aws

Entwicklerhandbuch

Amazon Rekognition

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Rekognition: Entwicklerhandbuch

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Marken und Handelsmarken von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, die geeignet ist, Kunden irrezuführen oder Amazon in irgendeiner Weise herabzusetzen oder zu diskreditieren. Alle anderen Marken, die nicht im Besitz von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist Amazon Rekognition? 1
Die wichtigsten Funktionen 1
Anwendungsfälle 2
Vorteile 4
Amazon Rekognition und HIPAA-Eignung 5
Verwenden Sie Amazon Rekognition zum ersten Mal? 5
So funktioniert Rekognition 6
Grundlegendes zu den Analysetypen von Rekognition 7
Labels 9
Benutzerdefinierte Labels 10
Erkennung der Echtheit von Gesichtern 11
Gesichtserkennung und -analyse 11
Gesichtssuche 12
Pfade von Personen 12
Persönliche Schutzausrüstung 12
Prominente 12
Texterkennung 13
Unangemessene oder anstößige Inhalte 13
Anpassung 13
Massenanalyse 14
Die Bild- und Videooperationen von Rekognition verstehen 14
Funktionsweise von Amazon Rekognition Image 14
Amazon-Rekognition-Video-Operationen 15
Nicht-speicherbasierte und speicherbasierte Operationen 15
Verwendung des AWS-SDK oder HTTP zum Aufrufen von Amazon-Rekognition-API-Vorgängen 16
Nicht-speicherbasierte und speicherbasierte API-Operationen 16
Nicht speicherbasierte Operationen 17
Speicherbasierte API-Operationen 19
Grundlegendes zur Modellversionierung 20
Erste Schritte 22
Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers 23
Erstellen Sie ein AWS Konto und einen Benutzer 23
Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs 26

Erteilen programmgesteuerten Zugriffs 28
Arbeitet mit AWS SDKs 32
Schritt 3: Erste Schritte mit der AWS CLIAWS SDK-API 33
Formatieren der AWS CLI Beispiele 34
Nächster Schritt 34
Schritt 4: Erste Schritte mit der Konsole 34
Konsolenberechtigungen einrichten 35
Übung 1: Erkennen von Objekten und Szenen (Konsole) 39
Übung 2: Gesichtsanalyse (Konsole) 47
Übung 3: Gesichtervergleich (Konsole) 49
Übung 4: Anzeigen von Gesamtmetriken (Konsole) 52
Mit Bildern und Videos arbeiten 55
Arbeiten mit Bildern 55
Bildspezifikationen 56
Analysieren von Bildern in einem Amazon-S3-Bucket 58
Verwenden eines lokalen Dateisystems 75
Anzeigen von Begrenzungsrahmen 90
Erhalten der Bildausrichtung und der Koordinaten von Begrenzungsrahmen 103
Arbeiten mit gespeicherten Videoanalysevorgängen 114
Analysearten 115
Überblick für die Amazon-Rekognition-Video-API 115
Amazon-Rekognition-Video-Operationen aufrufen 118
Amazon Rekognition Video konfigurieren 125
Analysieren eines gespeicherten Videos (SDK) 129
Analysieren eines Videos (AWS CLI) 159
Referenz: Videoanalyse-Ergebnisbenachrichtigung 163
Fehlerbehebung für Amazon Rekognition Video 164
Arbeiten mit Streaming-Videoereignissen 167
Überblick über den Betrieb des Amazon-Rekognition-Video-Stromprozessors 167
Hinzufügen von Tags zu einem neuen Stromprozessor 168
Hinzufügen von Tags zu einem bestehenden Stromprozessor 169
Hinzufügen von Tags in einem Stromprozessor 170
Löschen von Tags von einem Stromprozessor 170
Fehlerbehandlung 171
Fehlerkomponenten 171
Fehlermeldungen und Codes 172

Fehlerbehandlung in Ihrer Anwendung ... 177
Amazon Rekognition mit FedRAMP verwenden ... 179
Bewährte Methoden für Sensoren, Eingabebilder und Videos ... 183
Latenz der Amazon-Rekognition-Image-Operation ... 183
Empfehlungen zu Eingabebildern für den Gesichtsvergleich ... 184
Allgemeine Empfehlungen für Eingabebilder für Gesichtsoperationen ... 184
Empfehlungen für die Suche nach Gesichtern in einer Sammlung ... 185
Empfehlungen für die Kameraeinrichtung (Bild und Video) ... 186
Empfehlungen für die Kameraeinrichtung (Gespeicherte Bilder und Videostreaming) ... 188
Empfehlungen für die Kameraeinrichtung (Videostreaming) ... 188
Empfehlungen für die Verwendung von Face Liveness ... 189
Erkennung von Objekten und Konzepten ... 191
Label-Antwortobjekte ... 193
Begrenzungsrahmen ... 193
Zuverlässigkeitswert ... 194
Übergeordnete Kategorien ... 194
Kategorien ... 194
Aliasnamen ... 194
Bildeigenschaften ... 195
Modellversion ... 196
Einschluss- und Ausschlussfilter ... 197
Ergebnisse sortieren und aggregieren ... 197
Erkennen von Labels in einem Bild ... 197
DetectLabels Operationsanfrage ... 209
DetectLabels Antwort ... 210
Transformation der DetectLabels Antwort ... 214
Erkennen von Labels in einem Video ... 218
StartLabelDetectionAnfrage ... 218
GetLabelDetection Antwort auf die Operation ... 220
Transformation der Antwort GetLabelDetection ... 226
Erkennung von Labels in Streaming-Videoereignissen ... 235
Einrichtung Ihrer Amazon-Rekognition-Video- und Amazon-Kinesis-Ressourcen ... 236
Rufen Sie Operationen zur Erkennung von Bezeichnungen für Streaming-Videoereignisse auf ... 241
Erkennen benutzerdefinierter Label ... 248
Erkennung und Analyse von Gesichtern ... 249

Übersicht über Gesichtserkennung und Gesichtsvergleich 250
Richtlinien zu Gesichtsattributen 252
Erkennen von Gesichtern in einem Bild 253
DetectFaces Operationsanforderung 265
DetectFaces Antwort auf die Operation 265
Vergleichen von Gesichtern in Bildern 273
CompareFaces Operationsanforderung 285
CompareFaces Antwort auf die Operation 286
Erkennen von Gesichtern in einem gespeicherten Video 289
GetFaceDetection Antwort auf den Vorgang 298
Gesichtssuche in einer Sammlung 304
Verwaltung von Gesichtssammlungen, Gesichtern und Benutzern 307
Eine Sammlung verwalten 308
Verwalten von Gesichtern in einer Sammlung 308
Benutzer in einer Sammlung verwalten 309
Verwendung von Ähnlichkeitsschwellenwerten für die Zuordnung und Zuordnung von Gesichtern 309
Verwendung von Ähnlichkeitsschwellenwerten für die Zuordnung von Gesichtern 309
Verwenden von Übereinstimmungsschwellenwerten für den Gesichterabgleich 310
Anleitung zur Indexierung von Gesichtern in gängigen Szenarien 311
Kritische oder die öffentliche Sicherheit betreffende Anwendungen 311
Anwendungen für Bildaustausch und soziale Medien 311
Allgemeine Nutzung 311
Suche nach Gesichtern und Benutzern in einer Sammlung 312
Anwendungsfälle, die die öffentliche Sicherheit betreffen 312
Verwendung von Amazon Rekognition zur Unterstützung der öffentlichen Sicherheit 314
Erstellen einer Sammlung 315
CreateCollection Operationsanforderung 322
CreateCollection Antwort auf die Operation 322
Markieren von Sammlungen 322
Hinzufügen von Tags zu einer neuen Sammlung 323
Hinzufügen von Tags zu einer vorhandenen Sammlung 324
Stichwörter in einer Sammlung auflisten 325
Löschen von Tags aus einer Sammlung 326
Auflisten von Sammlungen 327
ListCollections Operationsanforderung 334

ListCollections Antwort auf die Operation ... 334
Beschreiben einer Sammlung ... 335
DescribeCollection Operationsanforderung ... 342
DescribeCollectionAntwort auf die Operation ... 342
Löschen einer Sammlung ... 343
DeleteCollection Operationsanforderung ... 349
DeleteCollection Antwort auf die Operation ... 349
Hinzufügen von Gesichtern zu einer Sammlung ... 349
Filtern von Gesichtern ... 351
IndexFaces Operationsanforderung ... 360
IndexFaces Antwort auf die Operation ... 361
Gesichter und zugehörige Benutzer in einer Sammlung auflisten ... 370
ListFaces Operationsanforderung ... 376
ListFaces Reaktion auf den Vorgang ... 376
Löschen von Gesichtern aus einer Sammlung ... 378
DeleteFaces Betriebsanfrage ... 383
DeleteFaces Reaktion auf den Vorgang ... 384
Erstellen eines Benutzers ... 384
Löschen eines Benutzers ... 388
Gesichter einem Benutzer zuordnen ... 391
AssociateFaces Antwort auf den Vorgang ... 394
Trennen der Zuordnung von Gesichtern zu einem Benutzer ... 395
DisassociateFaces Antwort auf den Vorgang ... 398
Auflisten von Benutzern in einer Sammlung ... 399
ListUsers Antwort auf den Vorgang ... 402
Suchen nach einem Gesicht (Gesichts-ID) ... 402
SearchFaces Operationsanforderung ... 409
SearchFaces Antwort auf die Operation ... 410
Suchen nach einem Gesicht (Bild) ... 411
SearchFacesByImage Operationsanforderung ... 418
SearchFacesByImage Antwort auf die Operation ... 419
Suche nach Benutzern (Gesichts-ID/Benutzer-ID) ... 420
SearchUsers Operationsanforderung ... 424
SearchUsers Antwort auf den Vorgang ... 425
Suchen nach Benutzern (Bild) ... 426
SearchUsersByImage Operationsanforderung ... 430

SearchUsersByImage Antwort auf die Operation 430
Suche nach Gesichtern in gespeicherten Videos 432
GetFaceSearch Antwort auf den Vorgang 441
Suchen nach Gesichtern in einer Sammlung im Streaming-Video 446
Einrichtung Ihrer Amazon-Rekognition-Video- und Amazon-Kinesis-Ressourcen 447
Suche nach Gesichtern in einem Streaming-Video 450
Streaming mit einem GStreamer Plugin 475
Fehlerbehebung beim Streamen von Videos 478
Pfade von Personen 486
GetPersonTracking Reaktion auf den Vorgang 496
Erkennung von persönlicher Schutzausrüstung 500
Arten von PSA 501
Gesichtsbedeckung 501
Handbedeckung 501
Kopfbedeckung 501
Zuverlässigkeit der PSA-Erkennung 502
Zusammenfassung der PSA, die in einem Bild erkannt wurde 502
Tutorial: Eine AWS Lambda Funktion erstellen, die Bilder mit PPE erkennt 503
Die API zur Erkennung persönlicher Schutzausrüstung verstehen 503
Bereitstellung eines Bilds 503
Die DetectProtectiveEquipment Antwort verstehen 505
Erkennen von PSA in einem Bild 511
Beispiel: Begrenzungsrahmen und Gesichtsbedeckungen 523
Erkennen von Prominenten 539
Prominentenerkennung im Vergleich zur Gesichtssuche 540
Erkennen von Prominenten in einem Bild 540
Anrufen RecognizeCelebrities 541
RecognizeCelebrities Operationsanfrage 551
RecognizeCelebrities Antwort auf die Operation 551
Erkennen von Prominenten in einem gespeicherten Video 554
GetCelebrityRecognition Reaktion auf den Vorgang 570
Abrufen von Informationen eines Prominenten 572
Anrufen GetCelebrityInfo 572
GetCelebrityInfo Betriebsanfrage 578
GetCelebrityInfo Antwort auf die Operation 578
Inhalte moderieren 579

