

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh selengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;
```

```
/// <summary>
/// Uses Amazon Rekognition to list the collection IDs in the
/// current account.
/// </summary>
public class ListCollections
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        Console.WriteLine("Listing collections");
        int limit = 10;

        var listCollectionsRequest = new ListCollectionsRequest
        {
            MaxResults = limit,
        };

        var listCollectionsResponse = new ListCollectionsResponse();

        do
        {
            if (listCollectionsResponse is not null)
            {
                listCollectionsRequest.NextToken =
listCollectionsResponse.NextToken;
            }

            listCollectionsResponse = await
rekognitionClient.ListCollectionsAsync(listCollectionsRequest);

            listCollectionsResponse.CollectionIds.ForEach(id =>
            {
                Console.WriteLine(id);
            });
        }
        while (listCollectionsResponse.NextToken is not null);
    }
}
```

- Untuk detail API, lihat [ListCollections](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk daftar koleksi yang tersedia

`list-collections` Perintah berikut mencantumkan koleksi yang tersedia di AWS akun.

```
aws rekognition list-collections
```

Output:

```
{
  "FaceModelVersions": [
    "2.0",
    "3.0",
    "3.0",
    "3.0",
    "4.0",
    "1.0",
    "3.0",
    "4.0",
    "4.0",
    "4.0"
  ],
  "CollectionIds": [
    "MyCollection1",
    "MyCollection2",
    "MyCollection3",
    "MyCollection4",
    "MyCollection5",
    "MyCollection6",
    "MyCollection7",
    "MyCollection8",
    "MyCollection9",
    "MyCollection10"
  ]
}
```

Untuk informasi selengkapnya, lihat [Daftar Koleksi](#) di Panduan Pengembang Rekognition Amazon.

- Untuk detail API, lihat [ListCollections](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class ListCollections {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Listing collections");
        listAllCollections(rekClient);
        rekClient.close();
    }

    public static void listAllCollections(RekognitionClient rekClient) {
        try {
            ListCollectionsRequest listCollectionsRequest =
                ListCollectionsRequest.builder()

```

```
        .maxResults(10)
        .build();

        ListCollectionsResponse response =
rekClient.listCollections(listCollectionsRequest);
        List<String> collectionIds = response.collectionIds();
        for (String resultId : collectionIds) {
            System.out.println(resultId);
        }

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

- Untuk detail API, lihat [ListCollections](#) di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
            maxResults = 10
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

```
}
```

- Untuk detail API, lihat [ListCollections](#) di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
    API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client

    def list_collections(self, max_results):
        """
        Lists collections for the current account.

        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
            response =
self.rekognition_client.list_collections(MaxResults=max_results)
            collections = [
```

```
        RekognitionCollection({"CollectionId": col_id},
self.rekognition_client)
        for col_id in response["CollectionIds"]
    ]
    except ClientError:
        logger.exception("Couldn't list collections.")
        raise
    else:
        return collections
```

- Untuk detail API, lihat [ListCollections](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **ListFaces** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `ListFaces`.

Untuk informasi selengkapnya, lihat [Daftar wajah dalam koleksi](#).

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh selengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;
```



```
/// <summary>
/// Uses the Amazon Rekognition Service to retrieve the list of faces
/// stored in a collection.
/// </summary>
public class ListFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection2";

        var rekognitionClient = new AmazonRekognitionClient();

        var listFacesResponse = new ListFacesResponse();
        Console.WriteLine($"Faces in collection {collectionId}");

        var listFacesRequest = new ListFacesRequest
        {
            CollectionId = collectionId,
            MaxResults = 1,
        };

        do
        {
            listFacesResponse = await
rekognitionClient.ListFacesAsync(listFacesRequest);
            listFacesResponse.Faces.ForEach(face =>
            {
                Console.WriteLine(face.FaceId);
            });

            listFacesRequest.NextToken = listFacesResponse.NextToken;
        }
        while (!string.IsNullOrEmpty(listFacesResponse.NextToken));
    }
}
```

- Untuk detail API, lihat [ListFaces](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk membuat daftar wajah dalam koleksi

`list-faces` Perintah berikut mencantumkan wajah-wajah dalam koleksi yang ditentukan.

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

Output:

```
{  
  "FaceModelVersion": "3.0",  
  "Faces": [  
    {  
      "BoundingBox": {  
        "Width": 0.5216310024261475,  
        "Top": 0.3256250023841858,  
        "Left": 0.13394300639629364,  
        "Height": 0.3918749988079071  
      },  
      "FaceId": "0040279c-0178-436e-b70a-e61b074e96b0",  
      "ExternalImageId": "image1.jpg",  
      "Confidence": 100.0,  
      "ImageId": "f976e487-3719-5e2d-be8b-ea2724c26991"  
    },  
    {  
      "BoundingBox": {  
        "Width": 0.5074880123138428,  
        "Top": 0.3774999976158142,  
        "Left": 0.18302799761295319,  
        "Height": 0.3812499940395355  
      },  
      "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",  
      "ExternalImageId": "image2.jpg",  
      "Confidence": 99.99930572509766,  
      "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"  
    },  
    {  
      "BoundingBox": {  
        "Width": 0.5574039816856384,
```

```
        "Top": 0.37187498807907104,
        "Left": 0.14559100568294525,
        "Height": 0.4181250035762787
    },
    "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
    "ExternalImageId": "image3.jpg",
    "Confidence": 99.99960327148438,
    "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
},
{
    "BoundingBox": {
        "Width": 0.18562500178813934,
        "Top": 0.1618019938468933,
        "Left": 0.5575000047683716,
        "Height": 0.24770599603652954
    },
    "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
    "ExternalImageId": "image4.jpg",
    "Confidence": 99.99340057373047,
    "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
},
{
    "BoundingBox": {
        "Width": 0.5307819843292236,
        "Top": 0.2862499952316284,
        "Left": 0.1564060002565384,
        "Height": 0.3987500071525574
    },
    "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
    "ExternalImageId": "image5.jpg",
    "Confidence": 99.99970245361328,
    "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
},
{
    "BoundingBox": {
        "Width": 0.5773710012435913,
        "Top": 0.34437501430511475,
        "Left": 0.12396000325679779,
        "Height": 0.4337500035762787
    },
    "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
    "ExternalImageId": "image6.jpg",
    "Confidence": 100.0,
    "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
```

```
  },
  {
    "BoundingBox": {
      "Width": 0.5349419713020325,
      "Top": 0.29124999046325684,
      "Left": 0.16389399766921997,
      "Height": 0.40187498927116394
    },
    "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
    "ExternalImageId": "image7.jpg",
    "Confidence": 99.99979400634766,
    "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
  },
  {
    "BoundingBox": {
      "Width": 0.41499999165534973,
      "Top": 0.09187500178813934,
      "Left": 0.28083300590515137,
      "Height": 0.3112500011920929
    },
    "FaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
    "ExternalImageId": "image8.jpg",
    "Confidence": 99.99769592285156,
    "ImageId": "a294da46-2cb1-5cc4-9045-61d7ca567662"
  },
  {
    "BoundingBox": {
      "Width": 0.48166701197624207,
      "Top": 0.20999999344348907,
      "Left": 0.21250000596046448,
      "Height": 0.36125001311302185
    },
    "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
    "ExternalImageId": "image9.jpg",
    "Confidence": 99.99949645996094,
    "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
  },
  {
    "BoundingBox": {
      "Width": 0.18562500178813934,
      "Top": 0.1618019938468933,
      "Left": 0.5575000047683716,
      "Height": 0.24770599603652954
    },
  },
```

```
        "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
        "ExternalImageId": "image10.jpg",
        "Confidence": 99.99340057373047,
        "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
    }
]
}
```

Untuk informasi selengkapnya, lihat [Daftar Wajah dalam Koleksi](#) di Panduan Pengembang Rekognition Amazon.

- Untuk detail API, lihat [ListFaces](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

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

```
final String usage = ""

        Usage:    <collectionId>

        Where:
            collectionId - The name of the collection.\s
        """;

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

String collectionId = args[0];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

System.out.println("Faces in collection " + collectionId);
listFacesCollection(rekClient, collectionId);
rekClient.close();
}

public static void listFacesCollection(RekognitionClient rekClient, String
collectionId) {
    try {
        ListFacesRequest facesRequest = ListFacesRequest.builder()
            .collectionId(collectionId)
            .maxResults(10)
            .build();

        ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
        List<Face> faces = facesResponse.faces();
        for (Face face : faces) {
            System.out.println("Confidence level there is a face: " +
face.confidence());
            System.out.println("The face Id value is " + face.faceId());
        }

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

```
}  
}
```

- Untuk detail API, lihat [ListFaces](#) di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
suspend fun listFacesCollection(collectionIdVal: String?) {  
    val request =  
        ListFacesRequest {  
            collectionId = collectionIdVal  
            maxResults = 10  
        }  
  
    RekognitionClient { region = "us-east-1" }.use { rekClient ->  
        val response = rekClient.listFaces(request)  
        response.faces?.forEach { face ->  
            println("Confidence level there is a face: ${face.confidence}")  
            println("The face Id value is ${face.faceId}")  
        }  
    }  
}
```

- Untuk detail API, lihat [ListFaces](#) di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
            create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
    collection
)
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```



```
        collection.get("FaceCount", 0),
        collection.get("CreationTimestamp"),
    )

def list_faces(self, max_results):
    """
    Lists the faces currently indexed in the collection.

    :param max_results: The maximum number of faces to return.
    :return: The list of faces in the collection.
    """
    try:
        response = self.rekognition_client.list_faces(
            CollectionId=self.collection_id, MaxResults=max_results
        )
        faces = [RekognitionFace(face) for face in response["Faces"]]
        logger.info(
            "Found %s faces in collection %s.", len(faces),
self.collection_id
        )
    except ClientError:
        logger.exception(
            "Couldn't list faces in collection %s.", self.collection_id
        )
        raise
    else:
        return faces
```

- Untuk detail API, lihat [ListFaces](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **RecognizeCelebrities** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `RecognizeCelebrities`.

Untuk informasi selengkapnya, lihat [Mengenal selebriti dalam sebuah gambar](#).

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

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

/// <summary>
/// Shows how to use Amazon Rekognition to identify celebrities in a photo.
/// </summary>
public class CelebritiesInImage
{
    public static async Task Main(string[] args)
    {
        string photo = "moviestars.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        var recognizeCelebritiesRequest = new RecognizeCelebritiesRequest();

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

```
    }

    img.Bytes = new MemoryStream(data);
    recognizeCelebritiesRequest.Image = img;

    Console.WriteLine($"Looking for celebrities in image {photo}\n");

    var recognizeCelebritiesResponse = await
rekognitionClient.RecognizeCelebritiesAsync(recognizeCelebritiesRequest);

Console.WriteLine($"{recognizeCelebritiesResponse.CelebrityFaces.Count}
celebrity(s) were recognized.\n");
    recognizeCelebritiesResponse.CelebrityFaces.ForEach(celeb =>
    {
        Console.WriteLine($"Celebrity recognized: {celeb.Name}");
        Console.WriteLine($"Celebrity ID: {celeb.Id}");
        BoundingBox boundingBox = celeb.Face.BoundingBox;
        Console.WriteLine($"position: {boundingBox.Left}
{boundingBox.Top}");
        Console.WriteLine("Further information (if available):");
        celeb.UrlsWithMetadata.ForEach(url =>
        {
            Console.WriteLine(url);
        });
    });

Console.WriteLine($"{recognizeCelebritiesResponse.UnrecognizedFaces.Count}
face(s) were unrecognized.");
    }
}
```

- Untuk detail API, lihat [RecognizeCelebrities](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk mengenali selebriti dalam sebuah gambar

recognize-celebrities Perintah berikut mengenali selebriti dalam gambar tertentu yang disimpan dalam bucket Amazon S3. :

```
aws rekognition recognize-celebrities \  
  --image "S3object={Bucket=MyImageS3Bucket,Name=moviestars.jpg}"
```

Output:

```
{  
  "UnrecognizedFaces": [  
    {  
      "BoundingBox": {  
        "Width": 0.14416666328907013,  
        "Top": 0.077777778059244156,  
        "Left": 0.625,  
        "Height": 0.2746031880378723  
      },  
      "Confidence": 99.9990234375,  
      "Pose": {  
        "Yaw": 10.80408763885498,  
        "Roll": -12.761146545410156,  
        "Pitch": 10.96889877319336  
      },  
      "Quality": {  
        "Sharpness": 94.1185531616211,  
        "Brightness": 79.18367004394531  
      },  
      "Landmarks": [  
        {  
          "Y": 0.18220913410186768,  
          "X": 0.6702951788902283,  
          "Type": "eyeLeft"  
        },  
        {  
          "Y": 0.16337193548679352,  
          "X": 0.7188183665275574,  
          "Type": "eyeRight"  
        },  
        {  
          "Y": 0.20739148557186127,  
          "X": 0.7055801749229431,  
          "Type": "nose"  
        }  
      ],  
    }  
  ],  
}
```

```
        {
            "Y": 0.2889308035373688,
            "X": 0.687512218952179,
            "Type": "mouthLeft"
        },
        {
            "Y": 0.2706988751888275,
            "X": 0.7250053286552429,
            "Type": "mouthRight"
        }
    ]
}
],
"CelebrityFaces": [
    {
        "MatchConfidence": 100.0,
        "Face": {
            "BoundingBox": {
                "Width": 0.14000000059604645,
                "Top": 0.1190476194024086,
                "Left": 0.82833331823349,
                "Height": 0.2666666805744171
            },
            "Confidence": 99.99359130859375,
            "Pose": {
                "Yaw": -10.509642601013184,
                "Roll": -14.51749324798584,
                "Pitch": 13.799399375915527
            },
            "Quality": {
                "Sharpness": 78.74752044677734,
                "Brightness": 42.201324462890625
            },
            "Landmarks": [
                {
                    "Y": 0.2290833294391632,
                    "X": 0.8709492087364197,
                    "Type": "eyeLeft"
                },
                {
                    "Y": 0.20639978349208832,
                    "X": 0.9153988361358643,
                    "Type": "eyeRight"
                }
            ]
        }
    }
]
```

```
        {
            "Y": 0.25417643785476685,
            "X": 0.8907724022865295,
            "Type": "nose"
        },
        {
            "Y": 0.32729196548461914,
            "X": 0.8876466155052185,
            "Type": "mouthLeft"
        },
        {
            "Y": 0.3115464746952057,
            "X": 0.9238573312759399,
            "Type": "mouthRight"
        }
    ]
},
"Name": "Celeb A",
"Urls": [
    "www.imdb.com/name/aaaaaaaaa"
],
"Id": "1111111"
},
{
    "MatchConfidence": 97.0,
    "Face": {
        "BoundingBox": {
            "Width": 0.13333334028720856,
            "Top": 0.24920634925365448,
            "Left": 0.4449999928474426,
            "Height": 0.2539682686328888
        },
        "Confidence": 99.99979400634766,
        "Pose": {
            "Yaw": 6.557040691375732,
            "Roll": -7.316643714904785,
            "Pitch": 9.272967338562012
        },
        "Quality": {
            "Sharpness": 83.23492431640625,
            "Brightness": 78.83267974853516
        },
        "Landmarks": [
            {
```

```
        "Y": 0.3625510632991791,
        "X": 0.48898839950561523,
        "Type": "eyeLeft"
    },
    {
        "Y": 0.35366007685661316,
        "X": 0.5313721299171448,
        "Type": "eyeRight"
    },
    {
        "Y": 0.3894785940647125,
        "X": 0.5173314809799194,
        "Type": "nose"
    },
    {
        "Y": 0.44889405369758606,
        "X": 0.5020005702972412,
        "Type": "mouthLeft"
    },
    {
        "Y": 0.4408611059188843,
        "X": 0.5351271629333496,
        "Type": "mouthRight"
    }
    ]
},
"Name": "Celeb B",
"Urls": [
    "www.imdb.com/name/bbbbbbbbbb"
],
"Id": "2222222"
},
{
    "MatchConfidence": 100.0,
    "Face": {
        "BoundingBox": {
            "Width": 0.12416666746139526,
            "Top": 0.2968254089355469,
            "Left": 0.2150000035762787,
            "Height": 0.23650793731212616
        },
        "Confidence": 99.99958801269531,
        "Pose": {
            "Yaw": 7.801797866821289,
```

```
        "Roll": -8.326810836791992,
        "Pitch": 7.844768047332764
    },
    "Quality": {
        "Sharpness": 86.93206024169922,
        "Brightness": 79.81291198730469
    },
    "Landmarks": [
        {
            "Y": 0.4027804136276245,
            "X": 0.2575301229953766,
            "Type": "eyeLeft"
        },
        {
            "Y": 0.3934555947780609,
            "X": 0.2956969439983368,
            "Type": "eyeRight"
        },
        {
            "Y": 0.4309830069541931,
            "X": 0.2837020754814148,
            "Type": "nose"
        },
        {
            "Y": 0.48186683654785156,
            "X": 0.26812544465065,
            "Type": "mouthLeft"
        },
        {
            "Y": 0.47338807582855225,
            "X": 0.29905644059181213,
            "Type": "mouthRight"
        }
    ]
},
"Name": "Celeb C",
"Urls": [
    "www.imdb.com/name/ccccccccc"
],
"Id": "3333333"
},
{
    "MatchConfidence": 97.0,
    "Face": {
```



```
"BoundingBox": {
  "Width": 0.11916666477918625,
  "Top": 0.3698412775993347,
  "Left": 0.008333333767950535,
  "Height": 0.22698412835597992
},
"Confidence": 99.99999237060547,
"Pose": {
  "Yaw": 16.38478660583496,
  "Roll": -1.0260354280471802,
  "Pitch": 5.975185394287109
},
"Quality": {
  "Sharpness": 83.23492431640625,
  "Brightness": 61.408443450927734
},
"Landmarks": [
  {
    "Y": 0.4632347822189331,
    "X": 0.049406956881284714,
    "Type": "eyeLeft"
  },
  {
    "Y": 0.46388113498687744,
    "X": 0.08722897619009018,
    "Type": "eyeRight"
  },
  {
    "Y": 0.5020678639411926,
    "X": 0.0758260041475296,
    "Type": "nose"
  },
  {
    "Y": 0.544157862663269,
    "X": 0.054029736667871475,
    "Type": "mouthLeft"
  },
  {
    "Y": 0.5463630557060242,
    "X": 0.08464983850717545,
    "Type": "mouthRight"
  }
]
},
```

```
        "Name": "Celeb D",
        "Urls": [
            "www.imdb.com/name/ddddddddd"
        ],
        "Id": "44444444"
    }
]
```

Untuk informasi selengkapnya, lihat [Mengenali Selebriti dalam Gambar](#) di Panduan Pengembang Rekognition Amazon.

- Untuk detail API, lihat [RecognizeCelebrities](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesRequest;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Celebrity;

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

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

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

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

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

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

    public static void recognizeAllCelebrities(RekognitionClient rekClient,
String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            RecognizeCelebritiesRequest request =
RecognizeCelebritiesRequest.builder()
                .image(souImage)
```

```

        .build();

        RecognizeCelebritiesResponse result =
rekClient.recognizeCelebrities(request);
        List<Celebrity> celebs = result.celebrityFaces();
        System.out.println(celebs.size() + " celebrity(s) were recognized.
\n");

        for (Celebrity celebrity : celebs) {
            System.out.println("Celebrity recognized: " + celebrity.name());
            System.out.println("Celebrity ID: " + celebrity.id());

            System.out.println("Further information (if available):");
            for (String url : celebrity.urls()) {
                System.out.println(url);
            }
            System.out.println();
        }
        System.out.println(result.unrecognizedFaces().size() + " face(s) were
unrecognized.");

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

```

- Untuk detail API, lihat [RecognizeCelebrities](#) di Referensi AWS SDK for Java 2.x API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```

suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =

```

```

        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
        println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
    }
}

```

- Untuk detail API, lihat [RecognizeCelebrities](#) di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```

class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

```

```
def __init__(self, image, image_name, rekognition_client):
    """
    Initializes the image object.

    :param image: Data that defines the image, either the image bytes or
                  an Amazon S3 bucket and object key.
    :param image_name: The name of the image.
    :param rekognition_client: A Boto3 Rekognition client.
    """
    self.image = image
    self.image_name = image_name
    self.rekognition_client = rekognition_client

def recognize_celebrities(self):
    """
    Detects celebrities in the image.

    :return: A tuple. The first element is the list of celebrities found in
             the image. The second element is the list of faces that were
             detected but did not match any known celebrities.
    """
    try:
        response =
self.rekognition_client.recognize_celebrities(Image=self.image)
        celebrities = [
            RekognitionCelebrity(celeb) for celeb in
response["CelebrityFaces"]
        ]
        other_faces = [
            RekognitionFace(face) for face in response["UnrecognizedFaces"]
        ]
        logger.info(
            "Found %s celebrities and %s other faces in %s.",
            len(celebrities),
            len(other_faces),
            self.image_name,
        )
    except ClientError:
        logger.exception("Couldn't detect celebrities in %s.",
self.image_name)
        raise
    else:
        return celebrities, other_faces
```

- Untuk detail API, lihat [RecognizeCelebrities](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **SearchFaces** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `SearchFaces`.

Untuk informasi selengkapnya, lihat [Mencari wajah \(ID wajah\)](#).

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh selengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to find faces in an image that
/// match the face Id provided in the method request.
/// </summary>
public class SearchFacesMatchingId
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        string faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx";
```

```
var rekognitionClient = new AmazonRekognitionClient();

// Search collection for faces matching the face id.
var searchFacesRequest = new SearchFacesRequest
{
    CollectionId = collectionId,
    FaceId = faceId,
    FaceMatchThreshold = 70F,
    MaxFaces = 2,
};

SearchFacesResponse searchFacesResponse = await
rekognitionClient.SearchFacesAsync(searchFacesRequest);

Console.WriteLine("Face matching faceId " + faceId);

Console.WriteLine("Matche(s): ");
searchFacesResponse.FaceMatches.ForEach(face =>
{
    Console.WriteLine($"FaceId: {face.Face.FaceId} Similarity:
{face.Similarity}");
});
}
```

- Untuk detail API, lihat [SearchFaces](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk mencari wajah dalam koleksi yang cocok dengan ID wajah.

`search-faces` Perintah berikut mencari wajah dalam koleksi yang cocok dengan ID wajah yang ditentukan.

```
aws rekognition search-faces \  
  --face-id 8d3cfc70-4ba8-4b36-9644-90fba29c2dac \  
  --collection-id MyCollection
```


Output:

```
{
  "SearchedFaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
  "FaceModelVersion": "3.0",
  "FaceMatches": [
    {
      "Face": {
        "BoundingBox": {
          "Width": 0.48166701197624207,
          "Top": 0.20999999344348907,
          "Left": 0.21250000596046448,
          "Height": 0.36125001311302185
        },
        "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
        "ExternalImageId": "image1.jpg",
        "Confidence": 99.99949645996094,
        "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
      },
      "Similarity": 99.30997467041016
    },
    {
      "Face": {
        "BoundingBox": {
          "Width": 0.18562500178813934,
          "Top": 0.1618019938468933,
          "Left": 0.5575000047683716,
          "Height": 0.24770599603652954
        },
        "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
        "ExternalImageId": "example-image.jpg",
        "Confidence": 99.99340057373047,
        "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
      },
      "Similarity": 99.24862670898438
    },
    {
      "Face": {
        "BoundingBox": {
          "Width": 0.18562500178813934,
          "Top": 0.1618019938468933,
          "Left": 0.5575000047683716,
          "Height": 0.24770599603652954
        },

```

```
        "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
        "ExternalImageId": "image3.jpg",
        "Confidence": 99.99340057373047,
        "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
    },
    "Similarity": 99.24862670898438
},
{
    "Face": {
        "BoundingBox": {
            "Width": 0.5349419713020325,
            "Top": 0.29124999046325684,
            "Left": 0.16389399766921997,
            "Height": 0.40187498927116394
        },
        "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
        "ExternalImageId": "image9.jpg",
        "Confidence": 99.99979400634766,
        "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
    },
    "Similarity": 96.73158264160156
},
{
    "Face": {
        "BoundingBox": {
            "Width": 0.5307819843292236,
            "Top": 0.2862499952316284,
            "Left": 0.1564060002565384,
            "Height": 0.3987500071525574
        },
        "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
        "ExternalImageId": "image10.jpg",
        "Confidence": 99.99970245361328,
        "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
    },
    "Similarity": 96.48291015625
},
{
    "Face": {
        "BoundingBox": {
            "Width": 0.5074880123138428,
            "Top": 0.3774999976158142,
            "Left": 0.18302799761295319,
            "Height": 0.3812499940395355
```

```
    },
    "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
    "ExternalImageId": "image6.jpg",
    "Confidence": 99.99930572509766,
    "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
  },
  "Similarity": 96.43287658691406
},
{
  "Face": {
    "BoundingBox": {
      "Width": 0.5574039816856384,
      "Top": 0.37187498807907104,
      "Left": 0.14559100568294525,
      "Height": 0.4181250035762787
    },
    "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
    "ExternalImageId": "image5.jpg",
    "Confidence": 99.99960327148438,
    "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
  },
  "Similarity": 95.25305938720703
},
{
  "Face": {
    "BoundingBox": {
      "Width": 0.5773710012435913,
      "Top": 0.34437501430511475,
      "Left": 0.12396000325679779,
      "Height": 0.4337500035762787
    },
    "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
    "ExternalImageId": "image8.jpg",
    "Confidence": 100.0,
    "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
  },
  "Similarity": 95.22837829589844
}
]
```

Untuk informasi selengkapnya, lihat [Mencari Wajah Menggunakan ID Wajahnya](#) di Panduan Pengembang Amazon Rekognition.

- Untuk detail API, lihat [SearchFaces](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import
    software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionId> <sourceImage>
```

```
        Where:
            collectionId - The id of the collection. \s
            sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\s

        """;

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

    String collectionId = args[0];
    String sourceImage = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Searching for a face in a collections");
    searchFaceInCollection(rekClient, collectionId, sourceImage);
    rekClient.close();
}

public static void searchFaceInCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(new
File(sourceImage));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
            .image(souImage)
            .maxFaces(10)
            .faceMatchThreshold(70F)
            .collectionId(collectionId)
            .build();
```

```

        SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest);
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face : faceImageMatches) {
            System.out.println("The similarity level is " +
face.similarity());
            System.out.println();
        }

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

```

- Untuk detail API, lihat [SearchFaces](#) di Referensi AWS SDK for Java 2.x API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```

class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
            create_collection.

```

```
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )

    def search_faces(self, face_id, threshold, max_faces):
        """
        Searches for faces in the collection that match another face from the
        collection.

        :param face_id: The ID of the face in the collection to search for.
        :param threshold: The match confidence must be greater than this value
            for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: The list of matching faces found in the collection. This list
does
            not contain the face specified by `face_id`.
        """
        try:
            response = self.rekognition_client.search_faces(
                CollectionId=self.collection_id,
                FaceId=face_id,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
```

```
        )
        faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
        logger.info(
            "Found %s faces in %s that match %s.",
            len(faces),
            self.collection_id,
            face_id,
        )
    except ClientError:
        logger.exception(
            "Couldn't search for faces in %s that match %s.",
            self.collection_id,
            face_id,
        )
        raise
    else:
        return faces
```

- Untuk detail API, lihat [SearchFaces](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **SearchFacesByImage** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `SearchFacesByImage`.

Untuk informasi selengkapnya, lihat [Mencari wajah \(gambar\)](#).

.NET

AWS SDK for .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh selengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).


```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to search for images matching those
/// in a collection.
/// </summary>
public class SearchFacesMatchingImage
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        string bucket = "bucket";
        string photo = "input.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        // Get an image object from S3 bucket.
        var image = new Image()
        {
            S3Object = new S3Object()
            {
                Bucket = bucket,
                Name = photo,
            },
        };

        var searchFacesByImageRequest = new SearchFacesByImageRequest()
        {
            CollectionId = collectionId,
            Image = image,
            FaceMatchThreshold = 70F,
            MaxFaces = 2,
        };

        SearchFacesByImageResponse searchFacesByImageResponse = await
        rekognitionClient.SearchFacesByImageAsync(searchFacesByImageRequest);

        Console.WriteLine("Faces matching largest face in image from " +
        photo);
        searchFacesByImageResponse.FaceMatches.ForEach(face =>
```

```

        {
            Console.WriteLine($"FaceId: {face.Face.FaceId}, Similarity:
{face.Similarity}");
        }
    }
}

```

- Untuk detail API, lihat [SearchFacesByImage](#) di Referensi AWS SDK for .NET API.

CLI

AWS CLI

Untuk mencari wajah dalam koleksi yang cocok dengan wajah terbesar dalam sebuah gambar.

`search-faces-by-image` Perintah berikut mencari wajah dalam koleksi yang cocok dengan wajah terbesar dalam gambar yang ditentukan. :

```

aws rekognition search-faces-by-image \
  --image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"ExamplePerson.jpg"}}' \
  --collection-id MyFaceImageCollection

{
  "SearchedFaceBoundingBox": {
    "Width": 0.18562500178813934,
    "Top": 0.1618015021085739,
    "Left": 0.5575000047683716,
    "Height": 0.24770642817020416
  },
  "SearchedFaceConfidence": 99.993408203125,
  "FaceMatches": [
    {
      "Face": {
        "BoundingBox": {
          "Width": 0.18562500178813934,
          "Top": 0.1618019938468933,
          "Left": 0.5575000047683716,
          "Height": 0.24770599603652954
        },
        "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",

```

```
        "ExternalImageId": "example-image.jpg",
        "Confidence": 99.99340057373047,
        "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
    },
    "Similarity": 99.97913360595703
},
{
    "Face": {
        "BoundingBox": {
            "Width": 0.18562500178813934,
            "Top": 0.1618019938468933,
            "Left": 0.5575000047683716,
            "Height": 0.24770599603652954
        },
        "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
        "ExternalImageId": "image3.jpg",
        "Confidence": 99.99340057373047,
        "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
    },
    "Similarity": 99.97913360595703
},
{
    "Face": {
        "BoundingBox": {
            "Width": 0.41499999165534973,
            "Top": 0.09187500178813934,
            "Left": 0.28083300590515137,
            "Height": 0.3112500011920929
        },
        "FaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
        "ExternalImageId": "image2.jpg",
        "Confidence": 99.99769592285156,
        "ImageId": "a294da46-2cb1-5cc4-9045-61d7ca567662"
    },
    "Similarity": 99.18069458007812
},
{
    "Face": {
        "BoundingBox": {
            "Width": 0.48166701197624207,
            "Top": 0.20999999344348907,
            "Left": 0.21250000596046448,
            "Height": 0.36125001311302185
        },
```

```
        "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
        "ExternalImageId": "image1.jpg",
        "Confidence": 99.99949645996094,
        "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
    },
    "Similarity": 98.66607666015625
},
{
    "Face": {
        "BoundingBox": {
            "Width": 0.5349419713020325,
            "Top": 0.29124999046325684,
            "Left": 0.16389399766921997,
            "Height": 0.40187498927116394
        },
        "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
        "ExternalImageId": "image9.jpg",
        "Confidence": 99.99979400634766,
        "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
    },
    "Similarity": 98.24278259277344
},
{
    "Face": {
        "BoundingBox": {
            "Width": 0.5307819843292236,
            "Top": 0.2862499952316284,
            "Left": 0.1564060002565384,
            "Height": 0.3987500071525574
        },
        "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
        "ExternalImageId": "image10.jpg",
        "Confidence": 99.99970245361328,
        "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
    },
    "Similarity": 98.10665893554688
},
{
    "Face": {
        "BoundingBox": {
            "Width": 0.5074880123138428,
            "Top": 0.3774999976158142,
            "Left": 0.18302799761295319,
            "Height": 0.3812499940395355
```

```
    },
    "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
    "ExternalImageId": "image6.jpg",
    "Confidence": 99.99930572509766,
    "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
  },
  "Similarity": 98.10526275634766
},
{
  "Face": {
    "BoundingBox": {
      "Width": 0.5574039816856384,
      "Top": 0.37187498807907104,
      "Left": 0.14559100568294525,
      "Height": 0.4181250035762787
    },
    "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
    "ExternalImageId": "image5.jpg",
    "Confidence": 99.99960327148438,
    "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
  },
  "Similarity": 97.94659423828125
},
{
  "Face": {
    "BoundingBox": {
      "Width": 0.5773710012435913,
      "Top": 0.34437501430511475,
      "Left": 0.12396000325679779,
      "Height": 0.4337500035762787
    },
    "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
    "ExternalImageId": "image8.jpg",
    "Confidence": 100.0,
    "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
  },
  "Similarity": 97.93476867675781
}
],
"FaceModelVersion": "3.0"
}
```

Untuk informasi selengkapnya, lihat [Mencari Wajah Menggunakan Gambar](#) di Panduan Pengembang Rekognition Amazon.

- Untuk detail API, lihat [SearchFacesByImage](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class SearchFaceMatchingIdCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <sourceImage>

                Where:
                    collectionId - The id of the collection. \s
                    sourceImage - The path to the image (for example, C:\\AWS\\
                    \\pic1.png).\s
    }
}
```

```
        """;

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

    String collectionId = args[0];
    String faceId = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Searching for a face in a collections");
    searchFaceById(rekClient, collectionId, faceId);
    rekClient.close();
}

public static void searchFaceById(RekognitionClient rekClient, String
collectionId, String faceId) {
    try {
        SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
            .collectionId(collectionId)
            .faceId(faceId)
            .faceMatchThreshold(70F)
            .maxFaces(2)
            .build();

        SearchFacesResponse imageResponse =
rekClient.searchFaces(searchFacesRequest);
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face : faceImageMatches) {
            System.out.println("The similarity level is " +
face.similarity());
            System.out.println();
        }

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

```
}
```

- Untuk detail API, lihat [SearchFacesByImage](#) di Referensi AWS SDK for Java 2.x API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
            create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
    collection
)
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.
        """
```



```
:param collection: The collection data.
:return: A tuple of the data in the collection.
"""
return (
    collection.get("CollectionArn"),
    collection.get("FaceCount", 0),
    collection.get("CreationTimestamp"),
)

def search_faces_by_image(self, image, threshold, max_faces):
    """
    Searches for faces in the collection that match the largest face in the
    reference image.

    :param image: The image that contains the reference face to search for.
    :param threshold: The match confidence must be greater than this value
        for a face to be included in the results.
    :param max_faces: The maximum number of faces to return.
    :return: A tuple. The first element is the face found in the reference
image.
        The second element is the list of matching faces found in the
collection.
    """
    try:
        response = self.rekognition_client.search_faces_by_image(
            CollectionId=self.collection_id,
            Image=image.image,
            FaceMatchThreshold=threshold,
            MaxFaces=max_faces,
        )
        image_face = RekognitionFace(
            {
                "BoundingBox": response["SearchedFaceBoundingBox"],
                "Confidence": response["SearchedFaceConfidence"],
            }
        )
        collection_faces = [
            RekognitionFace(face["Face"]) for face in response["FaceMatches"]
        ]
        logger.info(
            "Found %s faces in the collection that match the largest "
            "face in %s.",
            len(collection_faces),

```

```
        image.image_name,
    )
except ClientError:
    logger.exception(
        "Couldn't search for faces in %s that match %s.",
        self.collection_id,
        image.image_name,
    )
    raise
else:
    return image_face, collection_faces
```

- Untuk detail API, lihat [SearchFacesByImage](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Skenario untuk Amazon AWS Rekognition menggunakan SDK

Contoh kode berikut menunjukkan cara menerapkan skenario umum di Amazon AWS Rekognition dengan SDK. Skenario ini menunjukkan kepada Anda cara menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam Amazon Rekognition. Setiap skenario menyertakan tautan ke GitHub, di mana Anda dapat menemukan petunjuk tentang cara mengatur dan menjalankan kode.

Contoh

- [Buat koleksi Amazon Rekognition dan temukan wajah di dalamnya menggunakan SDK AWS](#)
- [Mendeteksi dan menampilkan elemen dalam gambar dengan Amazon Rekognition menggunakan SDK AWS](#)
- [Mendeteksi informasi dalam video menggunakan Amazon Rekognition dan SDK AWS](#)

Buat koleksi Amazon Rekognition dan temukan wajah di dalamnya menggunakan SDK AWS

Contoh kode berikut ini menunjukkan cara:

- Buat koleksi Amazon Rekognition.
- Tambahkan gambar ke koleksi dan deteksi wajah di dalamnya.
- Cari koleksi untuk wajah yang cocok dengan gambar referensi.
- Hapus koleksi.

Untuk informasi selengkapnya, lihat [Mencari wajah dalam koleksi](#).

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

Buat kelas yang membungkus fungsi Amazon Rekognition.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
from rekognition_objects import RekognitionFace
from rekognition_image_detection import RekognitionImage

logger = logging.getLogger(__name__)

class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
            an Amazon S3 bucket and object key.
```

```
    :param image_name: The name of the image.
    :param rekognition_client: A Boto3 Rekognition client.
    """
    self.image = image
    self.image_name = image_name
    self.rekognition_client = rekognition_client

    @classmethod
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened
and its
                                bytes are read.
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
                                file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from
the
                                file.
        """
        with open(image_file_name, "rb") as img_file:
            image = {"Bytes": img_file.read()}
            name = image_file_name if image_name is None else image_name
            return cls(image, name, rekognition_client)

class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client
```

```
def create_collection(self, collection_id):
    """
    Creates an empty collection.

    :param collection_id: Text that identifies the collection.
    :return: The newly created collection.
    """
    try:
        response = self.rekognition_client.create_collection(
            CollectionId=collection_id
        )
        response["CollectionId"] = collection_id
        collection = RekognitionCollection(response, self.rekognition_client)
        logger.info("Created collection %s.", collection_id)
    except ClientError:
        logger.exception("Couldn't create collection %s.", collection_id)
        raise
    else:
        return collection

def list_collections(self, max_results):
    """
    Lists collections for the current account.

    :param max_results: The maximum number of collections to return.
    :return: The list of collections for the current account.
    """
    try:
        response =
self.rekognition_client.list_collections(MaxResults=max_results)
        collections = [
            RekognitionCollection({"CollectionId": col_id},
self.rekognition_client)
            for col_id in response["CollectionIds"]
        ]
    except ClientError:
        logger.exception("Couldn't list collections.")
        raise
    else:
        return collections
```

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
            create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
    collection
)
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )

    def to_dict(self):
        """
        Renders parts of the collection data to a dict.

        :return: The collection data as a dict.
        """
```

```
    rendering = {
        "collection_id": self.collection_id,
        "collection_arn": self.collection_arn,
        "face_count": self.face_count,
        "created": self.created,
    }
    return rendering

def describe_collection(self):
    """
    Gets data about the collection from the Amazon Rekognition service.

    :return: The collection rendered as a dict.
    """
    try:
        response = self.rekognition_client.describe_collection(
            CollectionId=self.collection_id
        )
        # Work around capitalization of Arn vs. ARN
        response["CollectionArn"] = response.get("CollectionARN")
        (
            self.collection_arn,
            self.face_count,
            self.created,
        ) = self._unpack_collection(response)
        logger.info("Got data for collection %s.", self.collection_id)
    except ClientError:
        logger.exception("Couldn't get data for collection %s.",
            self.collection_id)
        raise
    else:
        return self.to_dict()

def delete_collection(self):
    """
    Deletes the collection.
    """
    try:
        self.rekognition_client.delete_collection(CollectionId=self.collection_id)
        logger.info("Deleted collection %s.", self.collection_id)
        self.collection_id = None
```

```
    except ClientError:
        logger.exception("Couldn't delete collection %s.",
self.collection_id)
        raise

def index_faces(self, image, max_faces):
    """
    Finds faces in the specified image, indexes them, and stores them in the
    collection.

    :param image: The image to index.
    :param max_faces: The maximum number of faces to index.
    :return: A tuple. The first element is a list of indexed faces.
             The second element is a list of faces that couldn't be indexed.
    """
    try:
        response = self.rekognition_client.index_faces(
            CollectionId=self.collection_id,
            Image=image.image,
            ExternalImageId=image.image_name,
            MaxFaces=max_faces,
            DetectionAttributes=["ALL"],
        )
        indexed_faces = [
            RekognitionFace(**face["Face"], **face["FaceDetail"])
            for face in response["FaceRecords"]
        ]
        unindexed_faces = [
            RekognitionFace(face["FaceDetail"])
            for face in response["UnindexedFaces"]
        ]
        logger.info(
            "Indexed %s faces in %s. Could not index %s faces.",
            len(indexed_faces),
            image.image_name,
            len(unindexed_faces),
        )
    except ClientError:
        logger.exception("Couldn't index faces in image %s.",
image.image_name)
        raise
    else:
        return indexed_faces, unindexed_faces
```



```
def list_faces(self, max_results):
    """
    Lists the faces currently indexed in the collection.

    :param max_results: The maximum number of faces to return.
    :return: The list of faces in the collection.
    """
    try:
        response = self.rekognition_client.list_faces(
            CollectionId=self.collection_id, MaxResults=max_results
        )
        faces = [RekognitionFace(face) for face in response["Faces"]]
        logger.info(
            "Found %s faces in collection %s.", len(faces),
self.collection_id
        )
    except ClientError:
        logger.exception(
            "Couldn't list faces in collection %s.", self.collection_id
        )
        raise
    else:
        return faces

def search_faces(self, face_id, threshold, max_faces):
    """
    Searches for faces in the collection that match another face from the
    collection.

    :param face_id: The ID of the face in the collection to search for.
    :param threshold: The match confidence must be greater than this value
        for a face to be included in the results.
    :param max_faces: The maximum number of faces to return.
    :return: The list of matching faces found in the collection. This list
does
        not contain the face specified by `face_id`.
    """
    try:
        response = self.rekognition_client.search_faces(
            CollectionId=self.collection_id,
            FaceId=face_id,
```

```
        FaceMatchThreshold=threshold,
        MaxFaces=max_faces,
    )
    faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
    logger.info(
        "Found %s faces in %s that match %s.",
        len(faces),
        self.collection_id,
        face_id,
    )
except ClientError:
    logger.exception(
        "Couldn't search for faces in %s that match %s.",
        self.collection_id,
        face_id,
    )
    raise
else:
    return faces

def search_faces_by_image(self, image, threshold, max_faces):
    """
    Searches for faces in the collection that match the largest face in the
    reference image.

    :param image: The image that contains the reference face to search for.
    :param threshold: The match confidence must be greater than this value
        for a face to be included in the results.
    :param max_faces: The maximum number of faces to return.
    :return: A tuple. The first element is the face found in the reference
    image.

        The second element is the list of matching faces found in the
        collection.
    """
    try:
        response = self.rekognition_client.search_faces_by_image(
            CollectionId=self.collection_id,
            Image=image.image,
            FaceMatchThreshold=threshold,
            MaxFaces=max_faces,
        )
        image_face = RekognitionFace(
```

```
        {
            "BoundingBox": response["SearchedFaceBoundingBox"],
            "Confidence": response["SearchedFaceConfidence"],
        }
    )
    collection_faces = [
        RekognitionFace(face["Face"]) for face in response["FaceMatches"]
    ]
    logger.info(
        "Found %s faces in the collection that match the largest "
        "face in %s.",
        len(collection_faces),
        image.image_name,
    )
except ClientError:
    logger.exception(
        "Couldn't search for faces in %s that match %s.",
        self.collection_id,
        image.image_name,
    )
    raise
else:
    return image_face, collection_faces
```

```
class RekognitionFace:
```

```
    """Encapsulates an Amazon Rekognition face."""
```

```
    def __init__(self, face, timestamp=None):
```

```
        """
```

```
        Initializes the face object.
```

```
        :param face: Face data, in the format returned by Amazon Rekognition
                    functions.
```

```
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
```

```
        """
```

```
        self.bounding_box = face.get("BoundingBox")
```

```
        self.confidence = face.get("Confidence")
```

```
        self.landmarks = face.get("Landmarks")
```

```
        self.pose = face.get("Pose")
```

```
        self.quality = face.get("Quality")
```

```
        age_range = face.get("AgeRange")
```

```
        if age_range is not None:
```

```
        self.age_range = (age_range.get("Low"), age_range.get("High"))
    else:
        self.age_range = None
    self.smile = face.get("Smile", {}).get("Value")
    self.eyeglasses = face.get("Eyeglasses", {}).get("Value")
    self.sunglasses = face.get("Sunglasses", {}).get("Value")
    self.gender = face.get("Gender", {}).get("Value", None)
    self.beard = face.get("Beard", {}).get("Value")
    self.mustache = face.get("Mustache", {}).get("Value")
    self.eyes_open = face.get("EyesOpen", {}).get("Value")
    self.mouth_open = face.get("MouthOpen", {}).get("Value")
    self.emotions = [
        emo.get("Type")
        for emo in face.get("Emotions", [])
        if emo.get("Confidence", 0) > 50
    ]
    self.face_id = face.get("FaceId")
    self.image_id = face.get("ImageId")
    self.timestamp = timestamp

def to_dict(self):
    """
    Renders some of the face data to a dict.

    :return: A dict that contains the face data.
    """
    rendering = {}
    if self.bounding_box is not None:
        rendering["bounding_box"] = self.bounding_box
    if self.age_range is not None:
        rendering["age"] = f"{self.age_range[0]} - {self.age_range[1]}"
    if self.gender is not None:
        rendering["gender"] = self.gender
    if self.emotions:
        rendering["emotions"] = self.emotions
    if self.face_id is not None:
        rendering["face_id"] = self.face_id
    if self.image_id is not None:
        rendering["image_id"] = self.image_id
    if self.timestamp is not None:
        rendering["timestamp"] = self.timestamp
    has = []
    if self.smile:
        has.append("smile")
```

```
if self.eyeglasses:
    has.append("eyeglasses")
if self.sunglasses:
    has.append("sunglasses")
if self.beard:
    has.append("beard")
if self.mustache:
    has.append("mustache")
if self.eyes_open:
    has.append("open eyes")
if self.mouth_open:
    has.append("open mouth")
if has:
    rendering["has"] = has
return rendering
```

Gunakan kelas pembungkus untuk membangun koleksi wajah dari satu set gambar dan kemudian mencari wajah dalam koleksi.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Rekognition face collection demo!")
    print("-" * 88)

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

    rekognition_client = boto3.client("rekognition")
    images = [
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128316.jpg",
            rekognition_client,
            image_name="sitting",
        ),
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128317.jpg",
            rekognition_client,
            image_name="hopping",
        ),
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128318.jpg",
```

```
        rekognition_client,
        image_name="biking",
    ),
]

collection_mgr = RekognitionCollectionManager(rekognition_client)
collection = collection_mgr.create_collection("doc-example-collection-demo")
print(f"Created collection {collection.collection_id}")
pprint(collection.describe_collection())

print("Indexing faces from three images:")
for image in images:
    collection.index_faces(image, 10)
print("Listing faces in collection:")
faces = collection.list_faces(10)
for face in faces:
    pprint(face.to_dict())
input("Press Enter to continue.")

print(
    f"Searching for faces in the collection that match the first face in the "
    f"list (Face ID: {faces[0].face_id}."
)
found_faces = collection.search_faces(faces[0].face_id, 80, 10)
print(f"Found {len(found_faces)} matching faces.")
for face in found_faces:
    pprint(face.to_dict())
input("Press Enter to continue.")

print(
    f"Searching for faces in the collection that match the largest face in "
    f"{images[0].image_name}."
)
image_face, match_faces = collection.search_faces_by_image(images[0], 80, 10)
print(f"The largest face in {images[0].image_name} is:")
pprint(image_face.to_dict())
print(f"Found {len(match_faces)} matching faces.")
for face in match_faces:
    pprint(face.to_dict())
input("Press Enter to continue.")

collection.delete_collection()
print("Thanks for watching!")
```

```
print("-" * 88)
```

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mendeteksi dan menampilkan elemen dalam gambar dengan Amazon Rekognition menggunakan SDK AWS

Contoh kode berikut ini menunjukkan cara:

- Mendeteksi elemen dalam gambar dengan menggunakan Amazon Rekognition.
- Tampilkan gambar dan gambar kotak pembatas di sekitar elemen yang terdeteksi.

Untuk informasi selengkapnya, lihat [Menampilkan kotak pembatas](#).

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh selengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

Buat kelas untuk membungkus fungsi Amazon Rekognition.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests

from rekognition_objects import (
    RekognitionFace,
```

```
RekognitionCelebrity,
RekognitionLabel,
RekognitionModerationLabel,
RekognitionText,
show_bounding_boxes,
show_polygons,
)

logger = logging.getLogger(__name__)

class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
            an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client

    @classmethod
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened
        and its
            bytes are read.
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
            file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from
        the
```



```
        file.
    """
    with open(image_file_name, "rb") as img_file:
        image = {"Bytes": img_file.read()}
        name = image_file_name if image_name is None else image_name
    return cls(image, name, rekognition_client)

    @classmethod
    def from_bucket(cls, s3_object, rekognition_client):
        """
        Creates a RekognitionImage object from an Amazon S3 object.

        :param s3_object: An Amazon S3 object that identifies the image. The
image
                        is not retrieved until needed for a later call.
        :param rekognition_client: A Boto3 Rekognition client.
        :return: The RekognitionImage object, initialized with Amazon S3 object
data.
        """
        image = {"S3Object": {"Bucket": s3_object.bucket_name, "Name":
s3_object.key}}
        return cls(image, s3_object.key, rekognition_client)

    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
            raise
        else:
            return faces
```

```
def detect_labels(self, max_labels):
    """
    Detects labels in the image. Labels are objects and people.

    :param max_labels: The maximum number of labels to return.
    :return: The list of labels detected in the image.
    """
    try:
        response = self.rekognition_client.detect_labels(
            Image=self.image, MaxLabels=max_labels
        )
        labels = [RekognitionLabel(label) for label in response["Labels"]]
        logger.info("Found %s labels in %s.", len(labels), self.image_name)
    except ClientError:
        logger.info("Couldn't detect labels in %s.", self.image_name)
        raise
    else:
        return labels

def recognize_celebrities(self):
    """
    Detects celebrities in the image.

    :return: A tuple. The first element is the list of celebrities found in
             the image. The second element is the list of faces that were
             detected but did not match any known celebrities.
    """
    try:
        response =
self.rekognition_client.recognize_celebrities(Image=self.image)
        celebrities = [
            RekognitionCelebrity(celeb) for celeb in
response["CelebrityFaces"]
        ]
        other_faces = [
            RekognitionFace(face) for face in response["UnrecognizedFaces"]
        ]
        logger.info(
            "Found %s celebrities and %s other faces in %s.",
            len(celebrities),
            len(other_faces),
            self.image_name,
        )
    )
```

```
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.",
self.image_name)
            raise
        else:
            return celebrities, other_faces

def compare_faces(self, target_image, similarity):
    """
    Compares faces in the image with the largest face in the target image.

    :param target_image: The target image to compare against.
    :param similarity: Faces in the image must have a similarity value
greater
                        than this value to be included in the results.
    :return: A tuple. The first element is the list of faces that match the
reference image. The second element is the list of faces that
have
                a similarity value below the specified threshold.
    """
    try:
        response = self.rekognition_client.compare_faces(
            SourceImage=self.image,
            TargetImage=target_image.image,
            SimilarityThreshold=similarity,
        )
        matches = [
            RekognitionFace(match["Face"]) for match in
response["FaceMatches"]
        ]
        unmatched_faces = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
        logger.info(
            "Found %s matched faces and %s unmatched faces.",
            len(matches),
            len(unmatched_faces),
        )
    except ClientError:
        logger.exception(
            "Couldn't match faces from %s to %s.",
            self.image_name,
            target_image.image_name,
```

```
        )
        raise
    else:
        return matches, unmatches

def detect_moderation_labels(self):
    """
    Detects moderation labels in the image. Moderation labels identify
    content
    that may be inappropriate for some audiences.

    :return: The list of moderation labels found in the image.
    """
    try:
        response = self.rekognition_client.detect_moderation_labels(
            Image=self.image
        )
        labels = [
            RekognitionModerationLabel(label)
            for label in response["ModerationLabels"]
        ]
        logger.info(
            "Found %s moderation labels in %s.", len(labels), self.image_name
        )
    except ClientError:
        logger.exception(
            "Couldn't detect moderation labels in %s.", self.image_name
        )
        raise
    else:
        return labels

def detect_text(self):
    """
    Detects text in the image.

    :return The list of text elements found in the image.
    """
    try:
        response = self.rekognition_client.detect_text(Image=self.image)
        texts = [RekognitionText(text) for text in
response["TextDetections"]]
```

```
        logger.info("Found %s texts in %s.", len(texts), self.image_name)
    except ClientError:
        logger.exception("Couldn't detect text in %s.", self.image_name)
        raise
    else:
        return texts
```

Buat fungsi pembantu untuk menggambar kotak pembatas dan poligon.