Verwendung der Bild- und Videomoderation APIs 581
Labelkategorien 582
Inhaltstyp 595
Wahrscheinlichkeit 595
Versionsverwaltung 596
Sortieren und Aggregieren 596
Status des benutzerdefinierten Moderationsadapters 596
Content Moderation Version 7 testen und die API-Antwort transformieren 597
AWS SDK und Nutzerhandbuch für Content Moderation, Version 7 598
Labelzuordnungen für die Versionen 6.1 bis 7 598
Erkennen unangemessener Bilder 603
Erkennen von unangemessenen Inhalten in einem Bild 604
..... 604
DetectModerationLabels Operationsanforderung 611
DetectModerationLabels Antwort auf die Operation 611
Erkennung unangemessener gespeicherter Videos 613
GetContentModeration Antwort auf den Vorgang 621
Verbesserung der Genauigkeit mit benutzerdefinierter Moderation 624
Adapter erstellen und verwenden 625
Vorbereiten Ihrer Datensätze 628
Adapter mit der AWS CLI verwalten und SDKs 630
Tutorial zum benutzerdefinierten Moderationsadapter 637
Evaluierung und Verbesserung Ihres Adapters 656
Manifest-Dateiformate 658
Bewährte Methoden für Trainingsadapter 664
AutoUpdateBerechtigungen einrichten 664
AWS Health Dashboard-Benachrichtigung für Rekognition 667
Überprüfen unangemessener Inhalte mit Amazon A2I 669
Erkennen von Text 675
Erkennen von Text in einem Bild 677
DetectText Operationsanforderung 686
DetectText Reaktion auf den Betrieb 687
Erkennen von Text in einem gespeicherten Video 692
Filter 702
GetTextDetection — Antwort 703
Erkennen von Videosegmenten 709

Technische Signale ... 710
Schwarze Frames ... 710
Guthaben ... 710
Farbbalken ... 711
Slates ... 711
Studiologos ... 711
Inhalt ... 711
Einstellungserkennung ... 712
Über die Amazon-Rekognition-Video-Segmenterkennungs-API ... 713
Verwenden der Amazon-Rekognition-Segment-API ... 713
Starten der Segmentanalyse ... 714
Abrufen der Ergebnisse der Segmentanalyse ... 715
Beispiel: Erkennen von Segmenten in einem gespeicherten Video ... 720
Echtheit von Gesichtern erkennen ... 733
Benutzerseitige Anforderungen an Face Liveness ... 735
Architektur- und Reihenfolgediagramme ... 736
Voraussetzungen ... 738
Schritt 1: Einrichten eines AWS -Kontos ... 738
Schritt 2: Face Liveness einrichten AWS SDKs ... 738
Schritt 3: AWS Amplify Resources einrichten ... 739
Bewährte Methoden zur Erkennung von Face Liveness ... 739
Programmierung von Amazon Rekognition Face Liveness APIs ... 739
Schritt 1: CreateFaceLivenessSession ... 740
Schritt 2: StartFaceLivenessSession ... 741
Schritt 3: GetFaceLivenessSessionResults ... 742
Schritt 4: Reagieren Sie auf Ergebnisse ... 743
Ich nenne Face Liveness APIs ... 743
Konfiguration und Anpassung Ihrer Anwendung ... 749
Konfigurieren Ihrer -Anwendung ... 749
Anpassen Ihrer Anwendung ... 750
Modell der gemeinsamen Verantwortung für Face Liveness ... 750
Richtlinien für das Update von Face Liveness ... 755
Versionsverwaltung und Zeitrahmen ... 755
Versionsveröffentlichung und Kompatibilitätsmatrix ... 756
Kommunikation von Neuversionen ... 757
Häufig gestellte Fragen zu Face Liveness ... 757

Massenanalyse 761
Bilder werden in großen Mengen verarbeitet 761
So erstellen Sie einen Massenanalyseauftrag (CLI) 761
Die Ergebnisse der Massenanalyse werden angezeigt 763
Inhaltstyp 764
Überprüfung von Prognosen und Trainingsadaptern 764
Tutorials 765
Speichern von Amazon-Rekognition-Daten mit Amazon RDS und DynamoDB 765
Voraussetzungen 766
Labels für Bilder in einem Amazon-S3-Bucket abrufen 766
Erstellen einer Amazon DynamoDB-Tabelle 768
Hochladen von Daten in DynamoDB 769
Eine MySQL-Datenbank in Amazon RDS erstellen 771
Daten in eine Amazon-RDS-MySQL-Tabelle hochladen 772
Verwenden von Amazon Rekognition und Lambda zum Taggen von Assets in einem Amazon-S3-Bucket 775
Voraussetzungen 776
Konfigurieren der IAM-Lambda-Rolle 777
Erstellen des Projekts 777
Schreiben des Codes 780
Projekt verpacken 791
Stellen Sie die Lambda-Funktion bereit 792
Lambda-Methode testen 792
AWS Videoanalysator-Anwendungen erstellen 794
Voraussetzungen 795
Verfahren 795
Erstellen einer Lambda-Funktion für Amazon Rekognition 795
Voraussetzungen 798
Erstellen eines SNS-Themas 798
So erstellen Sie die Lambda-Funktion: 798
Konfigurieren der Lambda-Funktion 799
Konfigurieren der IAM-Lambda-Rolle 800
Erstellen Sie das AWS Toolkit for Eclipse -Lambda-Projekt 801
Lambda-Funktion testen 805
Amazon Rekognition zur Identitätsprüfung verwenden 806
Voraussetzungen 807

Erstellen einer Sammlung ... 808
Registrierung neuer Benutzer ... 809
Anmeldung bestehender Benutzer ... 818
Verwendung von Amazon Rekognition (REK) zur Erkennung von Etiketten für Marketinganwendungen ... 821
Voraussetzungen ... 822
DynamoDB-Tabelle erstellen und auffüllen ... 822
Eine Lambda-Funktion mit Protokollierung und Benachrichtigung erstellen ... 825
Berechtigungen und die Lambda-Funktion einrichten ... 827
Bilder mit Metadaten auf Amazon S3 hochladen ... 829
CloudWatch Abfrageprotokolle vom Client ... 830
Label in einem Bild mit Lambda und Python erkennen ... 832
Erstellen einer Lambda-Funktion (Konsole) ... 832
(Optional) Erstellen einer Ebene (Konsole) ... 834
Python-Code hinzufügen (Konsole) ... 835
So fügen Sie Python-Code hinzu (Konsole) ... 837
Codebeispiele ... 842
Grundlagen ... 846
Hallo Amazon Rekognition ... 847
Aktionen ... 850
Szenarien ... 1009
Erstellen Sie eine Sammlung und finden Sie Gesichter darin ... 1010
Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos ... 1022
Erkennen von PSA in Bildern ... 1027
Elemente in Bildern erkennen und anzeigen ... 1027
Gesichter in einem Bild erkennen ... 1043
Informationen in Videos erkennen ... 1044
Erkennen von Objekten in Bildern ... 1084
Erkennen Sie Personen und Objekte in einem Video ... 1087
EXIF- und andere Bildinformationen speichern ... 1089
API-Referenz ... 1091
Sicherheit ... 1092
Identity and Access Management ... 1092
Zielgruppe ... 1093
Authentifizierung mit Identitäten ... 1093
Verwalten des Zugriffs mit Richtlinien ... 1097

So funktioniert Amazon Rekognition mit IAM 1100
Von AWS verwaltete Richtlinien 1106
Bewährte Methoden für Richtlinien 1115
Verwenden der Konsole 1116
Beispiele für die Verwendung identitätsbasierter Richtlinienbeispiele 1116
Beispiele für ressourcenbasierte Richtlinien 1119
Fehlerbehebung 1120
Datenschutz 1123
Datenverschlüsselung 1124
Richtlinie für den Datenverkehr zwischen Netzwerken 1126
Verwenden von Amazon Rekognition mit Amazon-VPC-Endpunkten 1127
Erstellen von Amazon VPC-Endpunkten für Amazon Rekognition 1128
Eine VPC-Endpunktrichtlinie für Amazon Rekognition erstellen 1128
Compliance-Validierung 1130
Ausfallsicherheit 1130
Konfigurations- und Schwachstellenanalyse 1131
Serviceübergreifende Confused-Deputy-Prävention 1131
Sicherheit der Infrastruktur 1134
Überwachen 1135
Überwachung von Rekognition mit Amazon CloudWatch 1136
CloudWatch Metriken für Rekognition verwenden 1136
Zugreifen auf Rekognition-Metriken 1137
Alarm erstellen 1138
CloudWatchMetriken für Rekognition 1140
Protokollieren Amazon Rekognition Rekognition-API-Aufrufen mit AWS CloudTrail 1145
Informationen zu Amazon Rekognition in CloudTrail 1146
Erläuterungen zu Amazon-Rekognition-Protokolldateieinträgen 1147
Richtlinien und Kontingente 1150
Unterstützte -Regionen 1150
Festgelegte Kontingente 1150
Amazon Rekognition Image 1150
Amazon Rekognition Bild-Massenanalyse 1150
Gespeichertes Video von Amazon Rekognition Video 1152
Streaming-Video von Amazon Rekognition Video 1152
Standardkontingente 1152
Berechnen der TPS-Kontingentänderung 1153

Bewährte Methoden für TPS-Kontingente 1153
Erstellen Sie einen Fall, um TPS-Kontingente zu ändern 1154
Dokumentverlauf 1157
AWS Glossar 1172
..... mclxxiii

Was ist Amazon Rekognition?

Amazon Rekognition ist ein cloudbasierter Bild- und Videoanalysedienst, mit dem Sie Ihre Anwendungen auf einfache Weise um erweiterte Computer-Vision-Funktionen erweitern können. Der Service basiert auf bewährter Deep-Learning-Technologie und erfordert für die Nutzung keine Fachkenntnisse im Bereich maschinelles Lernen. Amazon Rekognition umfasst eine einfache easy-to-use API, mit der alle in Amazon S3 gespeicherten Bild- oder Videodateien schnell analysiert werden können.

Sie können Ihrer Anwendung mithilfe von Rekognition Funktionen hinzufügen, die Objekte, Text und unsichere Inhalte erkennen, Bilder/Videos analysieren und Gesichter vergleichen. APIs Mit der Gesichtserkennung APIs von Amazon Rekognition können Sie Gesichter für eine Vielzahl von Anwendungsfällen erkennen, analysieren und vergleichen, darunter Benutzerverifizierung, Katalogisierung, Personenzählung und öffentliche Sicherheit.

Der Service basiert auf derselben bewährten, hoch skalierbaren Deep-Learning-Technologie, die von den Computer-Vision-Wissenschaftlern von Amazon entwickelt wurde. Diese Technologie kann täglich Milliarden von Bildern und Videos analysieren. Rekognition lernt routinemäßig aus neuen Daten, und wir fügen dem Service häufig neue Labels und Funktionen hinzu.

Weitere Informationen finden Sie unter Amazon Rekognition FAQs.