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

logger = logging.getLogger(__name__)

def show_bounding_boxes(image_bytes, box_sets, colors):
    """
    Draws bounding boxes on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param box_sets: A list of lists of bounding boxes to draw on the image.
    :param colors: A list of colors to use to draw the bounding boxes.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for boxes, color in zip(box_sets, colors):
        for box in boxes:
            left = image.width * box["Left"]
            top = image.height * box["Top"]
            right = (image.width * box["Width"]) + left
            bottom = (image.height * box["Height"]) + top
            draw.rectangle([left, top, right, bottom], outline=color, width=3)
    image.show()

def show_polygons(image_bytes, polygons, color):
    """
    Draws polygons on an image and shows it with the default image viewer.
```

```
:param image_bytes: The image to draw, as bytes.
:param polygons: The list of polygons to draw on the image.
:param color: The color to use to draw the polygons.
"""
image = Image.open(io.BytesIO(image_bytes))
draw = ImageDraw.Draw(image)
for polygon in polygons:
    draw.polygon(
        [
            (image.width * point["X"], image.height * point["Y"])
            for point in polygon
        ],
        outline=color,
    )
image.show()
```

Buat kelas untuk mengurai objek yang dikembalikan oleh Amazon Rekognition.

```
class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""

    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
            functions.
        :param timestamp: The time when the face was detected, if the face was
            detected in a video.
        """
        self.bounding_box = face.get("BoundingBox")
        self.confidence = face.get("Confidence")
        self.landmarks = face.get("Landmarks")
        self.pose = face.get("Pose")
        self.quality = face.get("Quality")
        age_range = face.get("AgeRange")
        if age_range is not None:
            self.age_range = (age_range.get("Low"), age_range.get("High"))
        else:
            self.age_range = None
```

```
self.smile = face.get("Smile", {}).get("Value")
self.eyeglasses = face.get("Eyeglasses", {}).get("Value")
self.sunglasses = face.get("Sunglasses", {}).get("Value")
self.gender = face.get("Gender", {}).get("Value", None)
self.beard = face.get("Beard", {}).get("Value")
self.mustache = face.get("Mustache", {}).get("Value")
self.eyes_open = face.get("EyesOpen", {}).get("Value")
self.mouth_open = face.get("MouthOpen", {}).get("Value")
self.emotions = [
    emo.get("Type")
    for emo in face.get("Emotions", [])
    if emo.get("Confidence", 0) > 50
]
self.face_id = face.get("FaceId")
self.image_id = face.get("ImageId")
self.timestamp = timestamp

def to_dict(self):
    """
    Renders some of the face data to a dict.

    :return: A dict that contains the face data.
    """
    rendering = {}
    if self.bounding_box is not None:
        rendering["bounding_box"] = self.bounding_box
    if self.age_range is not None:
        rendering["age"] = f"{self.age_range[0]} - {self.age_range[1]}"
    if self.gender is not None:
        rendering["gender"] = self.gender
    if self.emotions:
        rendering["emotions"] = self.emotions
    if self.face_id is not None:
        rendering["face_id"] = self.face_id
    if self.image_id is not None:
        rendering["image_id"] = self.image_id
    if self.timestamp is not None:
        rendering["timestamp"] = self.timestamp
    has = []
    if self.smile:
        has.append("smile")
    if self.eyeglasses:
        has.append("eyeglasses")
    if self.sunglasses:
```

```
        has.append("sunglasses")
    if self.beard:
        has.append("beard")
    if self.mustache:
        has.append("mustache")
    if self.eyes_open:
        has.append("open eyes")
    if self.mouth_open:
        has.append("open mouth")
    if has:
        rendering["has"] = has
    return rendering
```

```
class RekognitionCelebrity:
    """Encapsulates an Amazon Rekognition celebrity."""

    def __init__(self, celebrity, timestamp=None):
        """
        Initializes the celebrity object.

        :param celebrity: Celebrity data, in the format returned by Amazon
        Rekognition
                           functions.
        :param timestamp: The time when the celebrity was detected, if the
        celebrity
                           was detected in a video.
        """
        self.info_urls = celebrity.get("Urls")
        self.name = celebrity.get("Name")
        self.id = celebrity.get("Id")
        self.face = RekognitionFace(celebrity.get("Face"))
        self.confidence = celebrity.get("MatchConfidence")
        self.bounding_box = celebrity.get("BoundingBox")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the celebrity data to a dict.

        :return: A dict that contains the celebrity data.
        """
        rendering = self.face.to_dict()
```



```
if self.name is not None:
    rendering["name"] = self.name
if self.info_urls:
    rendering["info URLs"] = self.info_urls
if self.timestamp is not None:
    rendering["timestamp"] = self.timestamp
return rendering
```

```
class RekognitionPerson:
    """Encapsulates an Amazon Rekognition person."""

    def __init__(self, person, timestamp=None):
        """
        Initializes the person object.

        :param person: Person data, in the format returned by Amazon Rekognition
            functions.
        :param timestamp: The time when the person was detected, if the person
            was detected in a video.
        """
        self.index = person.get("Index")
        self.bounding_box = person.get("BoundingBox")
        face = person.get("Face")
        self.face = RekognitionFace(face) if face is not None else None
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the person data to a dict.

        :return: A dict that contains the person data.
        """
        rendering = self.face.to_dict() if self.face is not None else {}
        if self.index is not None:
            rendering["index"] = self.index
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering
```

```
class RekognitionLabel:
    """Encapsulates an Amazon Rekognition label."""

    def __init__(self, label, timestamp=None):
        """
        Initializes the label object.

        :param label: Label data, in the format returned by Amazon Rekognition
            functions.
        :param timestamp: The time when the label was detected, if the label
            was detected in a video.
        """
        self.name = label.get("Name")
        self.confidence = label.get("Confidence")
        self.instances = label.get("Instances")
        self.parents = label.get("Parents")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the label data to a dict.

        :return: A dict that contains the label data.
        """
        rendering = {}
        if self.name is not None:
            rendering["name"] = self.name
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering

class RekognitionModerationLabel:
    """Encapsulates an Amazon Rekognition moderation label."""

    def __init__(self, label, timestamp=None):
        """
        Initializes the moderation label object.

        :param label: Label data, in the format returned by Amazon Rekognition
            functions.
        :param timestamp: The time when the moderation label was detected, if the
```

```
        label was detected in a video.
    """
    self.name = label.get("Name")
    self.confidence = label.get("Confidence")
    self.parent_name = label.get("ParentName")
    self.timestamp = timestamp

def to_dict(self):
    """
    Renders some of the moderation label data to a dict.

    :return: A dict that contains the moderation label data.
    """
    rendering = {}
    if self.name is not None:
        rendering["name"] = self.name
    if self.parent_name is not None:
        rendering["parent_name"] = self.parent_name
    if self.timestamp is not None:
        rendering["timestamp"] = self.timestamp
    return rendering

class RekognitionText:
    """Encapsulates an Amazon Rekognition text element."""

    def __init__(self, text_data):
        """
        Initializes the text object.

        :param text_data: Text data, in the format returned by Amazon Rekognition
            functions.
        """
        self.text = text_data.get("DetectedText")
        self.kind = text_data.get("Type")
        self.id = text_data.get("Id")
        self.parent_id = text_data.get("ParentId")
        self.confidence = text_data.get("Confidence")
        self.geometry = text_data.get("Geometry")

    def to_dict(self):
        """
        Renders some of the text data to a dict.
```

```

:return: A dict that contains the text data.
"""
rendering = {}
if self.text is not None:
    rendering["text"] = self.text
if self.kind is not None:
    rendering["kind"] = self.kind
if self.geometry is not None:
    rendering["polygon"] = self.geometry.get("Polygon")
return rendering

```

Gunakan kelas pembungkus untuk mendeteksi elemen dalam gambar dan menampilkan kotak pembatasnya. Gambar yang digunakan dalam contoh ini dapat ditemukan [GitHub](#) bersama dengan instruksi dan lebih banyak kode.

```

def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Rekognition image detection demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    rekognition_client = boto3.client("rekognition")
    street_scene_file_name = ".media/pexels-kaique-rocha-109919.jpg"
    celebrity_file_name = ".media/pexels-pixabay-53370.jpg"
    one_girl_url = "https://dhei5unw3vrsx.cloudfront.net/images/
source3_resized.jpg"
    three_girls_url = "https://dhei5unw3vrsx.cloudfront.net/images/
target3_resized.jpg"
    swimwear_object = boto3.resource("s3").Object(
        "console-sample-images-pdx", "yoga_swimwear.jpg"
    )
    book_file_name = ".media/pexels-christina-morillo-1181671.jpg"

    street_scene_image = RekognitionImage.from_file(
        street_scene_file_name, rekognition_client
    )
    print(f"Detecting faces in {street_scene_image.image_name}...")
    faces = street_scene_image.detect_faces()
    print(f"Found {len(faces)} faces, here are the first three.")

```

```
for face in faces[:3]:
    pprint(face.to_dict())
show_bounding_boxes(
    street_scene_image.image["Bytes"],
    [[face.bounding_box for face in faces]],
    ["aqua"],
)
input("Press Enter to continue.")

print(f"Detecting labels in {street_scene_image.image_name}...")
labels = street_scene_image.detect_labels(100)
print(f"Found {len(labels)} labels.")
for label in labels:
    pprint(label.to_dict())
names = []
box_sets = []
colors = ["aqua", "red", "white", "blue", "yellow", "green"]
for label in labels:
    if label.instances:
        names.append(label.name)
        box_sets.append([inst["BoundingBox"] for inst in label.instances])
print(f"Showing bounding boxes for {names} in {colors[:len(names)]}.")
show_bounding_boxes(
    street_scene_image.image["Bytes"], box_sets, colors[: len(names)]
)
input("Press Enter to continue.")

celebrity_image = RekognitionImage.from_file(
    celebrity_file_name, rekognition_client
)
print(f"Detecting celebrities in {celebrity_image.image_name}...")
celebs, others = celebrity_image.recognize_celebrities()
print(f"Found {len(celebs)} celebrities.")
for celeb in celebs:
    pprint(celeb.to_dict())
show_bounding_boxes(
    celebrity_image.image["Bytes"],
    [[celeb.face.bounding_box for celeb in celebs]],
    ["aqua"],
)
input("Press Enter to continue.")

girl_image_response = requests.get(one_girl_url)
girl_image = RekognitionImage(
```

```
        {"Bytes": girl_image_response.content}, "one-girl", rekognition_client
    )
    group_image_response = requests.get(three_girls_url)
    group_image = RekognitionImage(
        {"Bytes": group_image_response.content}, "three-girls",
    rekognition_client
    )
    print("Comparing reference face to group of faces...")
    matches, unmatches = girl_image.compare_faces(group_image, 80)
    print(f"Found {len(matches)} face matching the reference face.")
    show_bounding_boxes(
        group_image.image["Bytes"],
        [[match.bounding_box for match in matches]],
        ["aqua"],
    )
    input("Press Enter to continue.")

    swimwear_image = RekognitionImage.from_bucket(swimwear_object,
    rekognition_client)
    print(f"Detecting suggestive content in {swimwear_object.key}...")
    labels = swimwear_image.detect_moderation_labels()
    print(f"Found {len(labels)} moderation labels.")
    for label in labels:
        pprint(label.to_dict())
    input("Press Enter to continue.")

    book_image = RekognitionImage.from_file(book_file_name, rekognition_client)
    print(f"Detecting text in {book_image.image_name}...")
    texts = book_image.detect_text()
    print(f"Found {len(texts)} text instances. Here are the first seven:")
    for text in texts[:7]:
        pprint(text.to_dict())
    show_polygons(
        book_image.image["Bytes"], [text.geometry["Polygon"] for text in texts],
    "aqua"
    )

    print("Thanks for watching!")
    print("-" * 88)
```

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mendeteksi informasi dalam video menggunakan Amazon Rekognition dan SDK AWS

Contoh kode berikut ini menunjukkan cara:

- Mulai pekerjaan Amazon Rekognition untuk mendeteksi elemen seperti orang, objek, dan teks dalam video.
- Periksa status pekerjaan sampai pekerjaan selesai.
- Output daftar elemen yang terdeteksi oleh setiap pekerjaan.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

Dapatkan hasil selebriti dari video yang terletak di ember Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
```

```
import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;

/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class VideoCelebrityDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```



```
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

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

    startCelebrityDetection(rekClient, channel, bucket, video);
    getCelebrityDetectionResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

public static void startCelebrityDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

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

        StartCelebrityRecognitionRequest recognitionRequest =
        StartCelebrityRecognitionRequest.builder()
            .jobTag("Celebrities")
            .notificationChannel(channel)
            .video(vidObj)
            .build();
```

```
        StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient
            .startCelebrityRecognition(recognitionRequest);
        startJobId = startCelebrityRecognitionResult.jobId();

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

public static void getCelebrityDetectionResults(RekognitionClient rekClient)
{
    try {
        String paginationToken = null;
        GetCelebrityRecognitionResponse recognitionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (recognitionResponse != null)
                paginationToken = recognitionResponse.nextToken();

            GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {
rekClient.getCelebrityRecognition(recognitionRequest);
                status = recognitionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
            }
        }
    }
}
```

```
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is
    null.
    VideoMetadata videoMetaData =
    recognitionResponse.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " +
    videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<CelebrityRecognition> celebs =
    recognitionResponse.celebrities();
    for (CelebrityRecognition celeb : celebs) {
        long seconds = celeb.timestamp() / 1000;
        System.out.print("Sec: " + seconds + " ");
        CelebrityDetail details = celeb.celebrity();
        System.out.println("Name: " + details.name());
        System.out.println("Id: " + details.id());
        System.out.println();
    }

    } while (recognitionResponse.nextToken() != null);

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

Mendeteksi label dalam video dengan operasi deteksi label.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
```

```
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

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

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>
```

```
        Where:
            bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
            video - The name of the video (for example, people.mp4).\s
            queueUrl- The URL of a SQS queue.\s
            topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
            roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
        """;

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

    String bucket = args[0];
    String video = args[1];
    String queueUrl = args[2];
    String topicArn = args[3];
    String roleArn = args[4];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    SqsClient sqs = SqsClient.builder()
        .region(Region.US_EAST_1)
        .build();

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

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

public static void startLabels(RekognitionClient rekClient,
    NotificationChannel channel,
```

```
        String bucket,
        String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vid0b = Video.builder()
            .s3object(s3obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
        StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vid0b)
            .minConfidence(50F)
            .build();

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

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

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

            GetLabelDetectionResponse result =
            rekClient.getLabelDetection(detectionRequest);
            status = result.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                ans = false;
            else
                System.out.println(yy + " status is: " + status);
        }
    }
}
```

```
        Thread.sleep(1000);
        yy++;
    }

    System.out.println(startJobId + " status is: " + status);

} catch (RekognitionException | InterruptedException e) {
    e.getMessage();
    System.exit(1);
}
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message : messages) {
                String notification = message.body();

                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonMessageTree = mapper.readTree(notification);
                JsonNode messageBodyText = jsonMessageTree.get("Message");
                ObjectMapper operationResultMapper = new ObjectMapper();
                JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                JsonNode operationJobId = jsonResultTree.get("JobId");
                JsonNode operationStatus = jsonResultTree.get("Status");
                System.out.println("Job found in JSON is " + operationJobId);

                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

                String jobId = operationJobId.textValue();
                if (startJobId.compareTo(jobId) == 0) {
```

```
        System.out.println("Job id: " + operationJobId);
        System.out.println("Status : " +
operationStatus.toString());

        if (operationStatus.asText().equals("SUCCEEDED"))
            getResultsLabels(rekClient);
        else
            System.out.println("Video analysis failed");

        sqs.deleteMessage(deleteMessageRequest);
    } else {
        System.out.println("Job received was not job " +
startJobId);
        sqs.deleteMessage(deleteMessageRequest);
    }
}

} catch (RekognitionException e) {
    e.getMessage();
    System.exit(1);
} catch (JsonMappingException e) {
    e.printStackTrace();
} catch (JsonProcessingException e) {
    e.printStackTrace();
}
}

// Gets the job results by calling GetLabelDetection
private static void getResultsLabels(RekognitionClient rekClient) {

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

    try {
        do {
            if (labelDetectionResult != null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
```



```
        .maxResults(maxResults)
        .nextToken(paginationToken)
        .build();

    labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
    VideoMetadata videoMetaData =
labelDetectionResult.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " +
videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());

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

        System.out.println("  Label:" + label.name());
        System.out.println("  Confidence:" +
detectedLabel.label().confidence().toString());

        List<Instance> instances = label.instances();
        System.out.println("  Instances of " + label.name());

        if (instances.isEmpty()) {
            System.out.println("          " + "None");
        } else {
            for (Instance instance : instances) {
                System.out.println("          Confidence: " +
instance.confidence().toString());
                System.out.println("          Bounding box: " +
instance.boundingBox().toString());
            }
        }
        System.out.println("  Parent labels for " + label.name() +
":");

        List<Parent> parents = label.parents();

        if (parents.isEmpty()) {
            System.out.println("          None");
        }
    }
}
```

```

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

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

```

Mendeteksi wajah dalam video yang disimpan dalam bucket Amazon S3.

```

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

```

```
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

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

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
                (for example, (for example, myBucket).\s
                video - The name of the video (for example, people.mp4).\s
                queueUrl- The URL of a SQS queue.\s
                topicArn - The ARN of the Amazon Simple Notification Service
                (Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management
                (IAM) role to use.\s
                """;

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

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
```

```
String roleArn = args[4];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

SqsClient sqs = SqsClient.builder()
    .region(Region.US_EAST_1)
    .build();

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

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

public static void startLabels(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vid0b = Video.builder()
            .s3Object(s3obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
        StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vid0b)
            .minConfidence(50F)
            .build();
```

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

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

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

                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                        ans = false;
                else
                        System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
                yy++;
        }

        System.out.println(startJobId + " status is: " + status);

    } catch (RekognitionException | InterruptedException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
```

```
messages = sqs.receiveMessage(messageRequest).messages();

if (!messages.isEmpty()) {
    for (Message message : messages) {
        String notification = message.body();

        // Get the status and job id from the notification
        ObjectMapper mapper = new ObjectMapper();
        JsonNode jsonMessageTree = mapper.readTree(notification);
        JsonNode messageBodyText = jsonMessageTree.get("Message");
        ObjectMapper operationResultMapper = new ObjectMapper();
        JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
        JsonNode operationJobId = jsonResultTree.get("JobId");
        JsonNode operationStatus = jsonResultTree.get("Status");
        System.out.println("Job found in JSON is " + operationJobId);

        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
            .queueUrl(queueUrl)
            .build();

        String jobId = operationJobId.textValue();
        if (startJobId.compareTo(jobId) == 0) {
            System.out.println("Job id: " + operationJobId);
            System.out.println("Status : " +
operationStatus.toString());

            if (operationStatus.asText().equals("SUCCEEDED"))
                getResultsLabels(rekClient);
            else
                System.out.println("Video analysis failed");

            sqs.deleteMessage(deleteMessageRequest);
        } else {
            System.out.println("Job received was not job " +
startJobId);

            sqs.deleteMessage(deleteMessageRequest);
        }
    }
}

} catch (RekognitionException e) {
    e.getMessage();
}
```

```
        System.exit(1);
    } catch (JsonMappingException e) {
        e.printStackTrace();
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }
}

// Gets the job results by calling GetLabelDetection
private static void getResultsLabels(RekognitionClient rekClient) {

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

    try {
        do {
            if (labelDetectionResult != null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
                .maxResults(maxResults)
                .nextToken(paginationToken)
                .build();

            labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
            VideoMetadata videoMetaData =
labelDetectionResult.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " +
videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());

            List<LabelDetection> detectedLabels =
labelDetectionResult.labels();
            for (LabelDetection detectedLabel : detectedLabels) {
                long seconds = detectedLabel.timestamp();
                Label label = detectedLabel.label();
                System.out.println("Millisecond: " + seconds + " ");
            }
        } while (labelDetectionResult.nextToken() != null);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
        System.out.println("    Label:" + label.name());
        System.out.println("    Confidence:" +
detectedLabel.label().confidence().toString());

        List<Instance> instances = label.instances();
        System.out.println("    Instances of " + label.name());

        if (instances.isEmpty()) {
            System.out.println("        " + "None");
        } else {
            for (Instance instance : instances) {
instance.confidence().toString());
                System.out.println("        Bounding box: " +
instance.boundingBox().toString());
            }
        }
        System.out.println("    Parent labels for " + label.name() +
":"");

        List<Parent> parents = label.parents();

        if (parents.isEmpty()) {
            System.out.println("        None");
        } else {
            for (Parent parent : parents) {
                System.out.println("        " + parent.name());
            }
        }
        System.out.println();
    }
    } while (labelDetectionResult != null &&
labelDetectionResult.nextToken() != null);

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


Mendeteksi konten yang tidak pantas atau menyinggung dalam video yang disimpan di bucket Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import
    software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

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

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
                (for example, (for example, myBucket).\s
```

```
        video - The name of video (for example, people.mp4).\s
        topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
        roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
        """;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

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

    startModerationDetection(rekClient, channel, bucket, video);
    getModResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

public static void startModerationDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {

    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
```

```
        .s3Object(s3Obj)
        .build();

        StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
        .jobTag("Moderation")
        .notificationChannel(channel)
        .video(vidObj)
        .build();

        StartContentModerationResponse startModDetectionResult = rekClient
        .startContentModeration(modDetectionRequest);
        startJobId = startModDetectionResult.jobId();

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

public static void getModResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetContentModerationResponse modDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (modDetectionResponse != null)
                paginationToken = modDetectionResponse.nextToken();

            GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                modDetectionResponse =
rekClient.getContentModeration(modRequest);
                status = modDetectionResponse.jobStatusAsString();
            }
        }
    }
}
```

```
        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is
    null.
    VideoMetadata videoMetaData =
modDetectionResponse.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " +
videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
    for (ContentModerationDetection mod : mods) {
        long seconds = mod.timestamp() / 1000;
        System.out.print("Mod label: " + seconds + " ");
        System.out.println(mod.moderationLabel().toString());
        System.out.println();
    }

    } while (modDetectionResponse != null &&
modDetectionResponse.nextToken() != null);

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

Mendeteksi segmen isyarat teknis dan segmen deteksi bidikan dalam video yang disimpan dalam bucket Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
    software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import
    software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
import java.util.List;

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

```
public static void main(String[] args) {
    final String usage = ""

        Usage:    <bucket> <video> <topicArn> <roleArn>

        Where:
            bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
            video - The name of video (for example, people.mp4).\s
            topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
            roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
        """;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];

    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    SqsClient sqs = SqsClient.builder()
        .region(Region.US_EAST_1)
        .build();

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

    startSegmentDetection(rekClient, channel, bucket, video);
    getSegmentResults(rekClient);
    System.out.println("This example is done!");
    sqs.close();
    rekClient.close();
}
```

```
}

public static void startSegmentDetection(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vid0b = Video.builder()
            .s3Object(s3obj)
            .build();

        StartShotDetectionFilter cueDetectionFilter =
        StartShotDetectionFilter.builder()
            .minSegmentConfidence(60F)
            .build();

        StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
        StartTechnicalCueDetectionFilter.builder()
            .minSegmentConfidence(60F)
            .build();

        StartSegmentDetectionFilters filters =
        StartSegmentDetectionFilters.builder()
            .shotFilter(cueDetectionFilter)
            .technicalCueFilter(technicalCueDetectionFilter)
            .build();

        StartSegmentDetectionRequest segDetectionRequest =
        StartSegmentDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .segmentTypes(SegmentType.TECHNICAL_CUE, SegmentType.SHOT)
            .video(vid0b)
            .filters(filters)
            .build();

        StartSegmentDetectionResponse segDetectionResponse =
        rekClient.startSegmentDetection(segDetectionRequest);
        startJobId = segDetectionResponse.jobId();
    }
}
```

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

public static void getSegmentResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetSegmentDetectionResponse segDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (segDetectionResponse != null)
                paginationToken = segDetectionResponse.nextToken();

            GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
                status = segDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }
            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is
null.

```



```
        List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
        for (VideoMetadata metaData : videoMetaData) {
            System.out.println("Format: " + metaData.format());
            System.out.println("Codec: " + metaData.codec());
            System.out.println("Duration: " + metaData.durationMillis());
            System.out.println("FrameRate: " + metaData.frameRate());
            System.out.println("Job");
        }

        List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
        for (SegmentDetection detectedSegment : detectedSegments) {
            String type = detectedSegment.type().toString();
            if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
                System.out.println("Technical Cue");
                TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
                System.out.println("\tType: " + segmentCue.type());
                System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
            }

            if (type.contains(SegmentType.SHOT.toString())) {
                System.out.println("Shot");
                ShotSegment segmentShot = detectedSegment.shotSegment();
                System.out.println("\tIndex " + segmentShot.index());
                System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
            }

            long seconds = detectedSegment.durationMillis();
            System.out.println("\tDuration : " + seconds + "
milliseconds");
            System.out.println("\tStart time code: " +
detectedSegment.startTimecodeSMPTE());
            System.out.println("\tEnd time code: " +
detectedSegment.endTimecodeSMPTE());
            System.out.println("\tDuration time code: " +
detectedSegment.durationSMPTE());
            System.out.println();
        }
    }
```

```
        } while (segDetectionResponse != null &&
segDetectionResponse.nextToken() != null);

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

Mendeteksi teks dalam video yang disimpan dalam video yang disimpan dalam bucket Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

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

```
public static void main(String[] args) {
    final String usage = ""

        Usage:    <bucket> <video> <topicArn> <roleArn>

        Where:
            bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
            video - The name of video (for example, people.mp4).\s
            topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
            roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
        """;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];

    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startTextLabels(rekClient, channel, bucket, video);
    getTextResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

public static void startTextLabels(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
```

```
        String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vid0b = Video.builder()
            .s3object(s3obj)
            .build();

        StartTextDetectionRequest labelDetectionRequest =
        StartTextDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vid0b)
            .build();

        StartTextDetectionResponse labelDetectionResponse =
        rekClient.startTextDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void getTextResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetTextDetectionResponse textDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (textDetectionResponse != null)
                paginationToken = textDetectionResponse.nextToken();

            GetTextDetectionRequest recognitionRequest =
            GetTextDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
```

```
        .maxResults(10)
        .build();

    // Wait until the job succeeds.
    while (!finished) {
        textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
        status = textDetectionResponse.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is
null.

    VideoMetadata videoMetaData =
textDetectionResponse.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " +
videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<TextDetectionResult> labels =
textDetectionResponse.textDetections();
    for (TextDetectionResult detectedText : labels) {
        System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
        System.out.println("Id : " +
detectedText.textDetection().id());
        System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
        System.out.println("Type: " +
detectedText.textDetection().type());
        System.out.println("Text: " +
detectedText.textDetection().detectedText());
```

```
        System.out.println();
    }

    } while (textDetectionResponse != null &&
textDetectionResponse.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

Mendeteksi orang dalam video yang disimpan dalam video yang disimpan dalam bucket Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import
    software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import
    software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
*/
public class VideoPersonDetection {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        NotificationChannel channel = NotificationChannel.builder()
            .snsTopicArn(topicArn)
            .roleArn(roleArn)
            .build();

        startPersonLabels(rekClient, channel, bucket, video);
        getPersonDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }
}
```

```
public static void startPersonLabels(RekognitionClient rekClient,
    NotificationChannel channel,
    String bucket,
    String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vid0b = Video.builder()
            .s3object(s3obj)
            .build();

        StartPersonTrackingRequest personTrackingRequest =
        StartPersonTrackingRequest.builder()
            .jobTag("DetectingLabels")
            .video(vid0b)
            .notificationChannel(channel)
            .build();

        StartPersonTrackingResponse labelDetectionResponse =
        rekClient.startPersonTracking(personTrackingRequest);
        startJobId = labelDetectionResponse.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void getPersonDetectionResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetPersonTrackingResponse personTrackingResult = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (personTrackingResult != null)
                paginationToken = personTrackingResult.nextToken();
        }
    }
}
```



```
        GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
        .jobId(startJobId)
        .nextToken(paginationToken)
        .maxResults(10)
        .build();

        // Wait until the job succeeds
        while (!finished) {

                personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                status = personTrackingResult.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                }
                yy++;
        }

        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is
null.

        VideoMetadata videoMetaData =
personTrackingResult.videoMetadata();

        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " +
videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());
        System.out.println("Job");

        List<PersonDetection> detectedPersons =
personTrackingResult.persons();
        for (PersonDetection detectedPerson : detectedPersons) {
                long seconds = detectedPerson.timestamp() / 1000;
                System.out.print("Sec: " + seconds + " ");
                System.out.println("Person Identifier: " +
detectedPerson.person().index());
        }
    }
}
```

```
        System.out.println();
    }

    } while (personTrackingResult != null &&
personTrackingResult.nextToken() != null);

    } catch (RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
 - [GetCelebrityPengakuan](#)
 - [GetContentModerasi](#)
 - [GetLabelDeteksi](#)
 - [GetPersonMelacak](#)
 - [GetSegmentDeteksi](#)
 - [GetTextDeteksi](#)
 - [StartCelebrityPengakuan](#)
 - [StartContentModerasi](#)
 - [StartLabelDeteksi](#)
 - [StartPersonMelacak](#)
 - [StartSegmentDeteksi](#)
 - [StartTextDeteksi](#)

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [AWS Repositori Contoh Kode](#).

Mendeteksi wajah dalam video yang disimpan dalam bucket Amazon S3.

```
suspend fun startFaceDetection(
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
    val s3obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidObj =
        Video {
            s3Object = s3obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vidObj
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val startLabelDetectionResult = rekClient.startFaceDetection(request)
        startJobId = startLabelDetectionResult.jobId.toString()
    }
}

suspend fun getFaceResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var response: GetFaceDetectionResponse? = null

        val recognitionRequest =
            GetFaceDetectionRequest {
                jobId = startJobId
                maxResults = 10
            }
    }
```

```

// Wait until the job succeeds.
while (!finished) {
    response = rekClient.getFaceDetection(recognitionRequest)
    status = response.jobStatus.toString()
    if (status.compareTo("SUCCEEDED") == 0) {
        finished = true
    } else {
        println("$yy status is: $status")
        delay(1000)
    }
    yy++
}

// Proceed when the job is done - otherwise VideoMetadata is null.
val videoMetaData = response?.videoMetadata
println("Format: ${videoMetaData?.format}")
println("Codec: ${videoMetaData?.codec}")
println("Duration: ${videoMetaData?.durationMillis}")
println("FrameRate: ${videoMetaData?.frameRate}")

// Show face information.
response?.faces?.forEach { face ->
    println("Age: ${face.face?.ageRange}")
    println("Face: ${face.face?.beard}")
    println("Eye glasses: ${face?.face?.eyeglasses}")
    println("Mustache: ${face.face?.mustache}")
    println("Smile: ${face.face?.smile}")
}
}
}

```

Mendeteksi konten yang tidak pantas atau menyinggung dalam video yang disimpan di bucket Amazon S3.

```

suspend fun startModerationDetection(
    channel: NotificationChannel?,
    bucketVal: String?,
    videoVal: String?,
) {
    val s3obj =
        S3object {
            bucket = bucketVal

```

```
        name = videoVal
    }
    val vidObj =
        Video {
            s3Object = s3Obj
        }
    val request =
        StartContentModerationRequest {
            jobTag = "Moderation"
            notificationChannel = channel
            video = vidObj
        }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val startModDetectionResult = rekClient.startContentModeration(request)
        startJobId = startModDetectionResult.jobId.toString()
    }
}

suspend fun getModResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var modDetectionResponse: GetContentModerationResponse? = null

        val modRequest =
            GetContentModerationRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            modDetectionResponse = rekClient.getContentModeration(modRequest)
            status = modDetectionResponse.jobStatus.toString()
            if (status.compareTo("SUCCEEDED") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }
    }
}
```

```
// Proceed when the job is done - otherwise VideoMetadata is null.
val videoMetaData = modDetectionResponse?.videoMetadata
println("Format: ${videoMetaData?.format}")
println("Codec: ${videoMetaData?.codec}")
println("Duration: ${videoMetaData?.durationMillis}")
println("FrameRate: ${videoMetaData?.frameRate}")

modDetectionResponse?.moderationLabels?.forEach { mod ->
    val seconds: Long = mod.timestamp / 1000
    print("Mod label: $seconds ")
    println(mod.moderationLabel)
}
}
```

- Untuk detail API, lihat topik berikut di referensi API SDK untuk Kotlin AWS .
 - [GetCelebrityPengakuan](#)
 - [GetContentModerasi](#)
 - [GetLabelDeteksi](#)
 - [GetPersonMelacak](#)
 - [GetSegmentDeteksi](#)
 - [GetTextDeteksi](#)
 - [StartCelebrityPengakuan](#)
 - [StartContentModerasi](#)
 - [StartLabelDeteksi](#)
 - [StartPersonMelacak](#)
 - [StartSegmentDeteksi](#)
 - [StartTextDeteksi](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Contoh lintas layanan untuk Amazon Rekognition menggunakan SDK AWS

Contoh aplikasi berikut menggunakan AWS SDK untuk menggabungkan Amazon Layanan AWS Rekognition dengan yang lain. Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan petunjuk tentang cara mengatur dan menjalankan aplikasi.

Contoh

- [Membuat aplikasi manajemen aset foto yang memungkinkan pengguna mengelola foto menggunakan label](#)
- [Mendeteksi APD dalam gambar dengan Amazon AWS Rekognition menggunakan SDK](#)
- [Mendeteksi wajah dalam gambar menggunakan AWS SDK](#)
- [Mendeteksi objek dalam gambar dengan Amazon Rekognition menggunakan SDK AWS](#)
- [Mendeteksi orang dan objek dalam video dengan Amazon Rekognition menggunakan SDK AWS](#)
- [Simpan EXIF dan informasi gambar lainnya menggunakan SDK AWS](#)

Membuat aplikasi manajemen aset foto yang memungkinkan pengguna mengelola foto menggunakan label

Contoh kode berikut ini menunjukkan cara membuat aplikasi nirserver yang memungkinkan pengguna mengelola foto menggunakan label.

.NET

AWS SDK for .NET

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK untuk C++

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK untuk Java 2.x

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK untuk JavaScript (v3)

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK untuk Kotlin

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK untuk PHP

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK untuk Rust

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mendeteksi APD dalam gambar dengan Amazon AWS Rekognition menggunakan SDK

Contoh kode berikut menunjukkan cara membuat aplikasi yang menggunakan Amazon Rekognition untuk mendeteksi Alat Pelindung Diri (APD) dalam gambar.

Java

SDK untuk Java 2.x

Menunjukkan cara membuat AWS Lambda fungsi yang mendeteksi gambar dengan Alat Pelindung Diri.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK untuk JavaScript (v3)

Menunjukkan cara menggunakan Amazon Rekognition dengan AWS SDK for JavaScript membuat aplikasi untuk mendeteksi alat pelindung diri (APD) pada gambar yang terletak di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi tersebut menyimpan hasilnya ke tabel Amazon DynamoDB, dan mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Pelajari cara:

- Membuat pengguna yang tidak diautentikasi menggunakan Amazon Cognito.
- Menganalisis gambar untuk APD menggunakan Amazon Rekognition.
- Memverifikasi alamat email untuk Amazon SES.
- Memperbarui tabel DynamoDB dengan hasil.
- Mengirim notifikasi email menggunakan Amazon SES.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- DynamoDB

- Amazon Rekognition
- Amazon S3
- Amazon SES

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mendeteksi wajah dalam gambar menggunakan AWS SDK

Contoh kode berikut ini menunjukkan cara:

- Menyimpan gambar di bucket Amazon S3.
- Menggunakan Amazon Rekognition untuk mendeteksi detail wajah, seperti rentang usia, jenis kelamin, dan emosi (seperti tersenyum).
- Menampilkan detail tersebut.

Rust

SDK untuk Rust

Menyimpan gambar di bucket Amazon S3 dengan prefiks unggahan, menggunakan Amazon Rekognition untuk mendeteksi detail wajah, seperti rentang usia, jenis kelamin, dan emosi (tersenyum, dll.), dan menampilkan detail tersebut.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mendeteksi objek dalam gambar dengan Amazon Rekognition menggunakan SDK AWS

Contoh kode berikut menunjukkan cara membuat aplikasi yang menggunakan Amazon Rekognition untuk mendeteksi objek berdasarkan kategori dalam gambar.

.NET

AWS SDK for .NET

Menunjukkan cara menggunakan Amazon Rekognition .NET API untuk membuat aplikasi yang menggunakan Amazon Rekognition untuk mengidentifikasi objek berdasarkan kategori dalam gambar yang berada di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi ini mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK untuk Java 2.x

Menunjukkan cara menggunakan Amazon Rekognition Java API untuk membuat aplikasi yang menggunakan Amazon Rekognition untuk mengidentifikasi objek berdasarkan kategori dalam gambar yang terletak di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi ini mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition

- Amazon S3
- Amazon SES

JavaScript

SDK untuk JavaScript (v3)

Menunjukkan cara menggunakan Amazon Rekognition dengan membuat aplikasi AWS SDK for JavaScript yang menggunakan Amazon Rekognition untuk mengidentifikasi objek berdasarkan kategori dalam gambar yang terletak di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi ini mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Pelajari cara:

- Membuat pengguna yang tidak diautentikasi menggunakan Amazon Cognito.
- Menganalisis gambar untuk objek menggunakan Amazon Rekognition.
- Memverifikasi alamat email untuk Amazon SES.
- Mengirim notifikasi email menggunakan Amazon SES.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK untuk Kotlin

Menunjukkan cara menggunakan Amazon Rekognition Kotlin API untuk membuat aplikasi yang menggunakan Amazon Rekognition untuk mengidentifikasi objek berdasarkan kategori dalam gambar yang berada di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi ini mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK untuk Python (Boto3)

Menunjukkan cara menggunakan AWS SDK for Python (Boto3) untuk membuat aplikasi web yang memungkinkan Anda melakukan hal berikut:

- Mengunggah foto ke bucket Amazon Simple Storage Service (Amazon S3).
- Menggunakan Amazon Rekognition untuk menganalisis dan memberi label pada foto.
- Menggunakan Amazon Simple Email Service (Amazon SES) untuk mengirim laporan email analisis gambar.

Contoh ini berisi dua komponen utama: halaman web yang ditulis di dalamnya JavaScript yang dibangun dengan React, dan layanan REST yang ditulis dengan Python yang dibangun dengan Flask-RESTful.

Anda dapat menggunakan halaman web React untuk:

- Menampilkan daftar gambar yang disimpan di bucket S3 Anda.
- Mengunggah gambar dari komputer ke bucket S3.
- Menampilkan gambar dan label yang mengidentifikasi item yang terdeteksi dalam gambar.
- Mendapatkan laporan semua gambar di bucket S3 Anda dan mengirimkan email laporan tersebut.

Halaman web memanggil layanan REST. Layanan mengirimkan permintaan ke AWS untuk melakukan tindakan berikut:

- Mendapatkan dan memfilter daftar gambar dalam bucket S3 Anda.
- Mengunggah foto ke bucket S3 Anda.

- Menggunakan Amazon Rekognition untuk menganalisis foto individual dan mendapatkan daftar label yang mengidentifikasi item yang terdeteksi dalam foto.
- Menganalisis semua foto di bucket S3 Anda dan menggunakan Amazon SES untuk mengirim laporan melalui email.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mendeteksi orang dan objek dalam video dengan Amazon Rekognition menggunakan SDK AWS

Contoh kode berikut menunjukkan cara mendeteksi orang dan objek dalam video dengan Amazon Rekognition.

Java

SDK untuk Java 2.x

Menunjukkan cara menggunakan Amazon Rekognition Java API untuk membuat aplikasi guna mendeteksi wajah dan objek di video yang berada di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi ini mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition

- Amazon S3
- Amazon SES

JavaScript

SDK untuk JavaScript (v3)

Menunjukkan cara menggunakan Amazon Rekognition dengan AWS SDK for JavaScript membuat aplikasi untuk mendeteksi wajah dan objek dalam video yang terletak di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi ini mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Pelajari cara:

- Membuat pengguna yang tidak diautentikasi menggunakan Amazon Cognito.
- Menganalisis gambar untuk APD menggunakan Amazon Rekognition.
- Memverifikasi alamat email untuk Amazon SES.
- Mengirim notifikasi email menggunakan Amazon SES.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK untuk Python (Boto3)

Gunakan Amazon Rekognition untuk mendeteksi wajah, objek, dan orang dalam video dengan memulai tugas deteksi asinkron. Contoh ini juga mengonfigurasi Amazon Rekognition untuk memberi tahu topik Amazon Simple Notification Service (Amazon SNS) saat pekerjaan selesai dan berlangganan antrian Amazon Simple Queue Service (Amazon SQS) ke topik tersebut. Ketika antrian menerima pesan tentang pekerjaan, pekerjaan diambil dan hasilnya adalah output.

Contoh ini paling baik dilihat di GitHub. Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon SNS
- Amazon SQS

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Simpan EXIF dan informasi gambar lainnya menggunakan SDK AWS

Contoh kode berikut ini menunjukkan cara:

- Mendapatkan informasi EXIF dari file JPG, JPEG, atau PNG.
- Mengunggah file gambar ke bucket Amazon S3.
- Menggunakan Amazon Rekognition untuk mengidentifikasi tiga atribut teratas (label) dalam file.
- Menambahkan informasi EXIF dan label ke tabel Amazon DynamoDB di Wilayah.

Rust

SDK untuk Rust

Mendapatkan informasi EXIF dari file JPG, JPEG, atau PNG, mengunggah file gambar ke bucket Amazon S3, menggunakan Amazon Rekognition untuk mengidentifikasi tiga atribut teratas (label di Amazon Rekognition) dalam file, dan menambahkan EXIF dan informasi label ke tabel Amazon DynamoDB di Wilayah.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- DynamoDB
- Amazon Rekognition

- Amazon S3

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Menggunakan Rekognition dengan SDK AWS](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Referensi API

Referensi Amazon Rekognition API sekarang berada di [Referensi API Amazon Rekognition](#).

Keamanan Amazon Rekognition

Keamanan cloud di AWS merupakan prioritas tertinggi. Sebagai pelanggan AWS, Anda mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Gunakan topik berikut untuk mempelajari cara mengamankan sumber daya Amazon Rekognition Anda.

Topik

- [Identity and Access Management untuk Amazon Rekognition](#)
- [Perlindungan data di Amazon Rekognition](#)
- [Menggunakan Amazon Rekognition dengan Amazon VPC endpoint](#)
- [Validasi Kepatuhan Amazon Rekognition](#)
- [Ketahanan di Amazon Rekognition](#)
- [Konfigurasi dan analisis kelemahan di Amazon Rekognition](#)
- [Pencegahan wakil bingung lintas layanan](#)
- [Keamanan Infrastruktur di Amazon Rekognition](#)

Identity and Access Management untuk Amazon Rekognition

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol yang dapat diautentikasi (masuk) dan diotorisasi (mendapatkan izin) untuk menggunakan sumber daya Amazon Rekognition. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- [Audiens](#)
- [Mengautentikasi dengan identitas](#)
- [Mengelola akses menggunakan kebijakan](#)
- [Cara kerja Amazon Rekognition dengan IAM](#)
- [AWS kebijakan terkelola untuk Amazon Rekognition](#)
- [Contoh kebijakan berbasis identitas Amazon Rekognition.](#)

- [Contoh kebijakan berbasis sumber daya Amazon Rekognition](#)
- [Pemecahan masalah identitas dan akses Amazon Rekognition](#)

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Amazon Rekognition.

Pengguna layanan – Jika Anda menggunakan layanan Amazon Rekognition untuk melakukan tugas, administrator Anda akan memberikan kredensial dan izin yang dibutuhkan. Saat Anda menggunakan lebih banyak fitur Amazon Rekognition untuk melakukan pekerjaan, Anda mungkin memerlukan izin tambahan. Memahami cara pengelolaan akses dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di Amazon Rekognition, lihat [Pemecahan masalah identitas dan akses Amazon Rekognition](#).

Administrator layanan – Jika Anda bertanggung jawab atas sumber daya Amazon Rekognition di perusahaan Anda, maka Anda mungkin memiliki akses penuh ke Amazon Rekognition. Tugas Anda adalah menentukan fitur dan sumber daya Amazon Rekognition mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasar IAM. Untuk mempelajari selengkapnya cara perusahaan Anda dapat menggunakan IAM dengan Amazon Rekognition, lihat [Cara kerja Amazon Rekognition dengan IAM](#).

Administrator IAM – Jika Anda adalah administrator IAM, Anda mungkin ingin belajar dengan lebih detail cara menulis kebijakan untuk mengelola akses ke Amazon Rekognition. Untuk melihat contoh kebijakan berbasis identitas Amazon Rekognition yang dapat Anda gunakan di IAM, lihat [Contoh kebijakan berbasis identitas Amazon Rekognition..](#)

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensial identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensial yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensi Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya

menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat [Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS](#).

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensial Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat [Menandatangani permintaan AWS API](#) di Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat [Autentikasi multi-faktor](#) dalam Panduan Pengguna AWS IAM Identity Center dan [Menggunakan autentikasi multi-faktor \(MFA\) dalam AWS](#) dalam Panduan Pengguna IAM.

Pengguna dan grup IAM

[Pengguna IAM](#) adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial sementara, bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat [Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang](#) dalam Panduan Pengguna IAM.

[Grup IAM](#) adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin ke grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara.

Untuk mempelajari selengkapnya, lihat [Kapan harus membuat pengguna IAM \(bukan peran\)](#) dalam Panduan Pengguna IAM.

Peran IAM

[Peran IAM](#) adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara AWS Management Console dengan [beralih peran](#). Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat [Menggunakan peran IAM](#) dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat [Membuat peran untuk Penyedia Identitas pihak ketiga](#) dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat [Set izin](#) dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM](#).
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Sebagai contoh, ketika Anda memanggil suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.
 - Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna IAM atau peran untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan

beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat [Sesi akses maju](#).