Die wichtigsten Funktionen

Bildanalyse:

- Objekt-, Szenen- und Konzepterkennung — Erkennt und klassifiziert Objekte, Szenen, Konzepte und Prominente in Bildern.
- Texterkennung — Erkennt und erkennt gedruckten und handgeschriebenen Text in Bildern in einer Vielzahl von Sprachen.
- Unsichere Inhalte — Erkennt und filtert explizite, unangemessene und gewalttätige Inhalte und Bilder. Erkennt detaillierte Labels für unsichere Inhalte.
- Anerkennung von Prominenten — Erkennen Sie Zehntausende von Prominenten in Ihren Bildern aus verschiedenen Kategorien, wie Politiker, Sportler, Schauspieler und Musiker.

- Gesichtsanalyse — Erkennen, analysieren und vergleichen Sie Gesichter zusammen mit Gesichtsmerkmalen wie Geschlecht, Alter und Emotionen. Zu den Anwendungsfällen können Benutzerverifizierung, Katalogisierung, Personenzählung und öffentliche Sicherheit gehören.
- Benutzerdefinierte Labels — Erstellen Sie benutzerdefinierte Klassifikatoren, um Objekte zu erkennen, die für Ihren Anwendungsfall spezifisch sind, wie Logos, Produkte und Charaktere.
- Bildeigenschaften — Analysieren Sie Bildeigenschaften wie Qualität, Farbe, Schärfe und Kontrast.

Videoanalyse:

- Objekt-, Szenen- und Konzepterkennung — Erkennt und klassifiziert Objekte, Szenen, Konzepte und Prominente in Videos.
- Texterkennung — Erkennt und erkennt gedruckten und handgeschriebenen Text in Videos in einer Vielzahl von Sprachen.
- Personenverfolgung — Verfolge identifizierte Personen, während sie sich durch Videobilder bewegen.
- Gesichtsanalyse — Erkennen, analysieren und vergleichen Sie Gesichter in Streaming-Videos oder gespeicherten Videos.
- Anerkennung von Prominenten — Erkenne Zehntausende von Prominenten in deinen gespeicherten Videos aus verschiedenen Kategorien, wie Politiker, Sportler, Schauspieler und Musiker.
- Erkennung unsicherer Inhalte — Erkennt explizite, unangemessene und gewalttätige Inhalte in Videos.
- Videosegmentierung — Identifizieren Sie automatisch nützliche Videosegmente wie schwarze Frames und Abspann.
- Lebendigkeit von Gesichtern — Erkennt, ob bei der Gesichtsüberprüfung ein Live-Nutzer anwesend ist.

Anwendungsfälle

Durchsuchbare Medienbibliotheken — Rekognition erkennt Labels, Objekte, Konzepte und Szenen in Bildern und Videos. Auf der Grundlage dieser visuellen Inhaltsanalyse können Sie diese Labels durchsuchbar machen. Nützlich für den Aufbau durchsuchbarer Bild- und Videobibliotheken.

Gesichtsbasierte Überprüfung der Benutzeridentität — Bestätigen Sie Benutzeridentitäten, indem Sie Gesichter in Bildern mit Referenzgesichtsbildern vergleichen. Nützlich für die Identitätsprüfung in Anwendungen.

Gesichtserkennung — Rekognition Face Liveness ist eine vollständig verwaltete Funktion für maschinelles Lernen (ML), die Entwicklern dabei helfen soll, Betrug bei der Identitätsüberprüfung mit Gesicht zu verhindern. Mit diesem Feature können Sie überprüfen, ob ein Benutzer physisch vor der Kamera anwesend ist und es sich nicht um einen Kriminellen handelt, der das Gesicht des Benutzers vortäuscht. Mithilfe von Rekognition Face Liveness können Sie Spoof-Attacken erkennen, die auf eine Kamera gerichtet sind, z. B. gedruckte Fotos, digitale Fotos/Videos oder 3D-Masken. Es hilft auch dabei, Spoof-Angriffe zu erkennen, die eine Kamera umgehen, wie z. B. vorab aufgezeichnete oder Deepfake-Videos, die direkt in das Videoaufzeichnungs-Subsystem eingespeist werden.

Gesichtssuche — Mit Rekognition können Sie Bilder, gespeicherte Videos und Streaming-Videos nach Gesichtern durchsuchen, die denen entsprechen, die in einem Container gespeichert sind, der als Gesichtssammlung bezeichnet wird. Eine Gesichtersammlung ist ein Index mit Gesichtern, den Sie besitzen und verwalten. Um anhand ihrer Gesichter nach Personen zu suchen, müssen Sie die Gesichter indexieren und dann nach den Gesichtern suchen.

Erkennung unsicherer Inhalte — Erkennt und filtert explizite, unangemessene und gewalttätige Inhalte in Bildern und Videos. Verwendet Labels für eine detaillierte Filterung auf der Grundlage von Geschäftsanforderungen. Die Content Moderation API gibt außerdem eine hierarchische Liste aller erkannten Labels (Objekte und Konzepte) zusammen mit Konfidenzwerten zurück. Diese Objekte/Label verweisen auf spezifische Kategorien unsicherer Inhalte. Auf diese Weise können große Mengen an nutzergenerierten Inhalten fein gefiltert und verwaltet werden. Sie können die Ausgabe der Content Moderation API mit Adaptern anpassen, wodurch die Leistung von Bildern, wie denen, die Sie als Trainingsdaten bereitstellen, verbessert wird.

Erkennung persönlicher Schutzausrüstung — Erkennen Sie persönliche Schutzausrüstung in Bildern, um die Einhaltung der Sicherheitsvorschriften in verschiedenen Branchen zu überwachen. Sie können unsichere Bedingungen automatisch kennzeichnen, indem Sie ungeeignete Ausrüstung erkennen und Warnmeldungen zu diesen Bedingungen erhalten, was die Einhaltung von Vorschriften und die Schulung verbessern kann.

Anerkennung von Prominenten — Erkenne Prominente in deinen Bildern und Videos in allen Kategorien, wie Politiker, Sportler, Schauspieler und Musiker. Sie können Auftritte von Prominenten identifizieren, ohne Namen angeben zu müssen.

Texterkennung — Erkennt und extrahiert Text in Bildern für die visuelle Suche oder das Extrahieren von Metadaten. Das funktioniert bei verschiedenen Schriften und Stilen. Erkennt die Ausrichtung beim Umgang mit Text auf Schildern und Bannern.

Benutzerdefinierte Beschriftungen — Identifizieren Sie benutzerdefinierte Objekte, Konzepte und Szenen, die für geschäftliche Anwendungsfälle spezifisch sind, z. B. die Erkennung von Logos. Sie können benutzerdefinierte Klassifikatoren so trainieren, dass sie mit Nischen- oder proprietären Objekten umgehen können, wodurch die Genauigkeit von Schlüsselobjekten im Vergleich zu allgemeinen Klassifikatoren verbessert wird. Weitere Informationen finden Sie unter Was ist Amazon Rekognition Custom Labels? im Entwicklerhandbuch für Amazon Rekognition Custom Labels.

Vorteile

Integrieren Sie leistungsstarke Bild- und Videoanalysen in Ihre App — Fügen Sie Apps ohne Fachwissen genaue Bild- und Videoanalysen hinzu. Die Amazon Rekognition API ermöglicht Analysen über Deep Learning, ohne dass Kenntnisse über maschinelles Lernen erforderlich sind. Sie können Computer Vision schnell in Web-, Mobil- und Geräte-Apps integrieren.

Deep-Learning-basierte Bild- und Videoanalyse — Analysiert Bilder und Videos mithilfe von Deep Learning für eine hohe Genauigkeit. Amazon Rekognition kann Labels, Objekte, Szenen, Gesichter und Prominente erkennen. Filtern Sie die Ergebnisse, um bestimmte Labels ein- oder auszuschließen.

Skalierbare Bildanalyse — Analysiert Millionen von Bildern, um riesige visuelle Datensätze zu organisieren. Skaliert für die Bewältigung wachsender Bildbibliotheken und des wachsenden Datenverkehrs. Sie müssen keine Kapazität einplanen und zahlen nur für das, was Sie tatsächlich nutzen.

Analysieren und filtern Sie Bilder anhand von Eigenschaften — Analysieren und filtern Sie Bilder nach Eigenschaften wie Qualität, Farbe und visuellem Inhalt und ermitteln Sie Bildschärfe, Helligkeit und Kontrast.

Integration mit anderen AWS Diensten — Amazon Rekognition lässt sich sofort in S3 und Lambda integrieren. Sie können Amazon Rekognition APIs von Lambda aus aufrufen und Bilder in Amazon S3 verarbeiten, ohne Daten zu verschieben. Rekognition bietet integrierte Skalierbarkeit und Sicherheit mithilfe von AWS IAM.

Niedrige Kosten — Pay-as-you-go Preisgestaltung, keine Mindestanforderungen oder Verpflichtungen. Kostenloses Kontingent für den Einstieg verfügbar. Dank gestaffelter Preisgestaltung

können Sie bei steigender Nutzung noch mehr sparen. Kostengünstig im Vergleich zu internen Lösungen.

Einfache Anpassung — Passen Sie die Genauigkeit mit Adaptern an Ihren Anwendungsfall an. Stellen Sie Beispielbilder zur Verfügung, um Adapter zu trainieren. Verbessert die Objekt- und Labelerkennung für bestimmte Domänen. Einfache Möglichkeit, Analysen ohne ML-Fachkenntnisse maßzuschneidern.

Weitere Informationen finden Sie unter Amazon Rekognition FAQs.

Amazon Rekognition und HIPAA-Eignung

Dies ist ein HIPAA-berechtigter Service. Weitere Informationen AWS zum US-amerikanischen Health Insurance Portability and Accountability Act von 1996 (HIPAA) und zur Nutzung von AWS Diensten zur Verarbeitung, Speicherung und Übertragung geschützter Gesundheitsinformationen (PHI) finden Sie unter HIPAA Overview.

Verwenden Sie Amazon Rekognition zum ersten Mal?

Wenn Sie Amazon Rekognition zum ersten Mal verwenden, empfehlen wir Ihnen, nacheinander die folgenden Abschnitte zu lesen:

1. So funktioniert Amazon Rekognition— In diesem Abschnitt werden verschiedene Amazon Rekognition Rekognition-Komponenten vorgestellt, mit denen Sie arbeiten, um ein end-to-end Erlebnis zu schaffen.
2. Erste Schritte mit Amazon Rekognition – In diesem Abschnitt richten Sie Ihr Konto ein, installieren das SDK, das die Sprache Ihrer Wahl widerspiegelt, und testen die Amazon-Rekognition-API. Eine Liste der von Amazon Rekognition unterstützten Programmiersprachen finden Sie unter Rekognition mit einem SDK verwenden AWS.
3. Arbeiten mit Bildern – Dieser Abschnitt enthält Informationen zur Verwendung von Amazon Rekognition mit Bildern, die in Amazon-S3-Buckets gespeichert sind, und Bildern, die von einem lokalen Dateisystem geladen wurden.
4. Arbeiten mit gespeicherten Videoanalysevorgängen – Dieser Abschnitt enthält Informationen zur Verwendung von Amazon Rekognition mit Videos, die in Amazon-S3-Buckets gespeichert sind.
5. Arbeiten mit Streaming-Videoereignissen – Dieser Abschnitt enthält Informationen zur Verwendung von Amazon Rekognition mit Streaming-Videos.