- Peran layanan – Peran layanan adalah [peran IAM](#) yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat [Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS](#) dalam Panduan pengguna IAM.
- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensi sementara untuk aplikasi yang berjalan pada instans EC2 dan membuat atau permintaan API. AWS CLI AWS Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan AWS peran ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat [Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan dalam instans Amazon EC2](#) dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat [Kapan harus membuat peran IAM \(bukan pengguna\)](#) dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk

informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat [Gambaran umum kebijakan JSON](#) dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasinya. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

Menggunakan kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke identitas, seperti pengguna IAM, grup pengguna, atau peran. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat [Membuat kebijakan IAM](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam Akun AWS. Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan inline, lihat [Memilih antara kebijakan terkelola dan kebijakan inline](#) dalam Panduan Pengguna IAM.

Menggunakan kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus

[menentukan prinsipal](#) dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, lihat [Gambaran umum daftar kontrol akses \(ACL\)](#) dalam Panduan Developer Amazon Simple Storage Service.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM (pengguna IAM atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batasan izin, lihat [Batasan izin untuk entitas IAM](#) dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) — SCP adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur di organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat [Cara kerja SCP](#) dalam Panduan Pengguna AWS Organizations .
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau

peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya, lihat [Kebijakan sesi](#) dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat [Logika evaluasi kebijakan](#) di Panduan Pengguna IAM.

Cara kerja Amazon Rekognition dengan IAM

Sebelum menggunakan IAM untuk mengelola akses ke Amazon Rekognition, Anda harus memahami fitur IAM mana yang tersedia untuk digunakan dengan Amazon Rekognition. Untuk mendapatkan tampilan tingkat tinggi tentang cara Amazon Rekognition AWS dan layanan lainnya bekerja dengan IAM [AWS](#), lihat [Layanan yang Bekerja dengan IAM](#) di Panduan Pengguna IAM.

Topik

- [Kebijakan berbasis identitas Amazon Rekognition](#)
- [Kebijakan berbasis sumber daya Amazon Rekognition](#)
- [IAM role Amazon Rekognition](#)

Kebijakan berbasis identitas Amazon Rekognition

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak, serta kondisi tempat tindakan tersebut diperbolehkan atau ditolak. Amazon Rekognition mendukung tindakan, sumber daya, dan kunci syarat tertentu. Untuk mempelajari semua elemen yang Anda gunakan dalam kebijakan JSON, lihat [Referensi Elemen kebijakan IAM JSON](#) dalam Panduan Pengguna IAM.

Tindakan

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki

nama yang sama dengan operasi AWS API terkait. Ada beberapa pengecualian, misalnya tindakan hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Sertakan tindakan dalam kebijakan untuk memberikan izin guna melakukan operasi terkait.

Tindakan kebijakan di Amazon Rekognition menggunakan prefiks berikut sebelum tindakan: `rekognition:`. Misalnya, untuk memberikan seseorang izin untuk mendeteksi objek, adegan, atau konsep dalam citra dengan Amazon Rekognition Operasi API `DetectLabels`, Anda menyertakan tindakan `rekognition:DetectLabels` dalam kebijakan mereka. Pernyataan kebijakan harus memuat elemen `Action` atau `NotAction`. Amazon Rekognition menentukan set tindakannya sendiri yang menjelaskan tugas yang dapat Anda lakukan dengan layanan ini.

Untuk menetapkan beberapa tindakan dalam satu pernyataan, pisahkan dengan koma seperti berikut:

```
"Action": [  
    "rekognition:action1",  
    "rekognition:action2"
```

Anda dapat menentukan beberapa tindakan menggunakan wildcard (*). Misalnya, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut:

```
"Action": "rekognition:Describe*"
```

Untuk melihat daftar tindakan Amazon Rekognition, lihat [Tindakan yang Ditetapkan oleh Amazon Rekognition](#) di Panduan Pengguna IAM.

Sumber daya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan sumber daya menggunakan [Amazon Resource Name \(ARN\)](#). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk informasi selengkapnya tentang format ARN, lihat [Nama Sumber Daya Amazon \(ARN\) dan Ruang Nama AWS Layanan](#).

Misalnya, untuk menentukan kluster MyCollection dalam pernyataan Anda, gunakan ARN berikut:

```
"Resource": "arn:aws:rekognition:us-east-1:123456789012:collection/MyCollection"
```

Untuk menentukan semua instans DB milik akun tertentu, gunakan wildcard (*):

```
"Resource": "arn:aws:rekognition:us-east-1:123456789012:collection/*"
```

Beberapa tindakan Amazon Rekognition, seperti membuat sumber daya, tidak dapat dilakukan pada sumber daya tertentu. Dalam kondisi tersebut, Anda harus menggunakan wildcard (*).

```
"Resource": "*"
```

Untuk melihat daftar tipe sumber daya Amazon Rekognition dan ARN mereka, lihat [Sumber Daya Ditetapkan oleh Amazon Rekognition](#) di Panduan Pengguna IAM. Untuk mempelajari tindakan yang dapat menentukan ARN setiap sumber daya, lihat [Tindakan yang Ditetapkan oleh Amazon Rekognition](#).

Kunci syarat

Amazon Rekognition tidak menyediakan kunci syarat khusus layanan, tetapi mendukung menggunakan beberapa kunci syarat global. Untuk melihat semua kunci kondisi AWS global, lihat [Kunci Konteks Kondisi AWS Global](#) di Panduan Pengguna IAM.

Kebijakan berbasis sumber daya Amazon Rekognition

Amazon Rekognition mendukung kebijakan berbasis sumber daya untuk operasi penyalinan model Label Kustom. Untuk informasi selengkapnya, lihat contoh [kebijakan berbasis sumber daya Amazon Rekognition](#).

Layanan lain, seperti Amazon S3, juga mendukung kebijakan izin berbasis sumber daya. Misalnya, Anda dapat melampirkan kebijakan ke bucket S3 untuk mengelola izin akses ke bucket tersebut.

Untuk mengakses citra yang disimpan dalam bucket Amazon S3, Anda harus memiliki izin untuk mengakses objek dalam bucket S3. Dengan izin ini, Amazon Rekognition dapat mengunduh citra dari bucket S3. Contoh kebijakan izin berikut memungkinkan pengguna melakukan tindakan `s3:GetObject` pada bucket S3 dengan nama `Tests3bucket`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": [
        "arn:aws:s3:::Tests3bucket/*"
      ]
    }
  ]
}
```

Untuk menggunakan bucket S3 dengan versioning diaktifkan, tambahkan tindakan `s3:GetObjectVersion`, seperti yang ditunjukkan dalam contoh berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::Tests3bucket/*"
      ]
    }
  ]
}
```


IAM role Amazon Rekognition

[Peran IAM](#) adalah entitas dalam AWS akun Anda yang memiliki izin tertentu.

Menggunakan kredensial sementara dengan Amazon Rekognition

Anda dapat menggunakan kredensial sementara untuk masuk dengan gabungan, menjalankan IAM role, atau menjalankan peran lintas akun. [Anda memperoleh kredensial keamanan sementara dengan memanggil operasi AWS STS API seperti AssumeRole atau GetFederation Token.](#)

Amazon Rekognition mendukung penggunaan kredensial sementara.

Peran terkait layanan

[Peran terkait AWS layanan](#) memungkinkan layanan mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran terkait layanan muncul di akun IAM Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Amazon Rekognition tidak mendukung peran terkait layanan.

Peran layanan

Fitur ini memungkinkan layanan untuk menerima [peran layanan](#) atas nama Anda. Peran ini mengizinkan layanan untuk mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran layanan muncul di akun IAM Anda dan dimiliki oleh akun tersebut. Ini berarti administrator IAM dapat mengubah izin untuk peran ini. Namun, melakukannya dapat merusak fungsionalitas layanan.

Amazon Rekognition mendukung peran layanan.

Menggunakan peran layanan dapat menimbulkan masalah keamanan di mana Amazon Rekognition digunakan untuk memanggil layanan lain dan menindaklanjuti sumber daya yang seharusnya tidak dapat diakses. Untuk menjaga keamanan akun Anda, Anda harus membatasi ruang lingkup akses Amazon Rekognition hanya ke sumber daya yang Anda gunakan. Ini dapat dilakukan dengan melampirkan kebijakan kepercayaan ke peran layanan IAM Anda. Untuk informasi tentang cara melakukannya, lihat [Pencegahan wakil bingung lintas layanan](#).

Memilih IAM role di Amazon Rekognition

Saat Anda mengonfigurasi Amazon Rekognition untuk menganalisis video yang tersimpan, Anda harus memilih peran untuk mengizinkan Amazon Rekognition mengakses Amazon SNS atas nama

Anda. Jika sebelumnya Anda telah membuat peran layanan atau peran terkait layanan, maka Amazon Rekognition akan memberi Anda daftar peran untuk dipilih. Untuk informasi selengkapnya, lihat [the section called “Mengonfigurasi Amazon Rekognition Video”](#).

AWS kebijakan terkelola untuk Amazon Rekognition

Untuk menambahkan izin ke pengguna, grup, dan peran, lebih mudah menggunakan kebijakan AWS terkelola daripada menulis kebijakan sendiri. Dibutuhkan waktu dan keahlian untuk [membuat kebijakan yang dikelola pelanggan IAM](#) yang hanya memberi tim Anda izin yang mereka butuhkan. Untuk memulai dengan cepat, Anda dapat menggunakan kebijakan AWS terkelola kami. Kebijakan ini mencakup kasus penggunaan umum dan tersedia di AWS akun Anda. Untuk informasi selengkapnya tentang kebijakan AWS [AWS terkelola](#), lihat [kebijakan terkelola](#) di Panduan Pengguna IAM.

AWS layanan memelihara dan memperbarui kebijakan AWS terkelola. Anda tidak dapat mengubah izin dalam kebijakan AWS terkelola. Layanan terkadang menambahkan izin tambahan ke kebijakan yang dikelola AWS untuk mendukung fitur-fitur baru. Jenis pembaruan ini akan memengaruhi semua identitas (pengguna, grup, dan peran) di mana kebijakan tersebut dilampirkan. Layanan kemungkinan besar akan memperbarui kebijakan yang dikelola AWS saat ada fitur baru yang diluncurkan atau saat ada operasi baru yang tersedia. Layanan tidak menghapus izin dari kebijakan AWS terkelola, sehingga pembaruan kebijakan tidak akan merusak izin yang ada.

Selain itu, AWS mendukung kebijakan terkelola untuk fungsi pekerjaan yang mencakup beberapa layanan. Misalnya, kebijakan AWS terkelola `ReadOnlyAccess` menyediakan akses hanya-baca ke semua AWS layanan dan sumber daya. Saat layanan meluncurkan fitur baru, AWS menambahkan izin hanya-baca untuk operasi dan sumber daya baru. Untuk melihat daftar dan deskripsi dari kebijakan fungsi tugas, lihat [kebijakan yang dikelola AWS untuk fungsi tugas](#) di Panduan Pengguna IAM.

Kebijakan terkelola AWS: `AmazonRekognitionFullAccess`

`AmazonRekognitionFullAccess` memberikan akses penuh ke sumber daya Amazon Rekognition termasuk membuat dan menghapus koleksi.

Anda dapat melampirkan kebijakan `AmazonRekognitionFullAccess` ke identitas IAM Anda.

Detail izin

Kebijakan ini mencakup izin berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "rekognition:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Kebijakan terkelola AWS: AmazonRekognitionReadOnlyAccess

AmazonRekognitionReadOnlyAccess memberikan akses hanya-baca ke sumber daya Amazon Rekognition.

Anda dapat melampirkan kebijakan AmazonRekognitionReadOnlyAccess ke identitas IAM Anda.

Detail izin

Kebijakan ini mencakup izin berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AmazonRekognitionReadOnlyAccess",
      "Effect": "Allow",
      "Action": [
        "rekognition:CompareFaces",
        "rekognition:DetectFaces",
        "rekognition:DetectLabels",
        "rekognition:ListCollections",
        "rekognition:ListFaces",
        "rekognition:SearchFaces",
        "rekognition:SearchFacesByImage",

```

```
        "rekognition:DetectText",
        "rekognition:GetCelebrityInfo",
        "rekognition:RecognizeCelebrities",
        "rekognition:DetectModerationLabels",
        "rekognition:GetLabelDetection",
        "rekognition:GetFaceDetection",
        "rekognition:GetContentModeration",
        "rekognition:GetPersonTracking",
        "rekognition:GetCelebrityRecognition",
        "rekognition:GetFaceSearch",
        "rekognition:GetTextDetection",
        "rekognition:GetSegmentDetection",
        "rekognition:DescribeStreamProcessor",
        "rekognition:ListStreamProcessors",
        "rekognition:DescribeProjects",
        "rekognition:DescribeProjectVersions",
        "rekognition:DetectCustomLabels",
        "rekognition:DetectProtectiveEquipment",
        "rekognition:ListTagsForResource",
        "rekognition:ListDatasetEntries",
        "rekognition:ListDatasetLabels",
        "rekognition:DescribeDataset",
        "rekognition:ListProjectPolicies",
        "rekognition:ListUsers",
        "rekognition:SearchUsers",
        "rekognition:SearchUsersByImage",
        "rekognition:GetMediaAnalysisJob",
        "rekognition:ListMediaAnalysisJobs"
    ],
    "Resource": "*"
}
]
```

Kebijakan terkelola AWS: AmazonRekognitionServiceRole

AmazonRekognitionServiceRole memungkinkan Amazon Rekognition untuk memanggil Amazon Kinesis Data Streams dan layanan Amazon SNS atas nama Anda.

Anda dapat melampirkan kebijakan AmazonRekognitionServiceRole ke identitas IAM Anda.

Jika menggunakan peran layanan ini, Anda harus menjaga keamanan akun Anda dengan membatasi ruang lingkup akses Amazon Rekognition hanya ke sumber daya yang Anda gunakan. Ini dapat

dilakukan dengan melampirkan kebijakan kepercayaan ke peran layanan IAM Anda. Untuk informasi tentang cara melakukannya, lihat [Pencegahan wakil bingung lintas layanan](#).

Detail izin

Kebijakan ini mencakup izin berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "arn:aws:sns:*:*:AmazonRekognition*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kinesis:PutRecord",
        "kinesis:PutRecords"
      ],
      "Resource": "arn:aws:kinesis:*:*:stream/AmazonRekognition*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:GetDataEndpoint",
        "kinesisvideo:GetMedia"
      ],
      "Resource": "*"
    }
  ]
}
```

Kebijakan terkelola AWS: AmazonRekognitionCustomLabelsFullAccess

Kebijakan ini untuk Label Kustom Amazon Rekognition; pengguna. Gunakan AmazonRekognitionCustomLabelsFullAccess kebijakan ini untuk mengizinkan pengguna mengakses penuh ke API Label Kustom Rekognition Amazon dan akses penuh ke bucket konsol yang dibuat oleh konsol Label Kustom Rekognition Amazon.

Detail izin

Kebijakan ini mencakup izin berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:ListAllMyBuckets",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersion",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3::*custom-labels*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "rekognition:CopyProjectVersion",
        "rekognition:CreateProject",
        "rekognition:CreateProjectVersion",
        "rekognition:StartProjectVersion",
        "rekognition:StopProjectVersion",
        "rekognition:DescribeProjects",
        "rekognition:DescribeProjectVersions",
        "rekognition:DetectCustomLabels",
        "rekognition>DeleteProject",
        "rekognition>DeleteProjectVersion",
        "rekognition:TagResource",
        "rekognition:UntagResource",
        "rekognition:ListTagsForResource",
        "rekognition:CreateDataset",
        "rekognition:ListDatasetEntries",
        "rekognition:ListDatasetLabels",
        "rekognition:DescribeDataset",
        "rekognition:UpdateDatasetEntries",
        "rekognition:DistributeDatasetEntries",

```

```

        "rekognition:DeleteDataset",
        "rekognition:PutProjectPolicy",
        "rekognition:ListProjectPolicies",
        "rekognition:DeleteProjectPolicy"
    ],
    "Resource": "*"
}
]
}

```

Amazon Rekognition memperbarui kebijakan terkelola AWS

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk Amazon Rekognition sejak layanan ini mulai melacak perubahan ini. Untuk pemberitahuan otomatis tentang perubahan pada halaman ini, harap berlangganan umpan RSS pada halaman riwayat Dokumen Amazon Rekognition halaman Sejarah dokumen.

Perubahan	Deskripsi	Tanggal
<p>Tindakan yang melibatkan pekerjaan analisis media telah ditambahkan ke kebijakan terkelola berikut:</p> <ul style="list-style-type: none"> • Kebijakan terkelola AWS: AmazonRekognitionReadOnlyAccess 	<p>Amazon Rekognition menambahkan tindakan berikut ke kebijakan terkelola : AmazonRekognitionReadOnlyAccess</p> <ul style="list-style-type: none"> • GetMediaAnalysisJob • ListMediaAnalysisJob 	31 Oktober 2023
<p>Tindakan yang melibatkan pengelolaan pengguna telah ditambahkan ke kebijakan terkelola berikut:</p> <ul style="list-style-type: none"> • Kebijakan terkelola AWS: AmazonRekognitionReadOnlyAccess 	<p>Amazon Rekognition menambahkan tindakan berikut ke kebijakan terkelola : AmazonRekognitionReadOnlyAccess</p> <ul style="list-style-type: none"> • ListUsers • SearchUsers 	12 Juni 2023

Perubahan	Deskripsi	Tanggal
	<ul style="list-style-type: none"> • SearchUsersByImage 	
<p>Tindakan untuk ProjectPolicy dan Salinan Model Label Kustom telah ditambahkan ke kebijakan terkelola berikut:</p> <ul style="list-style-type: none"> • Kebijakan terkelola AWS: AmazonRekognitionFullAccess • Kebijakan terkelola AWS: AmazonRekognitionCustomLabelsFullAccess 	<p>Amazon Rekognition menambahkan tindakan berikut ke dan kebijakan terkelolaAmazonRekognitionCustomLabelsFullAccess :</p> <p>AmazonRekognitionFullAccess</p> <ul style="list-style-type: none"> • CopyProjectVersion • PutProjectPolicy • ListProjectPolicies • DeleteProjectPolicy 	21 Juli 2022
<p>Tindakan untuk ProjectPolicy dan Salinan Model Label Kustom telah ditambahkan ke kebijakan terkelola berikut:</p> <ul style="list-style-type: none"> • Kebijakan terkelola AWS: AmazonRekognitionReadOnlyAccess 	<p>Amazon Rekognition menambahkan tindakan berikut ke kebijakan terkelola : AmazonRekognitionReadOnlyAccess</p> <ul style="list-style-type: none"> • ListProjectPolicies 	21 Juli 2022

Perubahan	Deskripsi	Tanggal
<p>Pembaruan manajemen kumpulan data untuk kebijakan terkelola berikut:</p> <ul style="list-style-type: none"> • Kebijakan terkelola AWS: AmazonRekognitionReadOnlyAccess • Kebijakan terkelola AWS: AmazonRekognitionFullAccess • Kebijakan terkelola AWS: AmazonRekognitionCustomLabelsFullAccess 	<p>Amazon Rekognition menambahkan tindakan berikut ke,, dan kebijakan AmazonRekognitionReadOnlyAccess AmazonRekognitionFullOnlyAccess terkelola AmazonRekognitionCustomLabelsFullAccess</p> <ul style="list-style-type: none"> • CreateDataset • ListDatasetEntries • ListDatasetLabels • DescribeDataset • UpdateDatasetEntries • DistributeDatasetEntries • DeleteDataset 	November 1, 2021
<p>Memberi tag pembaruan untuk Kebijakan terkelola AWS: AmazonRekognitionReadOnlyAccess dan Kebijakan terkelola AWS: AmazonRekognitionFullAccess</p>	<p>Amazon Rekognition menambahkan tindakan penandaan baru ke kebijakan dan kebijakan . AmazonRekognitionFullAccess AmazonRekognitionReadOnlyAccess</p>	2 April 2021
<p>Amazon Rekognition mulai melacak perubahan</p>	<p>Amazon Rekognition mulai melacak perubahan untuk AWS kebijakan yang dikelola.</p>	2 April 2021

Contoh kebijakan berbasis identitas Amazon Rekognition.

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya Amazon Rekognition. Mereka juga tidak dapat melakukan tugas menggunakan AWS Management Console, AWS CLI, atau AWS API. Administrator IAM harus membuat kebijakan IAM yang memberikan izin kepada pengguna dan peran untuk melakukan operasi API tertentu pada sumber daya yang diperlukan. Administrator kemudian harus melampirkan kebijakan tersebut ke pengguna atau grup yang memerlukan izin tersebut.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat [Membuat Kebijakan pada Tab JSON](#) dalam Panduan Pengguna IAM.

Topik

- [Praktik terbaik kebijakan](#)
- [Menggunakan konsol Amazon Rekognition](#)
- [Contoh kebijakan Amazon Rekognition Custom Labels](#)
- [Contoh 1: Mengizinkan akses baca ke semua sumber daya](#)
- [Contoh 2: Izinkan pengguna mengakses sumber daya secara penuh](#)
- [Izinkan pengguna untuk melihat izin mereka sendiri](#)

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Amazon Rekognition di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat [Kebijakan yang dikelola AWS](#) atau [Kebijakan yang dikelola AWS untuk fungsi tugas](#) dalam Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya

dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat [Kebijakan dan izin dalam IAM](#) dalam Panduan Pengguna IAM.

- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat [Elemen kebijakan JSON IAM: Kondisi](#) dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat [Validasi kebijakan IAM Access Analyzer](#) dalam Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat [Mengonfigurasi akses API yang dilindungi MFA](#) dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat [Praktik terbaik keamanan dalam IAM](#) dalam Panduan Pengguna IAM.

Menggunakan konsol Amazon Rekognition

Dengan pengecualian fitur Label Kustom Amazon Rekognition, Amazon Rekognition tidak memerlukan izin tambahan saat menggunakan konsol Amazon Rekognition. Untuk informasi tentang Label Kustom Amazon Rekognition, lihat [Langkah 5: Izin Konsol Label Kustom Mengatur Amazon Rekognition](#).

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau AWS API. Sebagai alternatif, hanya izinkan akses ke tindakan yang cocok dengan operasi API yang sedang Anda coba lakukan.

Contoh kebijakan Amazon Rekognition Custom Labels

Anda dapat membuat kebijakan berbasis identitas untuk Label Kustom Amazon Rekognition. Untuk informasi selengkapnya, lihat [Keamanan](#).

Contoh 1: Mengizinkan akses baca ke semua sumber daya

Contoh berikut memberikan akses hanya baca ke sumber daya Amazon Rekognition.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "rekognition:CompareFaces",
        "rekognition:DetectFaces",
        "rekognition:DetectLabels",
        "rekognition:ListCollections",
        "rekognition:ListFaces",
        "rekognition:SearchFaces",
        "rekognition:SearchFacesByImage",
        "rekognition:DetectText",
        "rekognition:GetCelebrityInfo",
        "rekognition:RecognizeCelebrities",
        "rekognition:DetectModerationLabels",
        "rekognition:GetLabelDetection",
        "rekognition:GetFaceDetection",
        "rekognition:GetContentModeration",
        "rekognition:GetPersonTracking",
        "rekognition:GetCelebrityRecognition",
        "rekognition:GetFaceSearch",
        "rekognition:GetTextDetection",
        "rekognition:GetSegmentDetection",
        "rekognition:DescribeStreamProcessor",
        "rekognition:ListStreamProcessors",
        "rekognition:DescribeProjects",
        "rekognition:DescribeProjectVersions",
        "rekognition:DetectCustomLabels",
        "rekognition:DetectProtectiveEquipment",
        "rekognition:ListTagsForResource",
        "rekognition:ListDatasetEntries",
        "rekognition:ListDatasetLabels",

```

```

        "rekognition:DescribeDataset"
    ],
    "Resource": "*"
}
]
}

```

Contoh 2: Izinkan pengguna mengakses sumber daya secara penuh

Contoh berikut memberikan akses penuh ke sumber daya Amazon Rekognition.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "rekognition:*"
      ],
      "Resource": "*"
    }
  ]
}

```

Izinkan pengguna untuk melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. AWS CLI AWS

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "iam:ListAttachedUserPolicies",

```

```

        "iam:ListUserPolicies",
        "iam:GetUser"
    ],
    "Resource": ["arn:aws:iam::*:user/${aws:username}"]
},
{
    "Sid": "NavigateInConsole",
    "Effect": "Allow",
    "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
    ],
    "Resource": "*"
}
]
}

```

Contoh kebijakan berbasis sumber daya Amazon Rekognition

Amazon Rekognition Custom Labels menggunakan kebijakan berbasis sumber daya, yang dikenal sebagai kebijakan proyek, untuk mengelola izin penyalinan untuk versi model.