So funktioniert Amazon Rekognition

Amazon Rekognition bietet zwei API-Sets für die visuelle Analyse:

- Amazon Rekognition Image für die Bildanalyse
- Amazon Rekognition Video für die Videoanalyse

Bildanalyse

Mit Amazon Rekognition Image können Ihre Anwendungen:

- Objekte, Szenen und Konzepte in Bildern erkennen
- Erkennen Sie Prominente
- Erkennen Sie Text in einer Vielzahl von Sprachen
- Erkennen Sie explizite, unangemessene oder gewalttätige Inhalte oder Bilder
- Erkennen, analysieren und vergleichen Sie Gesichter und Gesichtsmerkmale wie Alter und Emotionen
- Erkennen Sie das Vorhandensein von persönlicher Schutzausrüstung

Zu den Anwendungsfällen gehören die Verbesserung von Foto-Apps, das Katalogisieren von Bildern und das Moderieren von Inhalten.

Videoanalyse

Mit Amazon Rekognition Video können Ihre Anwendungen:

- Personen und Objekte über Videoframes hinweg verfolgen
- Objekte erkennen
- Erkennen Sie Prominente
- Suchen Sie nach gespeicherten und gestreamten Videos nach Personen von Interesse
- Analysieren Sie Gesichter auf Merkmale wie Alter und Emotionen
- Erkennen Sie explizite, unangemessene oder gewalttätige Inhalte oder Bilder
- Aggregieren und sortieren Sie die Analyseergebnisse nach Zeitstempeln und Segmenten
- Erkennen Sie Personen, Haustiere und Pakete in Streaming-Videos

Zu den Anwendungsfällen gehören Videoanalysen, das Katalogisieren von Videos und das Filtern unangemessener Inhalte.

Schlüsselfunktionen

- Leistungsstarke Deep-Learning-Analyse
- Hochgenaue Erkennung von Objekten, Szenen, Gesichtern und Text
- Einfach zu verwendende API für die Integration in Apps
- Anpassbare Modelle, die auf Ihre Daten abgestimmt sind
- Skalierbare Analyse von Medienbibliotheken

Mit Amazon Rekognition können Sie die Genauigkeit bestimmter Deep-Learning-Modelle verbessern, indem Sie einen benutzerdefinierten Adapter trainieren. Mit Amazon Rekognition Custom Moderation können Sie beispielsweise das grundlegende Bildanalysemodell von Amazon Rekognition anpassen, indem Sie einen benutzerdefinierten Adapter mit Ihren Bildern trainieren. Weitere Informationen finden Sie unter Verbesserung der Genauigkeit mit benutzerdefinierter Moderation.

Die folgenden Abschnitte behandeln die Analysetypen, die Amazon Rekognition bietet, und geben einen Überblick über die Funktionsweise von Amazon Rekognition Image und Amazon Rekognition Video. Ebenfalls abgedeckt wird der Unterschied zwischen nicht gespeicherten und gespeicherten Operationen.

Eine Demo von Amazon Rekognition APIs finden Sie unter Schritt 3: Erste Schritte mit der AWS-CLI und der AWS-SDK-API, wo Sie Rekognition in der Konsole ausprobieren können. AWS

Themen

- Grundlegendes zu den Analysetypen von Rekognition
- Die Bild- und Videooperationen von Rekognition verstehen
- Grundlegendes zu Nicht-Speicher- und Speicher-API-Vorgängen
- Grundlegendes zur Modellversionierung

Grundlegendes zu den Analysetypen von Rekognition

Im Folgenden sind die Analysetypen aufgeführt, die mit der Amazon-Rekognition-Image-API und der Amazon-Rekognition-Video-API durchgeführt werden können. Informationen zu dem finden APIs Sie unterDie Bild- und Videooperationen von Rekognition verstehen.

In der folgenden Tabelle sind die Operationen aufgeführt, die Sie je nach Medientyp, mit dem Sie arbeiten, und Ihrem Anwendungsfall verwenden müssen:

Anwendungsfall	Medientyp	Operationen
Inhalte moderieren	Bilder	DetectModerationLabels, StartMediaAnalysisJob, GetMediaAnalysisJob, ListMediaAnalysisJobs
	Gespeichertes Video	StartContentModeration, GetContentModeration
Identitätsüberprüfung	Bilder	CreateCollection, CreateUser, IndexFaces, AssociateFaces, SearchFacesByImage, SearchUsersByImage
	Gespeichertes Video	CreateCollection, IndexFaces, StartFaceSearch, GetFaceSearch
	Streaming-Video (Echtheit von Gesichtern erkennen)	CreateFaceLivenessSession, StartFaceLivenessSession, GetFaceLivenessSessionResults,
Gesichtsanalyse	Bilder	DetectFaces, CompareFaces
	Gespeichertes Video	StartFaceDetection, GetFaceDetection
	Streaming-Video	CreateStreamProcessor, StartStreamProcessor
Objekt- und Aktivitätserkennung	Bilder	DetectLabels

Anwendungsfall	Medientyp	Operationen
	Gespeicherte Videos	StartLabelDetection, GetLabelDetection
Connected Home	Streaming-Video	StartStreamProcessor
Medienanalyse	Gespeichertes Video	StartSegmentDetection, GetSegmentDetection
Sicherheit am Arbeitsplatz	Bilder	DetectProtectiveEquipment
Texterkennung	Bilder	DetectText
	Video	StartTextDetection, GetTextDetection
Pfade von Personen	Video	StartPersonTracking, GetPersonTracking
Prominentenerkennung	Bilder	RecognizeCelebrities
	Video	StartCelebrityRecognition, GetCelebrityRecognition
Erkennung benutzerdefinierter Label	Bilder	DetectCustomLabels
	Modelltraining	Weitere Informationen finden Sie im Entwicklerhandbuch für benutzerdefinierte Label

Labels

Ein Label bezieht sich auf eines der folgenden Dinge: Objekte (z. B. eine Blume, ein Baum oder ein Tisch), Ereignisse (z. B. eine Hochzeit, ein Schulabschluss oder eine Geburtstagsfeier), Konzepte (z. B. eine Landschaft, der Abend und die Natur) oder Aktivitäten (z. B. Laufen oder Basketball spielen). Amazon Rekognition kann Labels in Bildern und Videos erkennen. Weitere Informationen finden Sie unter Erkennung von Objekten und Konzepten.

Rekognition kann eine große Liste von Labels in Bildern und gespeicherten Videos erkennen. Rekognition kann auch eine kleine Anzahl von Labels in Streaming-Videos erkennen.

Verwenden Sie die folgenden Operationen, um Labels basierend auf Ihrem Anwendungsfall zu erkennen:

- Um Beschriftungen in Bildern zu erkennen: Verwenden Sie DetectLabels. Sie können Bildeigenschaften wie dominante Bildfarben und Bildqualität identifizieren. Um dies zu erreichen, verwenden Sie DetectLabelswith `IMAGE_PROPERTIES` als Eingabeparameter.
- Um Labels in gespeicherten Videos zu erkennen: Verwenden Sie StartLabelDetection. Die Erkennung dominanter Bildfarben und der vorherrschenden Bildqualität wird für gespeicherte Videos nicht unterstützt.
- Um Labels in Streaming-Videos zu erkennen: Verwenden CreateStreamProcessor. Die Erkennung dominanter Bildfarben und der vorherrschenden Bildqualität wird für Streaming-Videos nicht unterstützt.

Mithilfe von inklusiven und exklusiven Filteroptionen können Sie angeben, welche Labeltypen sowohl für die Erkennung von Bild- als auch für gespeicherte Videolabels zurückgegeben werden sollen.

Benutzerdefinierte Labels

Amazon Rekognition Custom Labels kann die Objekte und Szenen in den auf Ihre geschäftlichen Anforderungen zugeschnittenen Bildern identifizieren, indem ein maschinelles Lernmodell geschult wird. Sie können beispielsweise ein Modell schulen, um Logos oder technische Maschinenteile auf einem Fließband zu erkennen.

> **Note**
>
> Informationen über Amazon Rekognition Custom Labels finden Sie im Entwicklungsleitfaden für Amazon Rekognition Custom Labels.

Amazon Rekognition bietet eine Konsole, mit der Sie ein maschinelles Lernmodell erstellen, trainieren, bewerten und ausführen können. Weitere Informationen finden Sie unter Erste Schritte mit Amazon Rekognition Custom Labels im Entwicklerhandbuch für Amazon Rekognition Custom Labels. Sie können auch die Amazon-Rekognition-Custom-Labels-API von verwenden, um ein Modell zu trainieren und auszuführen. Weitere Informationen finden Sie unter Erste Schritte mit dem Amazon Rekognition Custom Labels SDK im Amazon CustomLabels Rekognition Developer Guide.

Um Bilder mit einem trainierten Modell zu analysieren, verwenden Sie. DetectCustomLabels

Erkennung der Echtheit von Gesichtern

Mit Amazon Rekognition Face Liveness können Sie überprüfen, ob ein Benutzer, der sich einer gesichtsbasierten Identitätsprüfung unterzieht, physisch vor der Kamera anwesend ist und kein schlechter Akteur ist, der das Gesicht des Benutzers fälscht. Es erkennt gefälschte Angriffe, die auf eine Kamera gerichtet werden, und Angriffe, die eine Kamera umgehen. Ein Benutzer kann eine Überprüfung der Echtheit von Gesichtern durchführen, indem er ein kurzes Video-Selfie aufnimmt. Für die Überprüfung wird ein Livness-Score zurückgegeben. Die Echtheit von Gesichtern wird anhand einer probabilistischen Berechnung bestimmt, und nach der Überprüfung wird ein Zuverlässigkeitswert (zwischen 0 und 100) zurückgegeben. Je höher der Wert, desto größer ist die Zuverlässigkeit, dass die Person, die den Scheck entgegennimmt, live ist.

Weitere Informationen zu Face Liveness finden Sie unter Echtheit von Gesichtern erkennen.

Gesichtserkennung und -analyse

Amazon Rekognition kann Gesichter in Bildern und gespeicherten Videos erkennen. Mit Amazon Rekognition erhalten Sie Informationen über Folgendes:

- Wo Gesichter in einem Bild oder Video erkannt werden
- Gesichtsmarken wie z. B. die Position der Augen.
- Das Vorhandensein von Gesichtsverdeckungen in Bildern
- Entdeckte Emotionen wie glücklich oder traurig
- Blickrichtung des Blicks einer Person in Bildern

Sie können auch demografische Informationen wie Geschlecht oder Alter interpretieren. Sie können ein Gesicht in einem Bild mit Gesichtern vergleichen, die in einem anderen Bild erkannt wurden. Informationen über Gesichter können auch gespeichert und später wieder abgerufen werden. Weitere Informationen finden Sie unter Erkennung und Analyse von Gesichtern.

Um Gesichter in Bildern zu entdecken, verwenden Sie DetectFaces. Um Gesichter in gespeicherten Videos zu entdecken, verwenden Sie StartFaceDetection.

Gesichtssuche

Amazon Rekognition kann nach Gesichtern suchen. Gesichtsinformationen werden in einem Container indiziert, der als Sammlung bezeichnet wird. Die Gesichtsinformationen in der Sammlung können dann mit Gesichtern abgeglichen werden, die in Bildern, gespeicherten Videos und Streaming-Videos erkannt wurden. Weitere Informationen finden Sie unter Gesichtssuche in einer Sammlung.

Um nach bekannten Gesichtern in Bildern zu suchen, verwenden Sie DetectFaces. Um nach bekannten Gesichtern in gespeicherten Videos zu suchen, verwenden Sie StartFaceDetection. Um nach bekannten Gesichtern in Streaming-Videos zu suchen, verwenden Sie CreateStreamProcessor.

Pfade von Personen

Amazon Rekognition kann die Wege von Personen verfolgen, die in einem gespeicherten Video erkannt werden. Amazon Rekognition Video bietet Pfadverfolgung, Gesichtsdetails und Standortinformationen für Personen, die in einem Video erkannt werden. Weitere Informationen finden Sie unter Pfade von Personen.

Um Personen in gespeicherten Videos zu entdecken, verwenden Sie StartPersonTracking.

Persönliche Schutzausrüstung

Amazon Rekognition kann persönliche Schutzausrüstung (PSA) erkennen, die von Personen auf einem erkannten Bild getragen wird. Amazon Rekognition erkennt Gesichtsbedeckungen, Handbedeckungen und Kopfbedeckungen. Amazon Rekognition sagt voraus, ob ein PSA den entsprechenden Körperteil bedeckt. Sie können sich auch Schutzboxen für erkannte Personen und persönliche Schutzausrüstung besorgen. Weitere Informationen finden Sie unter Erkennung von persönlicher Schutzausrüstung.

Um PPE in Bildern zu erkennen, verwenden Sie DetectProtectiveEquipment.

Prominente

Amazon Rekognition kann Tausende von Prominenten in Bildern und gespeicherten Videos erkennen. Sie können Informationen darüber erhalten, wo sich das Gesicht eines Prominenten auf einem Bild befindet, sowie Infos über Gesichtsmerkmale und die Pose des Gesichts eines Prominenten. Sie können Tracking-Informationen für Prominente erhalten, wie sie in einem

gespeicherten Video erscheinen. Sie können auch weitere Informationen über eine anerkannte Berühmtheit erhalten, z. B. die zum Ausdruck gebrachten Emotionen und die Darstellung des Geschlechts. Weitere Informationen finden Sie unter Erkennen von Prominenten.

Um Prominente in Bildern zu erkennen, verwenden Sie RecognizeCelebrities. Um Prominente in gespeicherten Videos zu erkennen, verwenden Sie StartCelebrityRecognition.

Texterkennung

Amazon Rekognition Text in Image kann Text in Bildern erkennen und in maschinenlesbaren Text umwandeln. Weitere Informationen finden Sie unter Erkennen von Text.

Um Text in Bildern zu erkennen, verwenden Sie DetectText.

Unangemessene oder anstößige Inhalte

Amazon Rekognition kann Bilder und gespeicherte Videos in Hinblick auf Erwachseneninhalte und gewalttätige Inhalte analysieren. Weitere Informationen finden Sie unter Inhalte moderieren.

Um unsichere Bilder zu erkennen, verwenden Sie DetectModerationLabels. Um unsichere gespeicherte Videos zu entdecken, verwenden Sie StartContentModeration.

Anpassung

Bestimmte von Rekognition APIs angebotene Bildanalysen ermöglichen es Ihnen, die Genauigkeit von Deep-Learning-Modellen zu verbessern, indem Sie benutzerdefinierte Adapter erstellen, die auf Ihren eigenen Daten trainiert werden. Adapter sind Komponenten, die sich in das vortrainierte Deep-Learning-Modell von Rekognition integrieren lassen und dessen Genauigkeit durch Fachwissen auf der Grundlage Ihrer Bilder verbessern. Sie trainieren einen Adapter so, dass er Ihren Bedürfnissen entspricht, indem Sie Beispielbilder bereitstellen und mit Anmerkungen versehen.

Nachdem Sie einen Adapter erstellt haben, erhalten Sie einen. AdapterId Sie können dies AdapterId für einen Vorgang bereitstellen, um anzugeben, dass Sie den von Ihnen erstellten Adapter verwenden möchten. Beispielsweise stellen Sie der DetectModerationLabelsAPI den für AdapterId die synchrone Bildanalyse zur Verfügung. Wenn Sie das AdapterId als Teil der Anfrage angeben, verwendet Rekognition es automatisch, um die Vorhersagen für Ihre Bilder zu verbessern. Auf diese Weise können Sie die Funktionen von Rekognition nutzen und es gleichzeitig an Ihre Bedürfnisse anpassen.

Sie haben auch die Möglichkeit, mithilfe der API Prognosen für Bilder in großen Mengen zu erhalten. StartMediaAnalysisJob Weitere Informationen finden Sie unter Massenanalyse.

Sie können die Genauigkeit der Operationen von Rekognition beurteilen, indem Sie Bilder auf die Rekognition-Konsole hochladen und diese Bilder analysieren. Rekognition kommentiert Ihre Bilder mithilfe des ausgewählten Features. Anschließend können Sie die Vorhersagen überprüfen und anhand der verifizierten Vorhersagen ermitteln, welche Labels von der Erstellung eines Adapters profitieren würden.

Derzeit können Sie Adapter mit dem verwenden DetectModerationLabels. Weitere Informationen zur Erstellung und Verwendung von Adaptern finden Sie unter Verbesserung der Genauigkeit mit benutzerdefinierter Moderation.

Massenanalyse

Mit Rekognition Bulk Analysis können Sie eine große Sammlung von Bildern asynchron verarbeiten, indem Sie zusammen mit dem Vorgang eine Manifestdatei verwenden. StartMediaAnalysisJob Weitere Informationen finden Sie unter Massenanalyse.

Die Bild- und Videooperationen von Rekognition verstehen

Amazon Rekognition bietet zwei primäre API-Sets für die Bild- und Videoanalyse:

- Amazon Rekognition Image: Diese API wurde für die Analyse von Bildern entwickelt.
- Amazon Rekognition Video: Diese API konzentriert sich auf die Analyse sowohl gespeicherter als auch gestreamter Videos.

Beide APIs können verschiedene Entitäten wie Gesichter und Objekte erkennen. Ein umfassendes Verständnis der unterstützten Vergleichs- und Erkennungstypen finden Sie im Abschnitt überGrundlegendes zu den Analysetypen von Rekognition.

Funktionsweise von Amazon Rekognition Image

Amazon Rekognition Image-Operationen sind synchron. Die Eingabe und die Antwort erfolgen im JSON-Format. Die Amazon-Rekognition-Image-Operationen analysieren ein Eingabebild, das im JPG- oder PNG-Bildformat vorliegt. Das an eine Amazon-Rekognition-Image-Operation übergebene Bild kann in einem Amazon-S3-Bucket gespeichert werden. Wenn Sie die AWS-CLI nicht verwenden,

können Sie auch Base64-kodierte Bildbytes direkt an einen Amazon Rekognition Rekognition-Vorgang übergeben. Weitere Informationen finden Sie unter Arbeiten mit Bildern.

Amazon-Rekognition-Video-Operationen

Die Amazon Rekognition Video API erleichtert die Analyse von Videos, die entweder in einem Amazon S3 S3-Bucket gespeichert oder über Amazon Kinesis Video Streams gestreamt wurden.

Beachten Sie bei gespeicherten Videooperationen Folgendes:

- Die Operationen sind asynchron.
- Die Analyse muss mit einem Startvorgang eingeleitet werden (z. B. StartFaceDetectionfür die Gesichtserkennung in gespeicherten Videos).
- Der Abschlussstatus der Analyse wird in einem Amazon SNS SNS-Thema veröffentlicht.
- Um die Ergebnisse einer Analyse abzurufen, verwenden Sie den entsprechenden Vorgang „Abrufen“ (z. B. GetFaceDetection).
- Weitere Informationen finden Sie unter Arbeiten mit gespeicherten Videoanalysen.

Für die Analyse von Streaming-Videos:

- Zu den Funktionen gehören die Gesichtssuche in Rekognition-Videosammlungen und die Erkennung von Bezeichnungen (Objekt oder Konzept).
- Analyseergebnisse für Labels werden als Amazon SNS- und Amazon S3 S3-Benachrichtigungen gesendet.
- Die Ergebnisse der Gesichtssuche werden in einen Kinesis-Datenstream ausgegeben.
- Die Verwaltung der Streaming-Videoanalyse erfolgt über einen Amazon Rekognition Video Video-Stream-Prozessor (z. B. erstellen Sie einen Prozessor mit CreateStreamProcessor).
- Weitere Informationen finden Sie unter Arbeiten mit Streaming-Videoereignissen.

Bei jeder Videoanalyse werden Metadaten über das zu analysierende Video sowie eine Auftrags-ID und ein Auftrags-Tag zurückgegeben. Funktionen wie Labelerkennung und Inhaltsmoderation für Videos ermöglichen das Sortieren nach Zeitstempel oder Labelnamen und das Aggregieren von Ergebnissen nach Zeitstempel oder Segment.

Nicht-speicherbasierte und speicherbasierte Operationen

Die Amazon-Rekognition-Operationen sind in die folgenden Kategorien unterteilt.

- Nicht-Speicher-API-Operationen – Bei diesen Operationen behält Amazon Rekognition keinerlei Informationen. Sie stellen Eingabebilder und -videos bereit, die Operation führt die Analyse durch und liefert die Ergebnisse. Dabei werden aber keine Informationen von Amazon Rekognition gespeichert. Weitere Informationen finden Sie unter Nicht speicherbasierte Operationen.
- Speicherbasierte API-Operationen – Der Amazon-Rekognition-Server kann entdeckte Gesichtsinformationen in Containern speichern, die Sammlungen genannt werden. Amazon Rekognition bietet zusätzliche API-Operationen, mit denen Sie die vorhandenen Gesichtsinformationen nach Gesichtsübereinstimmungen durchsuchen können. Weitere Informationen finden Sie unter Speicherbasierte API-Operationen.

Verwendung des AWS-SDK oder HTTP zum Aufrufen von Amazon-Rekognition-API-Vorgängen

Sie können Amazon-Rekognition-API-Operationen entweder mit dem AWS-SDK oder mit HTTP aufrufen. Wenn Sie keinen guten Grund haben, es nicht zu tun, sollten Sie immer das AWS-SDK nutzen. Für die Java-Beispiele in diesem Abschnitt wird das AWS-SDK verwendet. Eine Java-Projektdatei wird nicht bereitgestellt. Sie können jedoch das AWS Toolkit for Eclipse nutzen, um AWS-Anwendungen mit Java zu entwickeln.

Die .NET-Beispiele in diesem Abschnitt verwenden das AWS SDK für .NET. Sie können AWS Toolkit for Visual Studio verwenden, um AWS-Anwendungen mit .NET zu entwickeln. Es enthält hilfreiche Vorlagen und den AWS Explorer zur Bereitstellung von Anwendungen und Verwaltung von Services.

In der API-Referenz in diesem Leitfaden wird der Aufruf von Amazon-Rekognition-Vorgängen über HTTP beschrieben. Informationen zur Java-Referenz finden Sie unter AWS SDK für Java.

Die Amazon-Rekognition-Service-Endpunkte, die Sie verwenden können, sind unter AWS-Regionen und -Endpunkte dokumentiert.

Verwenden Sie POST-HTTP-Operationen für den Aufruf von Amazon Rekognition mit HTTP.

Grundlegendes zu Nicht-Speicher- und Speicher-API-Vorgängen

Amazon Rekognition bietet zwei Arten von API-Operationen. Dies sind nicht speicherbasierte Operationen, bei denen keine Informationen von Amazon Rekognition gespeichert werden, und Speicheroperationen, bei denen bestimmte Gesichtsinformationen von Amazon Rekognition gespeichert werden.