Kebijakan proyek memberikan atau menolak izin untuk menyalin versi model dari proyek sumber ke proyek tujuan. Anda memerlukan kebijakan proyek jika proyek tujuan berada di AWS akun yang berbeda atau jika Anda ingin membatasi akses dalam AWS akun, Misalnya, Anda mungkin ingin menolak izin salin ke peran IAM tertentu. Untuk informasi selengkapnya, lihat [Menyalin model](#).

Memberikan izin untuk menyalin versi model

Contoh berikut memungkinkan kepala sekolah `arn:aws:iam::123456789012:role/Admin` untuk menyalin versi model `arn:aws:rekognition:us-east-1:123456789012:project/my_project/version/test_1/1627045542080`.

```

{
    "Version": "2012-10-17",
    "Statement": [

```

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::123456789012:role/Admin"
  },
  "Action": "rekognition:CopyProjectVersion",
  "Resource": "arn:aws:rekognition:us-east-1:123456789012:project/my_project/
version/test_1/1627045542080"
}
]
```

Pemecahan masalah identitas dan akses Amazon Rekognition

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan mengatasi masalah umum yang mungkin Anda temui saat bekerja menggunakan Amazon Rekognition dan IAM.

Topik

- [Saya tidak memiliki otorisasi untuk mengambil tindakan di Amazon Rekognition](#)
- [Saya tidak berwenang untuk melakukan iam: PassRole](#)
- [Saya adalah administrator dan ingin mengizinkan orang lain mengakses Amazon Rekognition](#)
- [Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses sumber daya Rekognition Amazon saya](#)

Saya tidak memiliki otorisasi untuk mengambil tindakan di Amazon Rekognition

Jika AWS Management Console memberitahu Anda bahwa Anda tidak berwenang untuk melakukan tindakan, maka Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu *widget*, tetapi tidak memiliki izin `rekognition:GetWidget`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
rekognition:GetWidget on resource: my-example-widget
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya agar dia dapat mengakses *my-example-widget* menggunakan `rekognition:GetWidget` tindakan.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan bahwa Anda tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Amazon Rekognition.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk lulus peran ke layanan.

Contoh kesalahan berikut terjadi saat pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Amazon Rekognition. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya adalah administrator dan ingin mengizinkan orang lain mengakses Amazon Rekognition

Untuk mengizinkan orang lain mengakses Amazon Rekognition, Anda harus membuat entitas IAM (pengguna atau peran) untuk orang atau aplikasi yang memerlukan akses. Mereka akan menggunakan kredensial bagi entitas tersebut untuk mengakses AWS. Anda kemudian harus melampirkan kebijakan pada entitas yang memberi mereka izin yang benar di Amazon Rekognition.

Untuk langsung memulai, lihat [Membuat pengguna dan grup yang didelegasikan IAM untuk pertama kalinya](#) dalam Panduan Pengguna IAM.

Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses sumber daya Rekognition Amazon saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang

dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mengetahui apakah Amazon Rekognition mendukung fitur ini, lihat [Cara kerja Amazon Rekognition dengan IAM](#).
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat [Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki](#) di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat [Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat [Menyediakan akses ke pengguna terautentikasi eksternal \(federasi identitas\)](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM](#).

Perlindungan data di Amazon Rekognition

[model tanggung jawab bersama](#) AWS diterapkan untuk perlindungan data di Amazon Rekognition. Sebagaimana diuraikan dalam model ini, AWS bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk memelihara kendali atas isi yang dihost pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam [Pertanyaan Umum Privasi Data](#). Lihat informasi tentang perlindungan data di Eropa di pos blog [Model Tanggung Jawab Bersama dan GDPR AWS](#) di Blog Keamanan AWS.

Untuk tujuan perlindungan data, sebaiknya lindungi kredensial Akun AWS dan siapkan untuk masing-masing pengguna AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.

- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya AWS. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pengelogan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi enkripsi AWS, bersama semua kontrol keamanan bawaan dalam Layanan AWS.
- Gunakan layanan keamanan terkelola lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 ketika mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi yang lebih lengkap tentang titik akhir FIPS yang tersedia di [Standar Pemrosesan Informasi Federal \(FIPS\) 140-2](#).

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan Rekognition atau Layanan AWS lainnya menggunakan konsol, APIAWS CLI, atau SDK. AWS Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Enkripsi data

Informasi berikut menjelaskan lokasi Amazon Rekognition saat menggunakan enkripsi data untuk melindungi data Anda.

Enkripsi saat tidak digunakan

Amazon Rekognition Image

Citra

Gambar yang diteruskan ke operasi Amazon Rekognition API dapat disimpan dan digunakan untuk meningkatkan layanan kecuali Anda telah memilih keluar dengan mengunjungi halaman [kebijakan opt-out layanan AI](#) dan mengikuti proses yang dijelaskan di sana. Citra yang disimpan dienkripsi saat tidak digunakan (Amazon S3) menggunakan AWS Key Management Service (SSE-KMS).

Koleksi

Untuk operasi perbandingan wajah yang menyimpan informasi dalam koleksi, algoritme deteksi utama tersebut pertama-tama mendeteksi wajah dalam citra input, mengekstraksi vektor untuk setiap

wajah, lalu menyimpan vektor wajah dalam koleksi. Amazon Rekognition menggunakan vektor wajah ini saat melakukan performa perbandingan wajah. Vektor wajah disimpan sebagai susunan pelampung dan dienkripsi saat istirahat.

Amazon Rekognition Video

Video

Untuk menganalisis video, Amazon Rekognition menyalin video Anda ke layanan untuk diproses. Video dapat disimpan dan digunakan untuk meningkatkan layanan kecuali Anda telah memilih keluar dengan mengunjungi [halaman kebijakan opt-out layanan AI](#) dan mengikuti proses yang dijelaskan di sana. Video dienkripsi saat tidak digunakan (Amazon S3) menggunakan AWS Key Management Service (SSE-KMS).

Label Kustom Amazon Rekognition

Label Kustom Amazon Rekognition mengenkripsi data at rest Anda.

Citra

Untuk melatih model Anda, Label Kustom Amazon Rekognition membuat salinan pelatihan sumber dan tes citra Anda. Gambar yang disalin dienkripsi saat istirahat di Amazon Simple Storage Service (S3) Amazon Simple Storage Service (S3) menggunakan enkripsi sisi server dengan AWS KMS key yang Anda berikan atau kunci KMS yang dimiliki. AWS Amazon Rekognition Custom Labels hanya mendukung kunci KMS simetris. Citra sumber Anda tidak terpengaruh. Untuk informasi selengkapnya, lihat [Melatih Model Label Kustom Amazon Rekognition](#).

Model

Secara default, Amazon Rekognition Custom Labels mengenkripsi model terlatih dan file manifest yang disimpan di bucket Amazon S3 menggunakan enkripsi sisi server dengan file. Kunci milik AWS Untuk informasi selengkapnya, lihat [Melindungi Data Menggunakan Enkripsi Sisi Server](#). Hasil pelatihan ditulis ke bucket yang ditentukan dalam parameter input `OutputConfig` ke [CreateProjectVersion](#). Hasil pelatihan dienkripsi menggunakan pengaturan enkripsi yang dikonfigurasi untuk bucket (`OutputConfig`).

Bucket konsol

Konsol Label Kustom Amazon Rekognition menciptakan bucket Amazon S3 (bucket konsol) yang dapat Anda gunakan untuk mengelola proyek Anda. Bucket konsol tersebut dienkripsi

menggunakan enkripsi Amazon S3 default. Untuk informasi selengkapnya, lihat [Enkripsi default Layanan Penyimpanan Sederhana Amazon untuk bucket S3](#). Jika Anda menggunakan kunci KMS Anda sendiri, konfigurasi bucket konsol tersebut setelah dibuat. Untuk informasi selengkapnya, lihat [Melindungi Data Menggunakan Enkripsi Sisi Server](#). Label Kustom Amazon Rekognition memblokir akses publik ke bucket konsol tersebut.

Rekognition Face Liveness

Semua data terkait sesi yang disimpan di akun layanan Rekognition Face Liveness sepenuhnya dienkripsi saat istirahat. Secara default, gambar referensi dan audit dienkripsi menggunakan kunci yang AWS dimiliki di akun layanan. Namun, Anda dapat memilih untuk memberikan AWS KMS kunci Anda sendiri untuk mengenkripsi gambar-gambar ini.

Enkripsi dalam bergerak

Titik akhir API Amazon Rekognition hanya mendukung koneksi aman melalui HTTPS. Semua komunikasi dienkripsi dengan Keamanan Lapisan Pengangkutan (TLS).

Manajemen kunci

Anda dapat menggunakan AWS Key Management Service (KMS) untuk mengelola kunci untuk citra input dan video yang Anda simpan di bucket Amazon S3. Untuk informasi selengkapnya, lihat [Konsep AWS Key Management Service](#).

Enkripsi Kunci Dikelola Pelanggan untuk Keaktifan Wajah

[CreateFaceLivenessSession](#) API mengambil `KmsKeyId` parameter opsional. Anda dapat memberikan kunci KMS yang telah Anda buat di akun Anda. `id` Kunci ini akan digunakan untuk mengenkripsi referensi dan mengaudit gambar yang diperoleh selama [StartFaceLivenessSession](#) API, dan selama [GetFaceLivenessSessionResults](#) API gambar akan didekripsi menggunakan kunci ini sebelum mengembalikan hasilnya. Jika `CreateFaceLivenessSession` permintaan menyertakan `OutputConfig`, gambar referensi dan audit akan diunggah ke jalur Amazon S3 yang ditentukan. Sebaiknya aktifkan Enkripsi Sisi Server ([SSE-S3](#)) di bucket Amazon S3 Anda sehingga data tetap terenkripsi saat istirahat.

Saat Anda memberikan ID AWS KMS kunci Anda sendiri, layanan Rekognition Face Liveness mendapat izin untuk menggunakan kunci yang dikelola pelanggan atas nama prinsipal yang memanggil API. Prinsipal (pengguna atau peran) yang digunakan untuk memanggil API dari backend pelanggan (API `CreateFaceLivenessSession` dan `GetFaceLivenessSessionResults`) harus memiliki akses untuk melakukan hal berikut:

- km: DescribeKey
- km: GenerateDataKey
- kms:Decrypt

Privasi lalu lintas jaringan internet

Titik akhir Amazon Virtual Private Cloud (Amazon VPC) untuk Amazon Rekognition adalah entitas logis dalam VPC yang mengizinkan konektivitas hanya ke Amazon Rekognition. Amazon VPC merutekan permintaan ke Amazon Rekognition dan merutekan respons kembali ke VPC. Untuk informasi selengkapnya, lihat [VPC Endpoints](#) dalam Panduan Pengguna Amazon VPC. Untuk informasi tentang menggunakan titik akhir Amazon VPC dengan Amazon Rekognition lihat [Menggunakan Amazon Rekognition dengan Amazon VPC endpoint](#).

Menggunakan Amazon Rekognition dengan Amazon VPC endpoint

Jika Anda menggunakan Amazon Virtual Private Cloud (Amazon VPC) untuk meng-host sumber daya AWS, Anda dapat membuat koneksi privat antara VPC Anda dan Amazon Rekognition. Anda dapat menggunakan koneksi ini untuk mengaktifkan Amazon Rekognition untuk berkomunikasi dengan sumber daya Anda di VPC Anda tanpa melalui internet publik.

Amazon VPC adalah layanan AWS yang dapat Anda gunakan untuk meluncurkan sumber daya AWS dalam jaringan virtual yang Anda tentukan. Dengan VPC, Anda memiliki kendali terhadap pengaturan jaringan Anda, seperti rentang alamat IP, subnet, tabel rute, dan gateway jaringan. Dengan VPC endpoint, jaringan AWS menangani perutean antara VPC dan layanan AWS.

Untuk menghubungkan VPC Anda ke Amazon Rekognition, Anda menentukan antarmuka VPC endpoint untuk Amazon Rekognition. Antarmuka titik akhir adalah antarmuka jaringan elastis dengan alamat IP privat yang berfungsi sebagai titik masuk untuk lalu lintas ditujukan untuk layanan AWS yang didukung. Titik akhir memberikan konektivitas yang dapat diandalkan, dapat diskalakan ke Amazon Rekognition—dan itu tidak memerlukan gateway internet, instans terjemahan alamat jaringan (NAT), atau koneksi VPN. Untuk informasi selengkapnya, lihat [Apa yang dimaksud dengan Amazon VPC](#) dalam Panduan Pengguna Amazon VPC.

Endpoint VPC antarmuka diaktifkan oleh AWSPrivateLink. Teknologi AWS mengaktifkan komunikasi privat antara layanan AWS menggunakan antarmuka jaringan elastis dengan alamat IP privat.

Note

Semua titik akhir Amazon Rekognition Federal Information Processing Standard (FIPS) didukung oleh AWSPrivateLink.

Membuat titik akhir Amazon VPC untuk Amazon Rekognition

Anda dapat membuat dua tipe titik akhir Amazon VPC untuk digunakan dengan Amazon Rekognition.

- Sebuah VPC endpoint untuk digunakan dengan operasi Amazon Rekognition. Bagi sebagian besar pengguna, ini adalah tipe VPC endpoint yang paling sesuai.
- VPC endpoint untuk operasi Amazon Rekognition dengan titik akhir yang sesuai dengan standar pemerintah AS Publikasi Federal Information Processing Standard (FIPS) 140-2.

Untuk memulai menggunakan Amazon Rekognition dengan VPC Anda, gunakan konsol Amazon VPC untuk membuat antarmuka VPC endpoint untuk Amazon Rekognition. Untuk petunjuk, lihat prosedur "Untuk membuat antarmuka titik akhir untuk layanan AWS menggunakan konsol" di [Membuat Antarmuka Titik Akhir](#). Perhatikan langkah-langkah prosedur berikut:

- Langkah 3 — Untuk Kategori layanan, pilih Layanan AWS.
- Langkah 4 — Untuk Nama Layanan, pilih salah satu opsi berikut:
 - `com.amazonaws.region.rekognition`— Membuat VPC endpoint untuk operasi Amazon Rekognition.
 - `com.amazonaws.region.rekognition-fips` – Membuat VPC endpoint untuk operasi Amazon Rekognition dengan titik akhir yang sesuai dengan standar pemerintah AS Publikasi Federal Information Processing Standard (FIPS) 140-2.

Untuk informasi selengkapnya, lihat [Memulai](#) dalam Panduan Pengguna Amazon VPC.

Buat Kebijakan VPC endpoint untuk Amazon Rekognition

Anda dapat membuat kebijakan untuk Amazon VPC endpoint untuk Amazon Rekognition untuk menentukan hal berikut:

- Prinsipiel yang dapat melakukan tindakan.

- Tindakan yang dapat dilakukan.
- Sumber daya yang dapat digunakan untuk mengambil tindakan.

Untuk informasi selengkapnya, lihat [Mengontrol Akses ke Layanan dengan VPC Endpoint](#) dalam Panduan Pengguna Amazon VPC.

Kebijakan contoh berikut memungkinkan pengguna menghubungkan ke Amazon Rekognition melalui VPC endpoint untuk memanggil operasi API DetectFaces. Kebijakan ini mencegah pengguna melakukan operasi API Amazon Rekognition lainnya melalui VPC endpoint.

Pengguna masih dapat memanggil operasi API Amazon Rekognition lainnya dari luar VPC. Untuk informasi tentang cara menolak akses ke operasi API Amazon Rekognition yang berada di luar VPC, lihat [Kebijakan berbasis identitas Amazon Rekognition](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "rekognition:DetectFaces"
      ],
      "Resource": "*",
      "Effect": "Allow",
      "Principal": "*"
    }
  ]
}
```

Untuk mengubah kebijakan VPC endpoint untuk Amazon Rekognition

1. Buka konsol Amazon VPC di <https://console.aws.amazon.com/vpc/>.
2. Jika Anda belum membuat titik akhir untuk Amazon Rekognition, pilih Buat Titik Akhir. Kemudian pilih com.amazonaws.**Wilayah**.rekognition dan pilih Buat titik akhir.
3. Di panel navigasi, pilih Titik akhir.
4. Pilih titik akhir com.amazonaws.**Wilayah**.rekognition dan pilih tab Policy (Kebijakan) di bagian bawah layar.
5. Pilih Edit Kebijakan dan buat perubahan pada kebijakan.

Validasi Kepatuhan Amazon Rekognition

Auditor pihak ke tiga menilai keamanan dan kepatuhan Amazon Rekognition sebagai bagian dari beberapa program kepatuhan AWS. Mencakup SOC, PCI, FedRAMP, HIPAA, dan lainnya.

Untuk daftar layanan AWS dalam cakupan program kepatuhan spesifik, lihat [Layanan AWS dalam Cakupan berdasarkan Program Kepatuhan](#). Untuk informasi umum, lihat [Program Kepatuhan AWS](#).

Anda bisa mengunduh laporan audit pihak ke tiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat [Mengunduh Laporan di AWS Artifact](#).

Tanggung jawab kepatuhan Anda saat menggunakan Amazon Rekognition ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- [Panduan Quick Start Keamanan dan Kepatuhan](#) – Panduan deployment ini membahas pertimbangan arsitektur dan memberikan langkah untuk menerapkan lingkungan dasar yang berfokus pada keamanan dan kepatuhan di AWS.
- [Laporan Resmi Perancangan untuk Keamanan dan Kepatuhan HIPAA](#) – Laporan resmi ini akan menguraikan bagaimana perusahaan dapat menggunakan AWS untuk membuat aplikasi yang mematuhi HIPAA.
- [AWS Sumber Daya Kepatuhan](#) – Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- [AWS Config](#) – Layanan AWS ini menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- [AWS Security Hub](#) – Layanan AWS ini akan menyediakan tampilan komprehensif dari status keamanan Anda dalam AWS yang akan membantu Anda memeriksa kepatuhan Anda terhadap standar dan praktik terbaik industri.

Ketahanan di Amazon Rekognition

Infrastruktur global AWS dibangun di sekitar Wilayah AWS dan Availability Zone. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan jaringan berlatensi rendah, throughput yang tinggi, dan sangat redundan. Dengan Availability Zone, Anda dapat mendesain dan mengoperasikan aplikasi dan basis data yang secara otomatis mengalami kegagalan di antara zona tanpa gangguan. Availability Zone lebih tersedia, memiliki

toleransi kesalahan, dan dapat diskalakan dibandingkan dengan satu atau beberapa infrastruktur pusat data tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat [Infrastruktur Global AWS](#).

Selain infrastruktur global AWS, Amazon Rekognition memberi penawaran beberapa fitur untuk membantu mendukung ketahanan data dan kebutuhan cadangan Anda.

Konfigurasi dan analisis kelemahan di Amazon Rekognition

Konfigurasi dan kontrol IT merupakan tanggung jawab bersama antara AWS dan Anda, pelanggan kami. Untuk informasi selengkapnya, lihat [model tanggung jawab bersama](#) AWS.

Pencegahan wakil bingung lintas layanan

Dalam AWS, peniruan lintas layanan dapat terjadi ketika satu layanan (layanan panggilan) memanggil layanan lain (disebut layanan). Layanan panggilan dapat dimanipulasi untuk bertindak pada sumber daya pelanggan lain meskipun seharusnya tidak memiliki izin yang tepat, sehingga masalah wakil bingung.

Untuk mencegah hal ini, AWS menyediakan alat yang membantu Anda melindungi data Anda untuk semua layanan dengan prinsip-prinsip layanan yang telah diberikan akses ke sumber daya di akun Anda.

Sebaiknya gunakan [aws:SourceArn](#) dan [aws:SourceAccount](#) kunci konteks kondisi global dalam kebijakan sumber daya untuk membatasi izin yang diberikan Amazon Rekognition layanan lain ke sumber daya.

Jika nilai [aws:SourceArn](#) tidak berisi ID akun, seperti ARN bucket Amazon S3, Anda harus menggunakan kedua kunci untuk membatasi izin. Jika Anda menggunakan kedua tombol dan [aws:SourceArn](#) nilai berisi ID akun, [aws:SourceAccount](#) nilai dan akun di [aws:SourceArn](#) nilai harus menggunakan ID akun yang sama ketika digunakan dalam pernyataan kebijakan yang sama.

Gunakan [aws:SourceArn](#) jika Anda ingin hanya satu sumber daya yang terkait dengan akses lintas layanan. Gunakan [aws:SourceAccount](#) jika Anda ingin mengizinkan sumber daya apa pun di akun itu dikaitkan dengan penggunaan lintas-layanan.

Nilai [aws:SourceArn](#) harus ARN sumber daya yang digunakan oleh Rekognition, yang ditentukan dengan format berikut: `arn:aws:rekognition:region:account:resource`.

Nilai `arn:User ARN` harus ARN pengguna yang akan memanggil operasi analisis video (pengguna yang mengasumsikan peran).

Pendekatan yang disarankan untuk masalah wakil yang bingung adalah dengan menggunakan `aws:SourceArn` kunci konteks kondisi global dengan ARN sumber daya penuh.

Jika Anda tidak mengetahui ARN lengkap sumber daya atau jika Anda menentukan beberapa sumber daya, gunakan `aws:SourceArn` kunci dengan karakter wildcard (*) untuk bagian ARN yang tidak diketahui. Sebagai contoh, `arn:aws:rekognition:*:111122223333:*`.

Untuk melindungi terhadap masalah wakil yang bingung, lakukan langkah-langkah berikut:

1. Di panel navigasi konsol IAM pilih **Peran** pilihan. Konsol akan menampilkan peran untuk akun Anda saat ini.
2. Pilih nama peran yang ingin Anda modifikasi. Peran yang Anda modifikasi harus memiliki `AmazonRekognitionServiceRole` kebijakan perizinan. Pilih **Hubungan kepercayaan** tab.
3. Pilih **Mengedit** kebijakan kepercayaan.
4. Pada **Mengedit** kebijakan kepercayaan halaman, ganti kebijakan JSON default dengan kebijakan yang menggunakan salah satu atau kedua `aws:SourceArn` dan `aws:SourceAccount` kunci konteks kondisi global. Lihat contoh kebijakan berikut.
5. Pilih **Buat Kebijakan**.

Contoh berikut adalah kebijakan kepercayaan yang menunjukkan bagaimana Anda dapat menggunakan `aws:SourceArn` dan `aws:SourceAccount` kunci konteks kondisi global di Amazon Rekognition untuk mencegah masalah wakil yang membingungkan.

Jika Anda bekerja disimpan dan streaming video, Anda dapat menggunakan kebijakan seperti berikut dalam peran IAM Anda:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "rekognition.amazonaws.com",
        "AWS": "arn:User ARN"
      },
      "Action": "sts:AssumeRole",
    }
  ]
}
```

```

    "Condition":{
      "StringEquals":{
        "aws:SourceAccount":"Account ID"
      },
      "StringLike":{
        "aws:SourceArn":"arn:aws:rekognition:region:111122223333:streamprocessor/*"
      }
    }
  ]
}

```

Jika Anda bekerja secara eksklusif dengan video yang disimpan, Anda dapat menggunakan kebijakan seperti berikut dalam peran IAM Anda (perhatikan bahwa Anda tidak harus menyertakan `StringLike` argumen yang menentukan `streamprocessor`):

```

{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Principal":{
        "Service":"rekognition.amazonaws.com",
        "AWS":"arn:User ARN"
      },
      "Action":"sts:AssumeRole",
      "Condition":{
        "StringEquals":{
          "aws:SourceAccount":"Account ID"
        }
      }
    }
  ]
}

```

Keamanan Infrastruktur di Amazon Rekognition

Sebagai layanan terkelola, Amazon Rekognition dilindungi oleh AWS keamanan jaringan global. Untuk informasi tentang AWS layanan keamanan dan bagaimana AWS melindungi infrastruktur, lihat [AWS Keamanan Cloud](#). Untuk mendesain AWS lingkungan menggunakan praktik terbaik untuk

keamanan infrastruktur, lihat [Perlindungan Infrastruktur](#) di Pilar Keamanan AWS Kerangka Kerja yang Diarsiteksikan dengan Baik.

Anda menggunakan panggilan API yang dipublikasikan AWS untuk mengakses Amazon Rekognition melalui jaringan. Klien harus mendukung hal berikut:

- Transport Layer Security (TLS). Kami membutuhkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Suite cipher dengan kerahasiaan maju sempurna (PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan sistem yang lebih baru mendukung mode ini.

Selain itu, permintaan harus ditandatangani menggunakan access key ID dan secret access key yang terkait dengan principal IAM. Atau Anda bisa menggunakan [AWS Security Token Service](#) (AWS STS) untuk membuat kredensial keamanan sementara guna menandatangani permintaan.

Memantau Rekognition Amazon

Monitoring adalah bagian penting untuk menjaga keandalan, ketersediaan, dan kinerja Rekognition Amazon dan solusi AWS Anda yang lain. AWS menyediakan alat pemantauan berikut untuk memantau Rekognition, melaporkan ketika ada yang salah, dan mengambil tindakan otomatis ketika diperlukan:

- Amazon CloudWatch memonitor Anda AWS sumber daya dan dan aplikasi yang Anda jalankan di AWS dalam waktu nyata. Anda dapat mengumpulkan dan melacak metrik, membuat dasbor yang disesuaikan, dan mengatur alarm yang memberi tahu Anda atau mengambil tindakan saat metrik tertentu mencapai ambang batas yang ditentukan. Misalnya, Anda dapat memiliki CloudWatch melacak penggunaan CPU atau metrik instans Amazon EC2 dan meluncurkan instance baru ketika diperlukan. Untuk informasi selengkapnya, lihat [Amazon CloudWatch Panduan Pengguna](#).
- Amazon CloudWatch Log memungkinkan untuk memantau, menyimpan, dan mengakses file log dari instans Amazon EC2, CloudTrail, dan sumber-sumber lainnya. CloudWatch Log dapat memantau informasi dalam file log dan memberi tahu ketika ambang batas tertentu terpenuhi. Anda juga dapat mengarsipkan data log Anda dalam penyimpanan yang sangat tahan lama. Untuk informasi selengkapnya, lihat [Amazon CloudWatch Panduan Pengguna Log](#).
- Amazon EventBridge dapat digunakan untuk mengotomatisasi AWS layanan dan merespons secara otomatis untuk peristiwa sistem, seperti masalah ketersediaan aplikasi atau perubahan sumber daya. Acara dari AWS layanan dikirimkan ke EventBridge dalam waktu dekat. Anda dapat menuliskan aturan sederhana untuk menunjukkan peristiwa mana yang sesuai kepentingan Anda, dan tindakan otomatis mana yang diambil ketika suatu peristiwa sesuai dengan suatu aturan. Untuk informasi selengkapnya, lihat [Amazon EventBridge Panduan Pengguna](#).
- AWS CloudTrail merekam panggilan API dan peristiwa terkait yang dilakukan oleh atau atas nama akun AWS Anda dan mengirimkan berkas log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengidentifikasi pengguna dan akun mana yang memanggil AWS, alamat IP sumber yang melakukan panggilan, dan kapan panggilan tersebut terjadi. Untuk mengetahui informasi selengkapnya, lihat [Panduan Pengguna AWS CloudTrail](#).

Memantau Rekognition dengan Amazon CloudWatch

dengan CloudWatch, Anda bisa mendapatkan metrik untuk operasi Rekognition individual atau metrik Rekognition global untuk akun Anda, Anda dapat menggunakan metrik untuk melacak kondisi solusi berbasis Rekognition dan menyiapkan alarm untuk memberi tahu ketika satu atau beberapa metrik

berada di luar ambang batas yang ditentukan. Misalnya, Anda dapat melihat metrik untuk jumlah kesalahan server yang terjadi, atau metrik jumlah wajah yang terdeteksi. Anda juga dapat melihat metrik untuk berapa kali operasi Rekognition tertentu telah berhasil. Untuk melihat metrik, Anda dapat menggunakan [Amazon CloudWatch](#), [AmazonAWS Command Line Interface](#), atau [CloudWatch API](#).

Anda juga dapat melihat metrik agregat, untuk jangka waktu tertentu, dengan menggunakan konsol Rekognition. Untuk informasi selengkapnya, lihat [Latihan 4: Lihat metrik agregat \(konsol\)](#).

Menggunakan CloudWatch metrik untuk Rekognition

Untuk menggunakan metrik, Anda harus menentukan informasi berikut:

- Dimensi metrik, atau tanpa dimensi. Dimensi adalah pasangan nama-nilai yang membantu Anda mengidentifikasi metrik secara unik. Rekognition memiliki satu dimensi, bernama Operasi. Ini menyediakan metrik untuk operasi tertentu. Jika Anda tidak menentukan dimensi, metrik akan dicakup ke semua operasi Rekognition dalam akun Anda.
- Nama metrik, seperti `UserErrorCount`.

Anda bisa mendapatkan data pemantauan untuk Rekognition menggunakan AWS Management Console, AWS CLI, atau CloudWatch API. Anda juga dapat menggunakan CloudWatch API melalui salah satu Kit Pengembangan Perangkat Lunak (SDK) atau perangkat lunak Amazon AWS atau CloudWatch Alat API. Konsol menampilkan serangkaian grafik berdasarkan data mentah dari CloudWatch API. Tergantung kebutuhan, Anda mungkin lebih memilih menggunakan grafik yang ditampilkan di konsol atau diterima dari API.

Daftar berikut menunjukkan beberapa penggunaan umum untuk metrik. Berikut ini adalah saran untuk memulai, bukan daftar komprehensif.

Bagaimana caranya?	Metrik Terkait
Bagaimana cara melacak jumlah wajah yang dikenali?	Pantau statistik Sum metrik <code>DetectedFaceCount</code> .
Bagaimana saya tahu jika aplikasi saya telah mencapai jumlah maksimum permintaan per detik?	Pantau statistik Sum metrik <code>ThrottledCount</code> .

Bagaimana caranya?	Metrik Terkait
Bagaimana saya dapat memantau kesalahan permintaan?	Gunakan statistik Sum metrik <code>UserErrorCount</code> .
Bagaimana saya dapat menemukan jumlah total permintaan?	Gunakan statistik <code>ResponseTime</code> dan <code>Data Samples</code> dari metrik <code>ResponseTime</code> . Termasuk setiap permintaan yang menghasilkan kesalahan. Jika Anda ingin melihat panggilan operasi yang berhasil saja, maka gunakan metrik <code>SuccessfulRequestCount</code> .
Bagaimana saya bisa memantau latensi panggilan operasi Rekognition ?	Gunakan metrik <code>ResponseTime</code> .
Bagaimana saya dapat memantau berapa kali <code>IndexFaces</code> berhasil menambahkan wajah ke koleksi Rekognition?	Pantau statistik Sum dengan metrik <code>SuccessfulRequestCount</code> dan operasi <code>IndexFaces</code> . Gunakan dimensi <code>Operation</code> untuk memilih operasi dan metrik.

Anda harus memiliki yang sesuai CloudWatch izin untuk memantau Rekognition dengan CloudWatch. Untuk informasi lebih lanjut, lihat [Kontrol Autentikasi dan Akses untuk Amazon CloudWatch](#).

Akses metrik Rekognition

Contoh berikut menunjukkan cara mengakses metrik Rekognition menggunakan metrik Rekognition menggunakan metrik Rekognition menggunakan metrik Rekognition menggunakan CloudWatch konsol, yangAWS CLI, dan CloudWatchAPI.

Untuk melihat metrik (konsol)

1. Buka CloudWatch konsol di<https://console.aws.amazon.com/cloudwatch/>.
2. Pilih Metrik, pilih tab Semua Metrik, dan kemudian pilih Rekognition.
3. Pilih Metrik tanpa dimensi, dan kemudian pilih metrik.

Misalnya, pilih`DetectedFace`metrik untuk mengukur berapa banyak wajah yang terdeteksi.

4. Pilih nilai untuk rentang tanggal. Hitungan metrik ditampilkan dalam grafik.

Untuk melihat metrik, panggilan operasi **DetectFaces** yang berhasil telah dilakukan selama periode waktu tertentu (CLI).

- Buka AWS CLI dan masukkan perintah berikut:

```
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
```

Contoh ini menunjukkan panggilan operasi DetectFaces yang berhasil dilakukan selama periode waktu tertentu. Untuk informasi selengkapnya, lihat [get-metric-statistics](#).

Untuk mengakses metrik (CloudWatch API)

- Panggil [GetMetricStatistics](#). Untuk informasi selengkapnya, lihat [Amazon CloudWatch Referensi API](#).

Buat alarm

Anda dapat membuat sebuah CloudWatch alarm yang mengirimkan pesan Amazon Simple Nognition (Amazon SNS) ketika alarm berubah kondisi. Alarm mengawasi metrik tunggal selama periode waktu yang Anda tentukan, dan melakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas yang ditentukan selama periode waktu tertentu. Tindakan ini adalah notifikasi yang dikirim ke topik Amazon SNS atau kebijakan Auto Scaling.

Alarm memicu tindakan hanya untuk perubahan status berkelanjutan. CloudWatch alarm tidak menggunakan tindakan hanya karena berada dalam kondisi tertentu. Status harus diubah dan dipelihara selama jangka waktu tertentu.

Untuk mengatur alarm (konsol)

1. Masuk ke AWS Management Console dan membuka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Pilih Buat Alarm. Proses ini meluncurkan Wizard Buat Alarm.
3. Pada daftar metrik Metrik tanpa dimensi, pilih Metrik Rekognition, dan kemudian pilih metrik.

Misalnya, pilih `DetectedFaceCount` untuk menyiapkan alarm untuk jumlah wajah yang terdeteksi.

4. Di area Rentang Waktu, pilih nilai rentang tanggal yang mencakup operasi deteksi wajah yang telah Anda panggil. Pilih Selanjutnya
5. Isi Nama dan Deskripsi. Untuk Kapan pun, pilih `>=`, dan masukkan nilai maksimum pilihan Anda.
6. Jika Anda ingin CloudWatch untuk mengirim Anda email ketika status alarm tercapai, untuk Setiap kali alarm ini:, pilih Negara adalah ALARM. Untuk mengirim alarm ke topik Amazon SNS yang sudah ada, untuk Kirim notifikasi ke:, pilih topik SNS yang sudah ada. Untuk menyiapkan nama dan alamat email untuk daftar langganan email baru, pilih Buat topik CloudWatch menyimpan daftar dan menampilkannya di bidang sehingga Anda dapat menggunakannya untuk menyiapkan alarm di masa depan.

Note

Jika Anda menggunakan Buat topik untuk membuat topik Amazon SNS baru, maka alamat email harus diverifikasi sebelum penerima yang dituju menerima notifikasi. Amazon SNS hanya mengirim email ketika alarm memasuki status alarm. Jika perubahan status alarm ini terjadi sebelum alamat email diverifikasi, maka penerima yang dituju tidak akan menerima notifikasi.

7. Pratinjau alarm di bagian Pratinjau Alarm. Pilih Buat Alarm.

Untuk mengatur alarm (AWS CLI)

- Buka AWS CLI dan masukkan perintah berikut. Ubah nilai parameter `alarm-actions` untuk mereferensikan topik Amazon SNS yang sudah dibuat sebelumnya.

```
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
```

Contoh ini menunjukkan cara membuat alarm ketika lebih dari 10 kesalahan pengguna terjadi dalam waktu 5 menit. Untuk informasi selengkapnya, lihat [put-metric-alarm](#).

Untuk menyiapkan alarm (CloudWatch API)

- Panggil [PutMetricAlarm](#). Untuk informasi selengkapnya, lihat [Amazon CloudWatch Referensi API](#).

CloudWatchmetrik untuk Rekognition


Bagian ini berisi informasi tentang Amazon CloudWatch metrik danOperasidimensi yang tersedia untuk Rekognition Amazon.

Anda juga dapat melihat tampilan agregat metrik Rekognition dari konsol Rekognition. Untuk informasi selengkapnya, lihat [Latihan 4: Lihat metrik agregat \(konsol\)](#).

CloudWatch metrik untuk Rekognition

Tabel berikut merangkum metrik Rekognition.

Metrik	Deskripsi
SuccessfulRequestCount	<p>Jumlah permintaan yang berhasil. Kisaran kode respons untuk permintaan yang berhasil adalah 200 hingga 299.</p> <p>Unit: Jumlah</p> <p>Statistik valid: Sum, Average</p>
ThrottledCount	<p>Jumlah permintaan yang di-throttling. Rekognition men-throttling permintaan ketika menerima lebih banyak permintaan dari kuota transaksi per set detik untuk akun Anda. Jika batas yang ditetapkan untuk akun Anda sering terlampaui, Anda dapat meminta penambahan kuota. Untuk meminta penambahan, lihat Kuota Layanan AWS.</p> <p>Unit: Jumlah</p> <p>Statistik valid: Sum, Average</p>
ResponseTime	<p>Waktu dalam hitungan milidetik bagi Rekognition untuk mengomputasi respons.</p> <p>Unit:</p>

Metrik	Deskripsi
	<p>1. Jumlah untuk statistik Data Samples</p> <p>2. Milidetik untuk statistik Average</p> <p>Statistik valid: Data Samples, Average</p> <div data-bbox="456 457 1507 674" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p> Note</p> <p>Metrik ResponseTime tidak termasuk dalam panel metrik Rekognition.</p> </div>
DetectedFaceCount	<p>Jumlah wajah yang terdeteksi dengan operasi IndexFaces atau DetectFaces .</p> <p>Unit: Jumlah</p> <p>Statistik valid: Sum, Average</p>
DetectedLabelCount	<p>Jumlah label yang terdeteksi dengan operasi DetectLabels .</p> <p>Unit: Jumlah</p> <p>Statistik valid: Sum, Average</p>
ServerErrorCount	<p>Jumlah kesalahan server. Kisaran kode respons untuk kesalahan server adalah 500 hingga 599.</p> <p>Unit: Jumlah</p> <p>Statistik valid: Sum, Average</p>
UserErrorCount	<p>Jumlah kesalahan pengguna (parameter tidak valid, citra tidak valid, tidak ada izin, dll). Kisaran kode respons untuk kesalahan pengguna adalah 400 hingga 499.</p> <p>Unit: Jumlah</p> <p>Statistik valid: Sum, Average</p>

Metrik	Deskripsi
MinInferenceUnits	Jumlah minimum unit inferensi yang ditentukan selama <code>StartProjectVersion</code> permintaan. Unit: Jumlah Statistik valid: Average
MaxInferenceUnits	Jumlah unit inferensi maksimum yang ditentukan selama <code>StartProjectVersion</code> permintaan. Unit: Jumlah Statistik valid: Average
DesiredInferenceUnits	Jumlah unit inferensi yang diskalakan oleh Rekognition naik atau turun. Unit: Jumlah Statistik valid: Average
InServiceInferenceUnits	Jumlah unit inferensi yang digunakan model. Unit: Jumlah Statistik valid: Average Disarankan agar Anda menggunakan statistik Rata-rata untuk mendapatkan rata-rata 1 menit dari berapa banyak contoh yang digunakan.

CloudWatch metrik untuk Rekognition Streaming

Rekognition juga memiliki namespace kedua yang digunakan untuk operasi streaming, "Rekognition Streaming". Tabel berikut merangkum metrik Rekognition.

Metrik	Deskripsi
SuccessfulRequestCount	Jumlah permintaan yang berhasil. Kisaran kode respons untuk permintaan yang berhasil adalah 200 hingga 299.

Metrik	Deskripsi
	Unit: Jumlah Statistik valid: Sum, Average
CallCount	Jumlah operasi tertentu yang dilakukan di akun Anda. Statistik valid: Sum, Average
ThrottledCount	Jumlah permintaan yang di-throttling. Rekognition men-throttling permintaan ketika menerima lebih banyak permintaan dari kuota transaksi per set detik untuk akun Anda. Jika batas yang ditetapkan untuk akun Anda sering terlampaui, Anda dapat meminta penambahan kuota. Untuk meminta penambahan, lihat Kuota Layanan AWS . Unit: Jumlah Statistik valid: Sum, Average
ServerErrorCount	Jumlah kesalahan server. Kisaran kode respons untuk kesalahan server adalah 500 hingga 599. Unit: Jumlah Statistik valid: Sum, Average
UserErrorCount	Jumlah kesalahan pengguna (parameter tidak valid, citra tidak valid, tidak ada izin, dll). Kisaran kode respons untuk kesalahan pengguna adalah 400 hingga 499. Unit: Jumlah Statistik valid: Sum, Average

CloudWatch dimensi untuk Rekognition

Untuk mengambil metrik khusus operasi, gunakan namespace Rekognition dan tentukan dimensi operasi.

Untuk informasi selengkapnya tentang dimensi, lihat [Dimensi](#) di Amazon CloudWatch Panduan Pengguna.

CloudWatch dimensi untuk Rekognition Custom Rekognition

Tabel berikut menampilkan CloudWatch dimensi yang tersedia untuk digunakan dengan Label Kustom Rekognition:

Dimensi	Deskripsi
ProjectName	Nama proyek Label Kustom Rekognition yang Anda buat <code>CreateProject</code> .
VersionName	Nama versi proyek Label Kustom Rekognition yang Anda buat dengan <code>CreateProjectVersion</code> .

Untuk informasi selengkapnya tentang dimensi, lihat [Dimensi](#) di Amazon CloudWatch Panduan Pengguna.

Mencatat panggilan API Amazon Rekognition dengan AWS CloudTrail

Amazon Rekognition terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau layanan AWS di Amazon Rekognition. CloudTrail menangkap semua panggilan API untuk Amazon Rekognition sebagai peristiwa. Panggilan yang direkam mencakup panggilan dari konsol Amazon Rekognition dan panggilan kode ke operasi API Amazon Rekognition. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman berkelanjutan CloudTrail peristiwa ke bucket Amazon S3, termasuk acara untuk Amazon Rekognition. Jika Anda tidak membuat konfigurasi jejak, Anda masih dapat melihat kejadian terbaru dalam konsol CloudTrail di Riwayat peristiwa. Menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat untuk Amazon Rekognition, alamat IP dari mana permintaan dibuat, yang membuat permintaan, ketika dibuat, dan rincian tambahan.

Untuk mempelajari lebih lanjut tentang CloudTrail, lihat [AWS CloudTrail Panduan Pengguna](#).

Informasi Amazon Rekognition diCloudTrail

CloudTrail diaktifkan pada akun AWS Anda saat Anda membuat akun tersebut. Saat aktivitas terjadi di Amazon Rekognition, aktivitas tersebut direkam dalamCloudTrailacara bersama dengan lainnyaAWSacara layanan diRiwayat acara. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS Anda. Untuk informasi lain, lihat [Melihat Peristiwa dengan Riwayat Peristiwa CloudTrail](#).

Untuk catatan kejadian yang sedang berlangsung di akun AWS Anda, termasuk kejadian untuk Amazon Rekognition, buatlah jejak. Jejak memungkinkan CloudTrail untuk mengirim berkas log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol tersebut, jejak diterapkan ke semua Wilayah AWS. Jejak mencatat kejadian dari semua Wilayah di partisi AWS dan mengirimkan berkas log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat membuat konfigurasi layanan AWS lainnya untuk menganalisis lebih lanjut dan bertindak berdasarkan data peristiwa yang dikumpulkan di log CloudTrail. Untuk informasi selengkapnya, lihat yang berikut:

- [Ikhtisar untuk Membuat Jejak](#)
- [CloudTrailLayanan dan Integrasi yang Didukung](#)
- [Mengonfigurasi Pemberitahuan Amazon SNS untukCloudTrail](#)
- [Menerima File Log CloudTrail dari Beberapa Wilayah](#) dan [Menerima File Log CloudTrail dari Beberapa Akun](#)

Semua tindakan Amazon Rekognition dicatat olehCloudTraildan didokumentasikan dalam[Referensi API Rekognition Amazon](#). Misalnya, panggilan untuk tindakan `CreateCollection`, `CreateStreamProcessor`, dan `DetectCustomLabels` menghasilkan entri di berkas log CloudTrail.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut:

- Bahwa permintaan dibuat dengan kredensial pengguna root atau pengguna AWS Identity and Access Management (IAM).
- Bahwa permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan.
- Bahwa permintaan dibuat oleh layanan AWS lain.

Untuk informasi lain, lihat [Elemen userIdentity CloudTrail](#).

Memahami entri berkas log Amazon Rekognition

Jejak adalah konfigurasi yang mengaktifkan pengiriman kejadian sebagai berkas log ke bucket Amazon S3 yang telah Anda tentukan. Berkas log CloudTrail berisi satu atau beberapa entri log. Peristiwa mewakili satu permintaan dari sumber apa pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. Berkas log CloudTrail bukan jejak tumpukan terurut dari panggilan API publik, sehingga berkas tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan CloudTrail entri log dengan tindakan untuk API berikut: `StartLabelDetection` dan `DetectLabels`.

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDAJ45Q7YFFAREXAMPLE",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/JorgeSouza",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
          "sessionIssuer": {
            "type": "Role",
            "principalId": "AIDAJ45Q7YFFAREXAMPLE",
            "arn": "arn:aws:iam::111122223333:role/Admin",
            "accountId": "111122223333",
            "userName": "Admin"
          },
          "webIdFederationData": {},
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2020-06-30T20:10:09Z"
          }
        }
      },
      "eventTime": "2020-06-30T20:42:14Z",
      "eventSource": "rekognition.amazonaws.com",
      "eventName": "StartLabelDetection",
      "awsRegion": "us-east-1",
```



```
"sourceIPAddress": "192.0.2.0",
"userAgent": "aws-cli/3",
"requestParameters": {
  "video": {
    "s3Object": {
      "bucket": "my-bucket",
      "name": "my-video.mp4"
    }
  }
},
"responseElements": {
  "jobId":
"653de5a7ee03bd5083edde98ea8fce5794fcea66d077bdd4cfb39d71aff8fc25"
},
"requestID": "dfcef8fc-479c-4c25-bef0-d83a7f9a7240",
"eventID": "b602e460-c134-4ecb-ae78-6d383720f29d",
"readOnly": false,
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
},
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDAJ45Q7YFFAREXAMPLE",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/JorgeSouza",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDAJ45Q7YFFAREXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/Admin",
        "accountId": "111122223333",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-06-30T21:19:18Z"
      }
    }
  },
  "eventTime": "2020-06-30T21:21:47Z",
```

```
"eventSource": "rekognition.amazonaws.com",
"eventName": "DetectLabels",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.0.2.0",
"userAgent": "aws-cli/3",
"requestParameters": {
  "image": {
    "s3object": {
      "bucket": "my-bucket",
      "name": "my-image.jpg"
    }
  }
},
"responseElements": null,
"requestID": "5a683fb2-aec0-4af4-a7df-219018be2155",
"eventID": "b356b0fd-ea01-436f-a9df-e1186b275bfa",
"readOnly": true,
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
}
]
}
```

Pedoman dan kuota dalam Amazon Rekognition

Bagian berikut memberikan Pedoman dan kuota saat menggunakan Amazon Rekognition. Terdapat dua tipe kuota. Kuota Set seperti ukuran citra maksimum tidak dapat diubah. Kuota default tercantum pada halaman [AWS Service Quotas](#) dapat diubah dengan mengikuti prosedur yang dijelaskan di bagian [Kuota default](#).

Topik

- [Wilayah yang didukung](#)
- [Kuota Set](#)
- [Kuota default](#)

Wilayah yang didukung

Untuk daftar AWS Wilayah tempat Amazon Rekognition tersedia, lihat [Wilayah AWS dan Titik Akhir di Referensi Umum Amazon Web Services](#).

Kuota Set

Berikut ini adalah daftar batasan di Amazon Rekognition yang tidak dapat diubah. Untuk informasi tentang batasan yang dapat Anda ubah, seperti batas Transaksi Per Detik (TPS), lihat [Kuota default](#).

Untuk batas Custom Label Amazon Rekognition, lihat [Pedoman dan Kuota di Label Kustom Amazon Rekognition](#).

Citra Amazon Rekognition

- Ukuran citra maksimum yang disimpan sebagai objek Amazon S3 dibatasi hingga 15 MB.
- Dimensi gambar maksimum untuk DetectModerationLabels adalah 10K piksel untuk lebar dan tinggi.
- Dimensi gambar maksimum untuk DetectLabels adalah 10K piksel untuk lebar dan tinggi.
- Untuk dapat terdeteksi, wajah tidak boleh lebih kecil dari 40x40 piksel dalam citra dengan ukuran 1920X1080 piksel. Citra dengan dimensi yang lebih besar dari 1920X1080 piksel akan membutuhkan ukuran minimum wajah yang lebih besar secara proporsional.

- Dimensi citra minimum untuk tinggi dan lebar adalah 80 piksel. Dimensi citra minimum tinggi dan lebar untuk DetectProtectiveEquipment adalah 64 piksel.
- Dimensi citra maksimum tinggi dan lebar untuk DetectProtectiveEquipment adalah 4096 piksel.
- Pendeteksian oleh DetectProtectiveEquipment, seseorang tidak boleh lebih kecil dari 100x100 piksel dalam citra dengan ukuran 800x1300. Citra dengan dimensi yang lebih besar dari 800x1300 piksel akan membutuhkan ukuran minimum orang yang lebih besar secara proporsional.
- Ukuran gambar maksimum sebagai byte mentah yang diteruskan sebagai parameter ke API adalah 5 MB. Batasnya adalah 4 MB untuk API DetectProtectiveEquipment.
- Amazon Rekognition mendukung format citra PNG dan JPEG. Artinya, citra yang Anda berikan sebagai input untuk berbagai operasi API, seperti DetectLabels dan IndexFaces harus berada dalam salah satu format yang didukung.
- Jumlah maksimum vektor wajah yang dapat Anda simpan dalam satu koleksi wajah adalah 20 juta.
- Jumlah maksimum vektor pengguna default yang dapat Anda simpan dalam satu koleksi wajah adalah 10 juta.
- Vektor wajah pencocokan maksimum yang dikembalikan oleh API pencarian adalah 4096.
- Vektor pengguna pencocokan maksimum yang dikembalikan oleh API pencarian adalah 4096.
- DetectText dapat mendeteksi hingga 100 kata dalam sebuah gambar.
- DetectProtectiveEquipment dapat mendeteksi Alat Pelindung Diri hingga 15 orang.