Nicht speicherbasierte Operationen

Amazon Rekognition bietet die folgenden nicht speicherbasierten API-Operationen für Bilder:

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

Amazon Rekognition bietet die folgenden nicht speicherbasierten API-Operationen für Videos:

- StartLabelDetection
- StartFaceDetection
- StartPersonTracking
- StartCelebrityRecognition
- StartContentModeration

Diese werden als nicht speicherbasierte API-Operationen bezeichnet, da beim Aufrufen der Operation keine erkannten Informationen zum Eingabebild in Amazon Rekognition verbleiben. Wie bei allen anderen Amazon-Rekognition-API-Operationen bleiben bei nicht speicherbasierten API-Operationen keine Eingabebild-Bytes erhalten.

Die folgenden Beispielszenarien zeigen, in welchen Fällen die Integration von nicht speicherbasierten API-Operationen in Ihrer Anwendung zum Einsatz kommen kann. Diese Szenarien gehen davon aus, dass Sie eine lokale Bilddatenbank haben.

Example 1: Eine Anwendung, die Bilder in Ihrer örtlichen Bilddatenbank aufspürt, die bestimmte Labels enthalten

Zunächst erkennen Sie mithilfe der Amazon-Rekognition-`DetectLabels`-Operation Labels (Objekte und Konzepte) in jedem der Bilder in Ihrem Repository und erstellen einen clientseitigen Index, wie im Folgenden gezeigt:

```
Label        ImageID

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

Anschließend kann Ihre Anwendung diesen Index durchsuchen und in Ihrer örtlichen Bilddatenbank Bilder mit einem bestimmten Label suchen. Zum Beispiel können so alle Bilder angezeigt werden, die einen Baum enthalten.

Jedem Label, das Amazon Rekognition erkennt, wird ein Zuverlässigkeitswert zugeordnet. Er zeigt an, mit wie hoher Sicherheit das Eingabebild das Label enthält. Je nach Anforderungen an Ihre Anwendung in Bezug auf die Zuverlässigkeit der Erkennung können Sie diesen Zuverlässigkeitswert nutzen, um auf Wunsch die Labels clientseitig zusätzlich zu filtern. Falls Sie beispielsweise extrem präzise Labels benötigen, können Sie nur die Labels mit einem hohen Zuverlässigkeitswert (95% oder höher) filtern und wählen. Falls Ihre Anwendung keinen hohen Zuverlässigkeitswert erfordert, können Sie stattdessen auch die Labels mit einem niedrigeren Zuverlässigkeitswert (rund um 50 %) filtern.

Example 2: Eine Anwendung für die Anzeige von optimierten Bildern von Gesichtern

Zuerst können Sie mithilfe der Amazon-Rekognition-`DetectFaces`-Operation auf jedem Bild in Ihrer örtlichen Bilddatenbank Gesichter erkennen und einen clientseitigen Index zusammenstellen. Die Operation liefert für jedes Gesicht Metadaten. Diese umfassen unter anderem einen Begrenzungsrahmen, wichtige Gesichtsmerkmale (zum Beispiel die Positionen des Mundes oder der Ohren) und Gesichtsattribute (zum Beispiel das Geschlecht). Wie im Folgenden angezeigt können Sie diese Metadaten in einem clientseitigen örtlichen Index speichern:

```
ImageID     FaceID     FaceMetaData

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

In diesem Index ist der Primärschlüssel eine Kombination der `ImageID` und der `FaceID`.

Anschließend können Sie die Informationen im Index nutzen, um die Bilder zu optimieren, sobald die Anwendung diese in Ihrer örtlichen Bilddatenbank aufgerufen hat. Sie können beispielsweise dem Gesicht einen Begrenzungsrahmen hinzufügen oder Gesichtsmerkmale betonen.

Speicherbasierte API-Operationen

Amazon Rekognition Image unterstützt den IndexFacesVorgang, mit dem Sie Gesichter in einem Bild erkennen und Informationen über Gesichtsmerkmale, die in einer Amazon Rekognition Rekognition-Sammlung erkannt wurden, dauerhaft speichern können. Dies ist ein Beispiel für eine speicherbasierte API-Operation, da der Dienst Informationen auf dem Server beibehält.

Amazon Rekognition Image bietet die folgenden Speicher-API-Operationen:

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

Amazon Rekognition Video bietet die folgenden Speicher-API-Operationen:

- StartFaceSearch
- CreateStreamProcessor

Zur Speicherung von Gesichtsinformationen müssen Sie zuerst eine Gesichtersammlung in einer der AWS-Regionen in Ihrem Konto erstellen. Diese Gesichtersammlung wird näher spezifiziert, sobald Sie die `IndexFaces`-Operation aufrufen. Nach der Erstellung einer Gesichtersammlung und der Speicherung der Gesichtsmerkmalinformationen für alle Gesichter können Sie die Sammlung

nach Übereinstimmungen durchsuchen. Beispielsweise können Sie das größte Gesicht in einem Bild erkennen und nach übereinstimmenden Gesichtern in einer Sammlung suchen, indem Sie `searchFacesByImage.` aufrufen.

Gesichtsinformationen, die in Sammlungen durch `IndexFaces` gespeichert sind, sind für Amazon-Rekognition-Video-Operationen zugänglich. So können Sie beispielsweise ein Video nach Personen durchsuchen, deren Gesichter mit jenen einer bestehenden Sammlung übereinstimmen, indem Sie StartFaceSearch aufrufen.

Weitere Informationen zum Erstellen und Verwalten von Sammlungen finden Sie unter Gesichtssuche in einer Sammlung.

> **Note**
>
> Sammlungen speichern Gesichtsvektoren, bei denen es sich um mathematische Darstellungen von Gesichtern handelt. Sammlungen speichern keine Bilder von Gesichtern.

Example 1: Eine Anwendung, die den Zugang zu einem Gebäude authentifiziert

Sie beginnen mit dem Erstellen einer Gesichtersammlung, um gespeicherte Zugangsausweisbilder mithilfe der Operation `IndexFaces` zu speichern. Diese extrahiert Gesichter und speichert sie als durchsuchbare Bildvektoren. Sobald dann ein Mitarbeiter das Gebäude betritt, wird ein Foto seines Gesichts gemacht und an die `SearchFacesByImage`-Operation übermittelt. Wenn bei der Gesichtsübereinstimmung ein ausreichend hoher Ähnlichkeitswert erzielt wird (beispielsweise 99 %), können Sie den Mitarbeiter autorisierten.

Grundlegendes zur Modellversionierung

Amazon Rekognition verwendet Deep-Learning-Modelle, um die Gesichtserkennung und die Suche nach Gesichtern in Sammlungen auszuführen. An der Verbesserung der Genauigkeit der Modelle wird auf der Grundlage von Kundenfeedback und Fortschritten in der Forschung kontinuierlich gearbeitet. Diese Verbesserungen werden als Modell-Updates geliefert. Beispielsweise kann IndexFaces bei Version 1.0 des Modells die 15 größten Gesichter in einem Bild indizieren. Bei neueren Versionen des Modells kann `IndexFaces` die 100 größten Gesichter im Bild indizieren.

Wenn Sie eine neue Sammlung erstellen, wird diese mit der neuesten Version des Modells verknüpft. Um die Genauigkeit zu verbessern, wird das Modell gelegentlich aktualisiert.

Wenn eine neue Version des Modells veröffentlicht wird, geschieht Folgendes:

- Neue Sammlungen, die Sie erstellen, werden mit dem neuesten Modell verknüpft. Gesichter, die Sie mit IndexFaces zu neuen Kollektionen hinzufügen, werden mit dem neuesten Modell erkannt.
- Ihre bestehenden Sammlungen verwenden weiterhin die Version des Modells, mit der sie erstellt wurden. Die in diesen Sammlungen gespeicherten Gesichtsvektoren werden nicht automatisch auf die neueste Version des Modells aktualisiert.
- Neue Gesichter, die zu einer bestehenden Sammlung hinzugefügt werden, werden anhand des Modells erkannt, das bereits mit der Sammlung verknüpft ist.

Verschiedene Versionen des Modells sind nicht miteinander kompatibel. Insbesondere wenn ein Bild in mehrfachen Sammlungen indiziert ist, die verschiedene Versionen des Modells verwenden, sind die Identifikatoren für das gleiche Gesicht unterschiedlich. Wenn ein Bild in mehrfachen Sammlungen indiziert ist, die das gleiche Modell verwenden, sind auch die Identifikatoren gleich.

Ihre Anwendung könnte mit Kompatibilitätsproblemen konfrontiert sein, wenn Ihr Sammlungsmanagement keine Aktualisierungen des Modells berücksichtigt. Sie können die Version des von einer Sammlung verwendeten Modells über das Feld `FaceModelVersion` ermitteln, das in der Antwort einer Sammlungsoperation zurückgegeben wird (z. B. `CreateCollection`). Sie können die Modellversion einer vorhandenen Sammlung abrufen, indem Sie DescribeCollection aufrufen. Weitere Informationen finden Sie unter Beschreiben einer Sammlung.

Vorhandene Gesichtsvektoren in einer Sammlung können nicht auf eine spätere Version des Modells aktualisiert werden. Da Amazon Rekognition keine Quellbildbytes speichert, kann es Bilder nicht automatisch neu indizieren, indem es eine neuere Version des Modells verwendet.

Um das neueste Modell für die Gesichter zu verwenden, die in einer bestehenden Sammlung gespeichert sind, erstellen Sie eine neue Sammlung (CreateCollection) und indizieren Sie die Quellbilder erneut in die neue Sammlung (`Indexfaces`). Sie müssen alle Gesichtsidentifikatoren aktualisieren, die von Ihrer Anwendung gespeichert werden, da sich die Gesichtsidentifikatoren in der neuen Sammlung von den Gesichtsidentifikatoren in der alten Sammlung unterscheiden. Wenn Sie die alte Sammlung nicht mehr benötigen, können Sie ihn mit DeleteCollection löschen.

Zustandslose Operationen, wie z. B. DetectFaces, verwenden die neueste Version des Modells.

Erste Schritte mit Amazon Rekognition

Dieser Abschnitt enthält alle Themen für die ersten Schritte mit Amazon Rekognition. Falls Sie Amazon Rekognition zum ersten Mal benutzen, empfehlen wir, sich zuerst mit den Konzepten und der Terminologie in So funktioniert Amazon Rekognition vertraut zu machen.

Sie müssen ein AWS-Konto einrichten und eine AWS-Konto-ID haben, ehe Sie auf Rekognition zugreifen können. Sie sollten auch einen Benutzer erstellen, damit das Amazon-Rekognition-System feststellen kann, ob Sie über die erforderlichen Berechtigungen für den Zugriff auf seine Ressourcen verfügen.

Nachdem Sie Ihre Konten erstellt haben, sollten Sie und installieren AWS CLI und konfigurieren AWS SDKs. AWS CLI Mit dem können Sie über die Befehlszeile mit Amazon Rekognition und anderen Diensten interagieren, während Sie Programmiersprachen wie Java und Python verwenden können, um mit Amazon Rekognition zu interagieren. AWS SDKs

Sobald Sie das AWS CLI und eingerichtet haben AWS SDKs, können Sie sich einige Beispiele ansehen, wie Sie beide verwenden können. Sie können sich auch einige Beispiele für die Interaktion mit Amazon Rekognition über die Konsole ansehen.

Mit der Amazon Rekognition Rekognition-Konsole können Sie die Workflows Bulk Analysis und Custom Moderation verwenden. Neben diesen Funktionen können Sie die Konsole auch verwenden, um eine Reihe verschiedener Amazon Rekognition-Funktionen zu testen. Beachten Sie, dass die Verwendung der API erforderlich ist, um die meisten Funktionen von Amazon Rekognition nutzen zu können.