Untuk informasi praktik terbaik mengenai citra dan perbandingan wajah, lihat [Praktik terbaik untuk sensor, citra input, dan video](#).

Analisis Massal Gambar Rekognition Amazon

- Amazon Rekognition Image Bulk Analysis dapat menganalisis kumpulan gambar hingga 10k gambar dalam ukuran.
- Amazon Rekognition Image Bulk Analysis mendukung manifes input hingga ukuran 50MB.

Video yang disimpan Amazon Rekognition Video

- Amazon Rekognition Video dapat menganalisis video yang disimpan dengan ukuran hingga 10GB.
- Amazon Rekognition Video dapat menganalisis video yang disimpan dengan durasi hingga 6 jam.

- Amazon Rekognition Video mendukung maksimal 20 tugas secara bersamaan per akun.
- Video yang disimpan harus dikodekan menggunakan codec H.264. Format file yang didukung adalah MPEG-4 dan MOV.
- API Video Rekognition Amazon apa pun yang menganalisis data audio hanya mendukung codec audio AAC.
- Periode Waktu Untuk Tayang (TTL) untuk token pemberian nomor halaman adalah 24 jam. Token pemberian nomor halaman berada di bidang `NextToken` yang dikembalikan oleh operasi `Get` seperti `GetLabelDetection`.

Video streaming Amazon Rekognition Video

- Pengaliran input Kinesis Video dapat dikaitkan dengan paling banyak 1 pemroses pengaliran Amazon Rekognition Video.
- Pengaliran output Kinesis Data dapat dikaitkan dengan paling banyak 1 pemroses pengaliran Amazon Rekognition Video.
- Pengaliran input Kinesis Video dan pengaliran output Kinesis Data yang terkait dengan pemroses pengaliran Amazon Rekognition Video tidak dapat dibagikan oleh beberapa pemroses.
- API Video Rekognition Amazon apa pun yang menganalisis data audio hanya mendukung codec audio ACC.

Kuota default

Daftar kuota default dapat ditemukan di [AWS Service Quotas](#). Batas ini adalah default dan dapat diubah. Untuk meminta kenaikan batas, Anda membuat kasus. Untuk melihat batas kuota saat ini (nilai kuota yang diterapkan), lihat [Service Quotas Amazon Rekognition](#). Untuk melihat riwayat pemanfaatan TPS Anda untuk [API Citra Amazon Rekognition](#), lihat [Halaman Service Quotas Amazon Rekognition](#) dan pilih operasi API tertentu untuk melihat riwayat operasi tersebut.

Topik

- [Hitung perubahan kuota TPS](#)
- [Praktik terbaik untuk kuota TPS](#)
- [Buat kasus untuk mengubah kuota TPS](#)

Hitung perubahan kuota TPS

Berapa batas baru yang Anda minta? Transaksi Per Detik (TPS) yang paling relevan berada pada puncak beban kerja yang diharapkan. Hal ini penting untuk memahami maksimal panggilan API yang bersamaan di puncak beban kerja dan waktu untuk merespons (5 - 15 detik). Harap dicatat, 5 detik merupakan nilai minimum. Berikut adalah dua contoh:

- Contoh 1: Pengguna Face Authentication (CompareFaces API) bersamaan maksimal yang saya harapkan di awal jam tersibuk saya adalah 1000. Tanggapan ini akan disebar dalam jangka waktu 10 detik. Oleh karena itu, TPS yang diperlukan adalah 100 (1000/10) untuk CompareFaces API di wilayah saya yang relevan.
- Contoh 2: Panggilan Deteksi Objek (DetectLabels API) bersamaan maksimum yang diharapkan pada awal jam tersibuk saya adalah 250. Tanggapan ini akan disebar dalam jangka waktu 5 detik. Oleh karena itu, TPS yang diperlukan adalah 50 (250/5) untuk DetectLabels API di wilayah saya yang relevan.

Praktik terbaik untuk kuota TPS

Praktik terbaik yang disarankan untuk Transaksi Per Detik (TPS) termasuk memperlancar lalu lintas yang padat, mengonfigurasi percobaan ulang, dan mengonfigurasi backoff eksponensial dan jitter.

1. Perlancar lalu lintas yang padat. Lalu lintas yang padat mempengaruhi throughput. Untuk mendapatkan throughput maksimum untuk transaksi per detik (TPS) yang dialokasikan, gunakan arsitektur antrean nirserver atau mekanisme lain untuk “memperlancar” lalu lintas sehingga lebih konsisten. Untuk contoh kode dan referensi pemrosesan citra dan video berskala besar nirserver dengan Rekognition, lihat [Pemrosesan citra dan video berskala besar dengan Amazon Rekognition](#).
2. Konfigurasi percobaan ulang. Ikuti pedoman di [the section called “Penanganan kesalahan”](#) untuk mengonfigurasi percobaan ulang untuk kesalahan yang mengaktifkan mereka.
3. Konfigurasi backoff eksponensial dan jitter. Mengonfigurasi backoff eksponensial dan jitter saat Anda mengonfigurasi percobaan ulang memungkinkan Anda untuk meningkatkan throughput yang dapat dicapai. Lihat [Percobaan ulang kesalahan dan backoff eksponensial](#) di AWS.

Buat kasus untuk mengubah kuota TPS

Untuk membuat kasus, buka [Buat Kasus](#) dan jawab pertanyaan berikut:

- Apakah Anda sudah menerapkan [the section called “Praktik terbaik untuk kuota TPS”](#) untuk memperlancar lonjakan lalu lintas dan mengonfigurasi percobaan ulang, backoff eksponensial, dan jitter?
- Sudahkah Anda menghitung perubahan kuota TPS yang Anda butuhkan? Jika belum, lihat [the section called “Hitung perubahan kuota TPS”](#).
- Sudahkah Anda memeriksa riwayat penggunaan TPS Anda untuk memprediksi kebutuhan masa mendatang secara lebih akurat? Untuk melihat riwayat penggunaan TPS, lihat [Halaman Service Quotas Amazon Rekognition](#).
- Apa kasus penggunaan Anda?
- Apa API yang Anda rencanakan untuk digunakan?
- Di Wilayah mana Anda berencana untuk menggunakan API ini?
- Apakah Anda dapat menyebarkan beban di beberapa wilayah?
- Berapa banyak citra yang Anda proses setiap hari?
- Berapa lama Anda berharap untuk mempertahankan volume ini (Apakah hanya sekali lonjakan atau berkelanjutan)?
- Bagaimana Anda diblokir oleh batas default? Tinjau tabel pengecualian berikut untuk mengonfirmasi skenario yang Anda hadapi.

Kode Kesalahan	Pengecualian	Pesan	Apa artinya?	Bisakah dicoba lagi?
Kode status HTTP 400	ProvisionedThroughputExceededException	Tarif yang Disediakan terlampaui.	Menunjukkan throttling. Anda dapat mencoba lagi atau mengevaluasi permintaan peningkatan batas.	Ya
Kode status HTTP 400	ThrottlingException	Pelan-pelan; peningkatan mendadak dalam tingkat permintaan.	Anda mungkin mengirim lalu lintas yang padat dan menghadap	Ya

Kode Kesalahan	Pengecualian	Pesan	Apa artinya?	Bisakah dicoba lagi?
			i throttling. Anda harus membentuk lalu lintas dan membuatnya lebih lancar serta konsisten. Kemudian konfigurasi tambahan percobaan ulang. Lihat praktik terbaik.	
Kode status HTTP 5xx	ThrottlingException (HTTP 500)	Layanan Tidak Tersedia	Menunjukkan bahwa backend diskalakan ke atas untuk mendukung tindakan. Anda harus mencoba kembali permintaan tersebut.	Ya

Untuk detail rinci tentang kode kesalahan, lihat [the section called “Penanganan kesalahan”](#).

Note

Batas ini tergantung pada wilayah tempat Anda berada. Membuat kasus untuk mengubah batas memengaruhi operasi API yang Anda minta, di wilayah yang Anda minta. Operasi API dan wilayah lainnya tidak terpengaruh.

Riwayat dokumen untuk Amazon Rekognition

Tabel berikut menjelaskan perubahan penting dalam setiap perilsan dari Panduan Developer Amazon Rekognition. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan ke umpan RSS.

- Pembaruan dokumentasi terbaru: 15 Juni 2023

Perubahan	Deskripsi	Tanggal
Amazon Rekognition sekarang mendukung label moderasi baru dan peningkatan akurasi untuk moderasi konten gambar	Fitur Moderasi Konten Amazon Rekognition telah ditingkatkan untuk meningkatkan akurasi, deteksi label baru, dan kemampuan untuk mengidentifikasi konten animasi dan/atau bergambar.	Februari 1, 2024
Amazon Rekognition sekarang mendukung analisis gambar massal	Amazon Rekognition sekarang mendukung pemrosesan koleksi besar gambar secara asinkron dengan menggunakan file manifes dengan operasi. StartMediaAnalysis Job	23 Oktober 2023
Amazon Rekognition sekarang mendukung moderasi konten khusus dengan adaptor	Amazon Rekognition sekarang mendukung peningkatan akurasi API dengan menggunakan adaptor DetectModerationLabels yang memperluas kemampuan model pembelajaran mendalam Rekognition yang ada.	12 Oktober 2023

[Rekognition sekarang mendukung vektor pengguna dengan koleksi](#)

Koleksi wajah Rekognition sekarang mendukung pembuatan vektor pengguna. Vektor pengguna menggabungkan beberapa vektor wajah dari pengguna yang sama, meningkatkan akurasi dengan penggambaran pengguna yang lebih kuat.

12 Juni 2023

[Tindakan untuk melibatkan pengelolaan pengguna telah ditambahkan ke kebijakan terkelola berikut: AmazonRekognitionReadOnlyAccess](#)

Amazon Rekognition menambahkan tindakan berikut ke kebijakan terkelola: AmazonRekognitionReadOnlyAccess, ListUsers, SearchUsers, SearchUsersByImage

12 Juni 2023

[Gambar Rekognition Amazon sekarang dapat menyimpulkan arah pandangan](#)

Perbaikan telah dilakukan pada operasi deteksi wajah Amazon Rekognition Image, yang sekarang dapat menyimpulkan arah pandangan pada wajah yang terdeteksi.

31 Mei 2023

[API moderasi konten Rekognition ditingkatkan](#)

Rekognition meningkatkan model moderasi konten untuk moderasi gambar dan video. Peningkatan ini secara signifikan memperluas deteksi konten eksplisit, kekerasan, dan sugestif. Pelanggan sekarang dapat mendeteksi konten eksplisit dan kekerasan dengan akurasi yang lebih tinggi untuk meningkatkan pengalaman pengguna akhir, melindungi identitas merek mereka, dan memastikan bahwa semua konten mematuhi peraturan dan kebijakan industri mereka.

9 Mei 2023

[Amazon Rekognition Image sekarang dapat mendeteksi wajah yang tersumbat](#)

Amazon Rekognition Image sekarang dapat mendeteksi oklusi wajah. FaceOccluded Atribut baru dikembalikan oleh Amazon Rekognition Image DetectFaces IndexFaces dan API, yang menunjukkan apakah wajah dalam gambar sebagian ditangkap atau tidak sepenuhnya terlihat karena objek, pakaian, dan bagian tubuh yang tumpang tindih.

5 Mei 2023

[Rekognition sekarang dapat mendeteksi keaktifan wajah](#)

Amazon Rekognition Video sekarang dapat digunakan untuk mendeteksi keaktifan dalam video, memverifikasi bahwa pengguna di depan kamera hadir secara fisik. Detektor Face Liveness juga mendeteksi serangan spoof yang disajikan ke kamera atau mencoba melewati kamera.

11 April 2023

[Perbarui ke Video Rekognition Amazon.](#)

Amazon Rekognition Video sekarang dapat mendeteksi lebih banyak label dan mengembalikan lebih banyak informasi tentang atribut gambar dan label. GetLabelDetection API sekarang mengembalikan informasi tentang alias dan kategori. Informasi label yang dikembalikan dapat disaring dengan opsi filter inklusif dan eksklusif. Hasil dapat dikumpulkan berdasarkan Stempel Waktu atau Segmen video.

7 Desember 2022

[Perbarui ke Gambar Rekogniti
on Amazon.](#)

Amazon Rekognition Image sekarang dapat mendeteksi lebih banyak label dan sekarang mengembalikan lebih banyak informasi tentang atribut gambar dan label. DetectLabels API sekarang mengembalikan informasi tentang alias, kategori, dan properti gambar seperti warna dominan. Informasi label yang dikembalikan dapat disaring dengan opsi filter inklusif dan eksklusif.

11 November 2022

[Tindakan untuk ProjectPo
licy dan Salinan Model Label
Kustom telah ditambahk
an ke kebijakan dikelola
berikut: AmazonRekognitionR
eadOnlyAccess](#)

Amazon Rekognition menambahkan tindakan berikut ke kebijakan dikelola : AmazonRekognitionReadOnlyAccess ListProjectPolicies

21 Juli 2022

[Tindakan untuk ProjectPo
licy dan Salinan Model Label
Kustom telah ditambahk
an ke kebijakan dikelola
berikut: AmazonRekognitionF
ullAccess, AmazonRek
ognitionCustomLabelsFullAcc
ess](#)

Rekognition menambahk an tindakan berikut ke AmazonRekognitionFullAccess dan kebijakan AmazonRekognitionCustomLabelsFullAccess CopyProjectVersion yang dikelola: ,, PutProjectPolicy ListProjectPolicies DeleteProjectPolicy

21 Juli 2022

[Amazon Rekognition Video sekarang dapat mendeteksi label dalam streaming video](#)

Amazon Rekognition Video dapat mendeteksi label seperti hewan peliharaan dan paket dalam streaming video. Ini dilakukan dengan menggunakan ConnectedHome pengaturan pada prosesor aliran yang dibuat dengan CreateStreamProcessor operasi.

28 April 2022

[Referensi API dipindahkan dari panduan pengembang Amazon Rekognition](#)

Referensi API Amazon Rekognition sekarang tersedia di Referensi API [Rekognition](#) Amazon.

Februari 24, 2022

[Pembaruan manajemen kumpulan data untuk kebijakan terkelola berikut:Kebijakan terkelola AWS: AmazonRekognitionReadOnlyAccess,, Kebijakan terkelola AWS: AmazonRekognitionFullAccessKebijakan terkelola AWS: AmazonRekognitionCustomLabelsFullAccess](#)

Amazon Rekognition menambahkan tindakan berikut ke,, AmazonRekognitionCustomLabelsFullAccess dan kebijakan AmazonRekognitionReadOnlyAccess terkelolaCreateDataset :AmazonRekognitionFullOnlyAccess ,,ListDatasetEntries ,,ListDatasetLabels , DescribeDataset UpdateDatasetEntries DistributeDatasetEntries DeleteDataset

November 1, 2021

[Node baru dalam daftar isi menunjukkan contoh Amazon Rekognition yang dihosting GitHub](#)

Contoh kode yang diperbarui dari Repositori Contoh AWS Kode sekarang muncul di node terpisah di panduan pengembang Amazon Rekognition untuk akses yang lebih mudah.

Oktober 22, 2021

[Amazon Rekognition dapat mendeteksi bingkai hitam dan konten program utama di segmen video](#)

Amazon Rekognition dapat mengidentifikasi bingkai hitam, bilah warna, kredit pembuka, kredit akhir, logo studio, dan konten program utama sebagai isyarat teknis dalam video menggunakan `StartSegmentDetection` dan operasi. `StartSegmentDetection` dan `GetSegmentDetection`

7 Juni 2021

[Pembaruan manajemen kumpulan data untuk kebijakan terkelola berikut:](#)

Anda dapat menggunakan operasi `DetectText` Amazon Rekognition untuk mendeteksi hingga 100 kata dalam sebuah citra.

21 Mei 2021

[Memberi tag pembaruan untuk `AmazonRekognitionReadOnlyAccess` dan `AmazonRekognitionFullAccess`](#)

Rekognition menambahkan tindakan penandaan baru ke dan kebijakan. `AmazonRekognitionFullAccess` dan `AmazonRekognitionReadOnlyAccess`

2 April 2021

Amazon Rekognition sekarang mendukung penandaan	Anda sekarang dapat menggunakan tanda untuk mengidentifikasi, mengatur, mencari, dan memfilter koleksi Amazon Rekognition, pemroses aliran, dan model Label Kustom.	25 Maret 2021
Amazon Rekognition sekarang dapat mendeteksi peralatan pelindung pribadi	Amazon Rekognition sekarang dapat mendeteksi sarung tangan, pelindung wajah, dan pelindung kepala pada orang di dalam citra.	15 Oktober 2020
Amazon Rekognition memiliki kategori moderasi konten baru	Kategori moderasi konten Amazon Rekognition sekarang mencakup 6 kategori baru: Narkoba, Tembakau, Alkohol, Perjudian, Gerakan Tidak Sopan, dan Simbol Kebencian.	12 Oktober 2020
Tutorial baru untuk menampilkan hasil Video Rekognition Amazon dari Kinesis Video Streams secara lokal	Anda dapat menampilkan output Amazon Rekognition Video dari video streaming di Kinesis Video Streams dalam umpan video lokal.	20 Juli 2020
Tutorial Amazon Rekognition baru untuk menggunakan Gstreamer	Dengan menggunakan Gstreamer, Anda dapat menyerap video langsung dari sumber kamera perangkat ke Amazon Rekognition Video melalui Kinesis Video Streams.	17 Juli 2020

Amazon Rekognition sekarang mendukung segmentasi video yang disimpan	Dengan API segmentasi Amazon Rekognition Video tidak sinkron, Anda dapat mendeteksi bingkai hitam, bilah warna, kredit akhir, dan citra dalam video yang tersimpan.	22 Juni 2020
Amazon Rekognition sekarang mendukung kebijakan titik akhir Amazon VPC	Dengan menentukan kebijakan Anda dapat membatasi akses ke Amazon VPC endpoint dari Amazon Rekognition.	3 Maret 2020
Amazon Rekognition sekarang mendukung deteksi teks dalam video yang disimpan	Anda dapat menggunakan API Amazon Rekognition Video untuk mendeteksi teks secara tidak sinkron dalam video yang tersimpan.	17 Februari 2020
Amazon Rekognition sekarang mendukung Augmented AI (Pratinjau) dan Label Kustom Rekognition Amazon	Dengan Label Kustom Amazon Rekognition Anda dapat mendeteksi objek khusus, adegan, dan konsep dalam citra dengan membuat model machine learning Anda sendiri. DetectModerationLabels sekarang mendukung Amazon Augmented AI (Pratinjau).	3 Desember 2019
Amazon Rekognition sekarang mendukung AWS PrivateLink	Dengan AWS, PrivateLink Anda dapat membuat koneksi pribadi antara VPC dan Amazon Rekognition.	12 September 2019

Pemfilteran wajah Amazon Rekognition	Amazon Rekognition menambahkan dukungan penyaringan wajah yang disempurnakan ke IndexFaces operasi API, dan memperkenalkan pemfilteran wajah untuk operasi dan API. CompareFaces SearchFacesByImage	12 September 2019
Contoh Video Rekognition Amazon diperbarui	Kode contoh Amazon Rekognition Video diperbarui untuk membuat dan mengonfigurasi topik Amazon SNS dan antrean Amazon SQS.	5 September 2019
Contoh Ruby dan Node.js ditambahkan	Contoh Ruby dan Node.js Amazon Rekognition Image ditambahkan untuk label sinkron dan deteksi wajah.	19 Agustus 2019
Deteksi konten tidak aman diperbarui	Deteksi konten tidak aman Amazon Rekognition sekarang dapat mendeteksi konten kekerasan.	9 Agustus 2019
GetContentModeration operasi diperbarui	GetContentModeration sekarang mengembalikan versi model deteksi moderasi yang digunakan untuk mendeteksi konten yang tidak aman.	13 Februari 2019

[GetLabelDetection dan DetectModerationLabels operasi diperbarui](#)

GetLabelDetection sekarang mengembalikan informasi kotak pembatas untuk objek umum dan taksonomi hierarkis label yang terdeteksi. Versi model yang digunakan untuk deteksi label sekarang dikembalikan. DetectModerationLabels sekarang mengembalikan versi model yang digunakan untuk mendeteksi konten yang tidak aman.

17 Januari 2019

[DetectFaces dan IndexFaces operasi diperbarui](#)

Rilis ini memperbarui DetectFaces dan IndexFaces operasi. Ketika parameter input Atribut diatur ke SEMUA, landmark lokasi wajah mencakup 5 landmark baru: upperJawlineLeft,, midJawlineLeft ChinBottom,,. midJawlineRight upperJawlineRight

19 November 2018

[DetectLabels operasi diperbarui](#)

Kotak batas sekarang dikembalikan untuk objek tertentu. Taksonomi hierarkis sekarang tersedia untuk label. Anda sekarang bisa mendapatkan versi model deteksi yang digunakan untuk deteksi.

1 November 2018

IndexFaces operasi diperbarui	Dengan IndexFaces Anda sekarang dapat menggunakan parameter QualityFilter input untuk menyaring wajah yang terdeteksi dengan kualitas rendah. Anda juga dapat menggunakan parameter MaxFaces input untuk mengurangi jumlah wajah yang dikembalikan berdasarkan kualitas deteksi wajah, dan ukuran wajah yang terdeteksi.	18 September 2018
DescribeCollection operasi ditambahkan	Anda sekarang dapat memperoleh informasi tentang koleksi yang ada dengan memanggil DescribeCollection operasi.	22 Agustus 2018
Contoh Python baru	Contoh Python telah ditambahkan ke konten Amazon Rekognition Video bersama dengan beberapa reorganisasi konten.	26 Juni 2018
Tata letak konten diperbarui	Konten Amazon Rekognition Image telah direorganisasi bersama dengan contoh Python dan C# baru.	29 Mei 2018

[Amazon Rekognition mendukung AWS CloudTrail](#)

Amazon Rekognition terintegrasi dengan AWS CloudTrail, sebuah layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau layanan AWS di Amazon Rekognition. Untuk informasi selengkapnya, lihat [Mencatat Panggilan API Rekognition Amazon](#) dengan AWS. CloudTrail

6 April 2018

[Analisis video streaming dan yang disimpan. Daftar isi baru](#)

Untuk informasi tentang cara menganalisis video yang tersimpan, lihat [Bekerja dengan Video yang Tersimpan](#). Untuk informasi tentang cara menganalisis video streaming, lihat [Bekerja dengan Video Streaming](#). Daftar isi untuk dokumentasi Amazon Rekognition telah disusun ulang untuk mengakomodasi operasi citra dan video.

29 November 2017

[Teks dalam model deteksi gambar dan wajah](#)

Sekarang Amazon Rekognition dapat mendeteksi teks dalam citra. Untuk informasi selengkapnya, lihat [Mendeteksi Teks](#). Amazon Rekognition memperkenalkan versioning untuk model deep learning pendeteksi wajah. Untuk informasi selengkapnya, lihat [Versioning Model](#).

21 November 2017

Pengakuan selebriti	Amazon Rekognition sekarang dapat menganalisis citra untuk selebriti. Untuk informasi selengkapnya, lihat Mengenali Selebriti .	8 Juni 2017
Moderasi gambar	Sekarang Amazon Rekognition dapat menentukan apakah citra berisi konten dewasa yang eksplisit atau sugestif. Untuk informasi selengkapnya, lihat Mendeteksi Konten yang Tidak Aman .	19 April 2017
Rentang usia untuk wajah yang terdeteksi. Panel metrik Aggregated Rekognition	Amazon Rekognition sekarang mengirimkan rentang usia taksiran, dalam hitungan tahun, untuk wajah yang terdeteksi oleh API Rekognition. Untuk informasi lebih lanjut, lihat AgeRange . Konsol Rekognition sekarang memiliki panel metrik yang menampilkan grafik aktivitas untuk agregat metrik CloudWatch Amazon untuk Rekognition selama periode waktu tertentu. Untuk informasi selengkapnya, lihat Latihan 4: Lihat Metrik Agregat (Konsol) .	9 Februari 2017
Layanan dan panduan baru	Ini adalah perilisan awal dari layanan analisis citra, Amazon Rekognition, dan Panduan developer Amazon Rekognition.	30 November 2016

AWSGlosarium

Untuk AWS terminologi terbaru, lihat [AWSglosarium di Referensi](#). Glosarium AWS

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.