Themen

- Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers
- Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs
- Schritt 3: Erste Schritte mit der AWS CLIAWS SDK-API
- Schritt 4: Erste Schritte mit der Amazon-Rekognition-Konsole

Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers

Bevor Sie Amazon Rekognition zum ersten Mal verwenden können, müssen Sie die folgenden Aufgaben ausführen:

1. Eröffnen Sie ein AWS Konto.
2. Erstellen eines Benutzers.

In diesem Abschnitt des Entwicklerhandbuchs wird erklärt, warum und wie Sie ein AWS -Konto und einen Benutzer erstellen.

Themen

- Erstellen Sie ein AWS Konto und einen Benutzer

Erstellen Sie ein AWS Konto und einen Benutzer

AWS-Konten

Bei der Registrierung für Amazon Web Services (AWS) wird Ihr AWS-Konto automatisch für alle Dienste in AWS einschließlich Amazon Rekognition registriert. Berechnet werden Ihnen aber nur die Services, die Sie nutzen.

Mit Amazon Rekognition zahlen Sie nur für die Ressourcen, die Sie wirklich nutzen.

Wenn Sie ein neuer AWS Kunde sind, können Sie kostenlos mit Amazon Rekognition beginnen. Weitere Informationen finden Sie unter AWS Free Usage Tier (kostenloses Nutzungskontingent für AWS).

Anweisungen zur Kontoerstellung finden Sie im nächsten Melden Sie sich an für ein AWS-Konto-Abschnitt.

Wenn Sie bereits ein AWS Konto haben, überspringen Sie die Kontoeinrichtung und erstellen Sie einen Administratorbenutzer.

Benutzer

Services in AWS, wie z. B. Amazon Rekognition, fordern beim Zugriff die Eingabe von Anmeldeinformationen. Der Service kann feststellen, ob Sie über die Berechtigung für den Zugriff auf die Ressourcen im Besitz dieses Service verfügen.

Sie können Zugangsschlüssel für Ihr AWS Konto erstellen, um auf die Konsole zuzugreifen, AWS CLI oder APIs wenn Sie die Konsole verwenden, benötigen Sie Ihr Passwort. Wir raten Ihnen jedoch davon ab, mit den Anmeldeinformationen des Root-Benutzers für Ihr AWS-Konto auf AWS zuzugreifen. Stattdessen empfehlen wir, AWS Identity and Access Management (IAM) zu verwenden, um einen Administratorbenutzer zu erstellen.

Sie können dann auf AWS zugreifen, indem Sie eine spezielle URL und die Anmeldeinformationen dieses administrativen Benutzers verwenden.

Wenn Sie sich bei AWS angemeldet, aber noch keinen Benutzer für sich selbst angelegt haben, können Sie dies über die IAM-Konsole tun. Anweisungen zum Erstellen eines Administratorbenutzers finden Sie im nächsten Erstellen eines Benutzers mit Administratorzugriff-Abschnitt.

Melden Sie sich an für ein AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/die Anmeldung.
2. Folgen Sie den Online-Anweisungen.

   Während der Anmeldung erhalten Sie einen Telefonanruf oder eine Textnachricht und müssen einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können Ihre aktuellen Kontoaktivitäten jederzeit einsehen und Ihr Konto verwalten, indem Sie zu https://aws.amazon.com/gehen und Mein Konto auswählen.

Erstellen eines Benutzers mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS Management Consoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung Benutzerhandbuch zu.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen finden Sie unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Benutzers mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.
2. Gewähren Sie einem Administratorbenutzer im IAM Identity Center Benutzerzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie IAM-Identity-Center-Verzeichnis im Benutzerhandbuch unter Benutzerzugriff mit der Standardeinstellung konfigurieren.AWS IAM Identity Center

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Access-Portal.

Weiteren Benutzern Zugriff zuweisen

1. Erstellen Sie im IAM-Identity-Center einen Berechtigungssatz, der den bewährten Vorgehensweisen für die Anwendung von geringsten Berechtigungen folgt.

   Anweisungen hierzu finden Sie unter Berechtigungssatz erstellen im AWS IAM Identity Center Benutzerhandbuch.
2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

   Eine genaue Anleitung finden Sie unter Gruppen hinzufügen im AWS IAM Identity Center Benutzerhandbuch.

Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs

Themen

- Erteilen programmgesteuerten Zugriffs
- Rekognition mit einem SDK verwenden AWS

Die folgenden Schritte zeigen Ihnen, wie Sie das AWS Command Line Interface (AWS CLI) installieren und AWS SDKs was in den Beispielen in dieser Dokumentation verwendet wird. Es gibt verschiedene Möglichkeiten, AWS SDK-Aufrufe zu authentifizieren. Bei den Beispielen in diesem Handbuch wird davon ausgegangen, dass Sie ein Standard-Anmeldeinformationsprofil für den Aufruf von AWS CLI Befehlen und AWS SDK-API-Operationen verwenden.

Eine Liste der verfügbaren AWS Regionen finden Sie unter Regionen und Endpunkte in der Allgemeine Amazon Web Services-Referenz.

Befolgen Sie die Schritte zum Herunterladen und Konfigurieren der AWS SDKs.

Um das AWS CLI und das einzurichten AWS SDKs

1. Laden Sie das und das herunter AWS CLIund installieren Sie AWS SDKs es, das Sie verwenden möchten. Dieses Handbuch enthält Beispiele für Java AWS CLI, Python, Ruby, Node.js, PHP, .NET und JavaScript. Informationen zur Installation AWS SDKs finden Sie unter Tools für Amazon Web Services.

2. Erstellen Sie einen Zugriffsschlüssel für den Benutzer, den Sie unter Erstellen Sie ein AWS Konto und einen Benutzer erstellt haben.

   a. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.

   b. Klicken Sie im Navigationsbereich auf Users (Benutzer).

   c. Wählen Sie einen Namen für den Benutzer aus, den Sie unter Erstellen Sie ein AWS Konto und einen Benutzer erstellt haben.

   d. Wechseln Sie zur Registerkarte Sicherheitsanmeldeinformationen.

   e. Wählen Sie Zugriffsschlüssel erstellen aus. Wählen Sie dann CSV-Datei herunterladen, um die Zugriffsschlüssel-ID und den geheimen Zugriffsschlüssel in einer CSV-Datei auf dem Computer zu speichern. Speichern Sie die Datei an einem sicheren Ort. Sie haben keinen Zugriff auf den geheimen Zugriffsschlüssel mehr, nachdem das Dialogfeld geschlossen wird. Nachdem Sie die CSV-Datei heruntergeladen haben, klicken Sie auf Close.

3. Wenn Sie das installiert haben AWS CLI, können Sie die Anmeldeinformationen und die Region für die meisten konfigurieren, AWS SDKs indem Sie sie `aws configure` an der Befehlszeile eingeben. Befolgen Sie andernfalls die folgenden Anweisungen.

4. Gehen Sie auf Ihrem Computer zu Ihrem Stammverzeichnis und erstellen Sie ein `.aws` Verzeichnis. Bei Unix-basierten Systemen (wie Linux oder macOS) befindet es sich an der folgenden Position:

   ```
   ~/.aws
   ```

   Unter Windows befindet es sich an der folgenden Position:

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

5. Erstellen Sie in dem Verzeichnis `.aws` eine neue Datei namens `credentials`.

6. Öffnen Sie die CSV-Datei mit den Anmeldeinformationen, die Sie in Schritt 2 erstellt haben, und kopieren Sie ihren Inhalt mit dem folgenden Format in die Datei `credentials`:

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

Ersetzen Sie your_access_key_id und your_secret_access_key durch Ihre Zugriffsschlüssel-ID und den geheimen Zugriffsschlüssel.

7. Speichern Sie die Datei `Credentials`, und löschen Sie die CSV-Datei.
8. Erstellen Sie in dem Verzeichnis `.aws` eine neue Datei namens `config`.
9. Öffnen Sie die Datei `config`, und geben Sie Ihre Region im folgenden Format ein.

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

Ersetzen Sie die gewünschte AWS-Region (zum Beispiel `us-west-2`) für your_aws_region.

> **Note**
>
> Wenn Sie keine Region auswählen, wird standardmäßig „us-east-1“ verwendet.

10. Speichern Sie die `config`-Datei.

Erteilen programmgesteuerten Zugriffs

Sie können die Codebeispiele AWS CLI und die Codebeispiele in diesem Handbuch auf Ihrem lokalen Computer oder in anderen AWS Umgebungen, z. B. einer Amazon Elastic Compute Cloud-Instance, ausführen. Um die Beispiele auszuführen, müssen Sie Zugriff auf die AWS SDK-Operationen gewähren, die in den Beispielen verwendet werden.

Themen

- Ausführen von Code auf Ihrem lokalen Computer
- Code in AWS Umgebungen ausführen

Ausführen von Code auf Ihrem lokalen Computer

Um Code auf einem lokalen Computer auszuführen, empfehlen wir, dass Sie kurzfristige Anmeldeinformationen verwenden, um einem Benutzer Zugriff auf AWS SDK-Operationen zu gewähren. Spezifische Informationen zum Ausführen von AWS CLI und zu Codebeispielen auf einem lokalen Computer finden Sie unterVerwenden eines Profils auf Ihrem lokalen Computer.

Benutzer benötigen programmgesteuerten Zugriff, wenn sie mit AWS außerhalb des AWS Management Console interagieren möchten. Die Art und Weise, wie programmatischer Zugriff gewährt wird, hängt von der Art des Benutzers ab, der zugreift. AWS

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zur Verwendung AWS IAM Identity Center im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch AWS SDKs und im Tools-Ref erenzhandbuch.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Folgen Sie den Anweisungen unter Verwenden temporäre r Anmeldeinformationen mit AWS Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
	an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	• Informationen dazu AWS CLI finden Sie unter Authentifizierung mithilfe von IAM-Benutzeranmeldeinformationen im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. • Weitere Informationen finden Sie unter Verwaltung von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch. AWS APIs

Verwenden eines Profils auf Ihrem lokalen Computer

Sie können die Codebeispiele AWS CLI und die Codebeispiele in diesem Handbuch mit den kurzfristigen Anmeldeinformationen ausführen, die Sie in Ausführen von Code auf Ihrem lokalen Computer erstellen. Um die Anmeldeinformationen und andere Einstellungsinformationen abzurufen, verwenden die Beispiele ein Profil mit dem Namen `profile-name`. Zum Beispiel:

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

Der Benutzer, den das Profil repräsentiert, muss berechtigt sein, die Rekognition SDK-Operationen und andere AWS SDK-Operationen, die in den Beispielen benötigt werden, aufzurufen.

Um ein Profil zu erstellen, das mit den Codebeispielen AWS CLI und -Codebeispielen funktioniert, wählen Sie eine der folgenden Optionen. Stellen Sie sicher, dass der Name des von Ihnen erstellten Profils `profile-name` lautet.

- Von IAM verwaltete Benutzer — Folgen Sie den Anweisungen unter Zu einer IAM-Rolle wechseln (AWS CLI).
- Personalidentität (Benutzer verwaltet von AWS IAM Identity Center) — Folgen Sie den Anweisungen unter Konfiguration der zu verwendenden AWS-CLI AWS IAM Identity Center. Für die Codebeispiele empfehlen wir die Verwendung einer integrierten Entwicklungsumgebung (IDE), die das AWS-Toolkit unterstützt und die Authentifizierung über das IAM Identity Center ermöglicht. Die Java-Beispiele finden Sie unter Mit Java entwickeln. Die Python-Beispiele finden Sie unter Mit Python entwickeln. Weitere Informationen finden Sie unter Anmeldeinformation für das IAM Identity Center.

> **Note**
>
> Sie können Code verwenden, um kurzfristige Anmeldeinformationen zu erhalten. Weitere Informationen finden Sie unter Wechseln zu einer IAM-Rolle (AWS API). Rufen Sie für IAM Identity Center die kurzfristigen Anmeldeinformationen für eine Rolle ab, indem Sie den Anweisungen unter Abrufen von IAM-Rollenanmeldeinformationen für den CLI-Zugriff folgen.

Code in AWS Umgebungen ausführen

Sie sollten keine Benutzeranmeldedaten verwenden, um AWS SDK-Aufrufe in AWS Umgebungen zu signieren, wie z. B. Produktionscode, der in einer AWS Lambda Funktion ausgeführt wird. Stattdessen konfigurieren Sie eine Rolle, die die Berechtigungen definiert, die Ihr Code benötigt. Anschließend weisen Sie die Rolle der Umgebung zu, in der Ihr Code ausgeführt wird. Wie Sie die Rolle zuordnen und temporäre Anmeldeinformationen verfügbar machen, hängt von der Umgebung ab, in der Ihr Code ausgeführt wird:

- AWS Lambda function — Verwenden Sie die temporären Anmeldeinformationen, die Lambda Ihrer Funktion automatisch zur Verfügung stellt, wenn sie die Ausführungsrolle der Lambda-Funktion übernimmt. Die Anmeldeinformationen sind in den Lambda-Umgebungsvariablen verfügbar. Sie müssen kein Profil angeben. Weitere Informationen finden Sie unter Lambda-Ausführungsrolle.
- Amazon EC2 — Verwenden Sie den Anbieter für Anmeldeinformationen für EC2 Amazon-Instance-Metadaten. Der Anbieter generiert und aktualisiert automatisch Anmeldeinformationen für Sie

mithilfe des EC2 Amazon-Instance-Profils, das Sie der EC2 Amazon-Instance zuordnen. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf EC2 Amazon-Instances ausgeführt werden

- Amazon Elastic Container Service — Verwenden Sie den Anbieter für Container-Anmeldeinformationen. Amazon ECS sendet und aktualisiert Anmeldeinformationen an einen Metadaten-Endpunkt. Eine von Ihnen angegebene Aufgaben-IAM-Rolle bietet eine Strategie für die Verwaltung der Anmeldeinformationen, die Ihre Anwendung verwendet. Weitere Informationen finden Sie unter Interagieren mit AWS-Services.

Weitere Informationen zu Anbietern von Anmeldeinformationen finden Sie unter Standardisierte Anmeldeinformationsanbieter.

Rekognition mit einem SDK verwenden AWS

AWS Software Development Kits (SDKs) sind für viele gängige Programmiersprachen verfügbar. Jedes SDK bietet eine API, Codebeispiele und Dokumentation, die es Entwicklern erleichtern, Anwendungen in ihrer bevorzugten Sprache zu erstellen.

SDK-Dokumentation	Codebeispiele
AWS SDK für C++	AWS SDK für C++ Codebeispiele
AWS CLI	AWS CLI Codebeispiele
AWS SDK für Go	AWS SDK für Go Codebeispiele
AWS SDK für Java	AWS SDK für Java Codebeispiele
AWS SDK für JavaScript	AWS SDK für JavaScript Codebeispiele
AWS SDK für Kotlin	AWS SDK für Kotlin Codebeispiele
AWS SDK für .NET	AWS SDK für .NET Codebeispiele
AWS SDK für PHP	AWS SDK für PHP Codebeispiele
AWS -Tools für PowerShell	AWS -Tools für PowerShell Codebeispiele
AWS SDK für Python (Boto3)	AWS SDK für Python (Boto3) Codebeispiele

SDK-Dokumentation	Codebeispiele
AWS SDK für Ruby	AWS SDK für Ruby Codebeispiele
AWS SDK for Rust	AWS SDK for Rust Codebeispiele
AWS SDK für SAP ABAP	AWS SDK für SAP ABAP Codebeispiele
AWS SDK for Swift	AWS SDK for Swift Codebeispiele

Für Beispiele, die sich speziell auf Rekognition beziehen, siehe Codebeispiele für Amazon Rekognition mit AWS SDKs.

> **Beispiel für die Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie ein Codebeispiel an, indem Sie unten den Link Feedback geben auswählen.

Schritt 3: Erste Schritte mit der AWS CLIAWS SDK-API

Nachdem Sie das AWS CLI und AWS SDKs , das Sie verwenden möchten, eingerichtet haben, können Sie Anwendungen erstellen, die Amazon Rekognition verwenden. Die meisten Interaktionen mit Amazon Rekognition erfolgen über die API-Operationen, obwohl einige wenige Funktionen von Amazon Rekognition über Konsolen-Workflows verfügen.

Die folgenden Themen zeigen Ihnen, wie Sie mit Amazon Rekognition Image und Amazon Rekognition Video über das oder das beginnen können. AWS CLI AWS SDKs

- Arbeiten mit Bildern— Behandelt den Prozess der Bildanalyse mit Amazon Rekognition Image.
- Arbeiten mit gespeicherten Videoanalysevorgängen- Behandelt den Prozess der Analyse von gespeicherten, nicht gestreamten Videos mit Amazon Rekognition Video.
- Arbeiten mit Streaming-Videoereignissen- Behandelt den Prozess der Analyse von Streaming-Videos mit Amazon Rekognition Video.

Die oben aufgeführten Abschnitte enthalten Beispiele für die Verwendung von. AWS CLI Wenn Sie beabsichtigen AWS CLI, das zu verwenden, finden Sie im folgenden Abschnitt Informationen zur Formatierung Ihrer API-Aufrufe.

Formatieren der AWS CLI Beispiele

Die AWS CLI Beispiele in diesem Handbuch sind für das Linux-Betriebssystem formatiert. Um die Beispiele mit Microsoft Windows anzuwenden, müssen Sie die JSON-Formatierung des `--image`-Parameters ändern und die Zeilenumbrüche von Backslashes (\) zu Einfügezeichen (^) ändern. Weitere Informationen zur JSON-Formatierung finden Sie unter Angeben von Parameterwerten für die AWS Command Line Interface.

Im Folgenden finden Sie einen AWS CLI Beispielbefehl, der für Microsoft Windows formatiert ist (beachten Sie, dass diese Befehle nicht unverändert ausgeführt werden, es handelt sich lediglich um Formatierungsbeispiele):

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

Sie können auch eine JSON-Kurznotation bereitstellen, die sowohl für Microsoft Windows als auch für Linux geeignet ist.

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

Weitere Informationen finden Sie unter Verwenden von Syntax-Kurznotation mit der AWS Command Line Interface.

Nächster Schritt

Schritt 4: Erste Schritte mit der Amazon-Rekognition-Konsole

Schritt 4: Erste Schritte mit der Amazon-Rekognition-Konsole

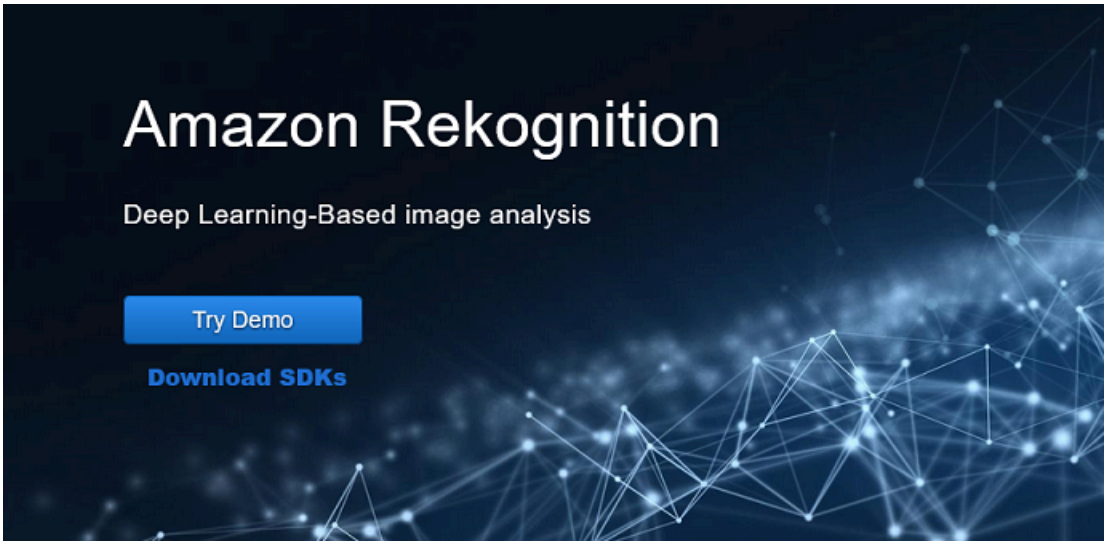
Mit der Amazon Rekognition Rekognition-Konsole können Sie Ressourcen im Zusammenhang mit Rekognition Custom Labels und der Custom Moderation-Funktion verwalten. Die Konsole bietet nur Demos anderer Rekognition-Funktionen.

In diesem Abschnitt erfahren Sie, wie Sie Unterkategorien der Amazon-Rekognition-Funktionen wie beispielsweise Objekt- und Szenenerkennung, Gesichtsanalyse und Vergleich von Gesichtern auf mehreren Bildern nutzen. Weitere Informationen finden Sie unter So funktioniert Amazon Rekognition. Sie können auch die Amazon Rekognition API verwenden, um Objekte und Szenen AWS CLI zu erkennen, Gesichter zu erkennen und Gesichter zu vergleichen und zu suchen. Weitere Informationen finden Sie unter Schritt 3: Erste Schritte mit der AWS CLIAWS SDK-API.

In diesem Abschnitt erfahren Sie auch, wie Sie mithilfe der Rekognition-Konsole aggregierte CloudWatch Amazon-Metriken für Rekognition einsehen können.

Themen

- Konsolenberechtigungen einrichten
- Übung 1: Erkennen von Objekten und Szenen (Konsole)
- Übung 2: Analyse von Gesichtern auf einem Bild (Konsole)
- Übung 3: Vergleich von Gesichtern auf Bildern (Konsole)
- Übung 4: Anzeigen von Gesamtmetriken (Konsole)



Konsolenberechtigungen einrichten

Um die Rekognition-Konsole verwenden zu können, benötigen Sie die entsprechenden Berechtigungen für die Rolle oder das Konto, das auf die Konsole zugreift. Bei einigen Operationen erstellt Rekognition automatisch einen Amazon-S3-Bucket zum Speichern von Dateien, die während des Betriebs verarbeitet wurden. Wenn Sie Ihre Trainingsdateien in einem anderen Bucket als diesem Konsolen-Bucket speichern möchten, benötigen Sie zusätzliche Berechtigungen.

Erlauben von Konsolenzugriff

Um die Rekognition-Konsole zu verwenden, können Sie eine IAM-Richtlinie wie die folgende verwenden, die Amazon-S3- und die Rekognition-Konsole abdeckt. Informationen zum Zuweisen von Berechtigungen finden Sie unter Zuweisen von Berechtigungen.

```
            {
    "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": "*"
        }
    ]
}
```

Zugreifen auf externe Amazon-S3-Bucket

Wenn Sie die Rekognition-Konsole zum ersten Mal in einer neuen AWS Region öffnen, erstellt Rekognition einen Bucket (Konsolen-Bucket), der zum Speichern von Projektdateien verwendet wird. Alternativ können Sie Ihren eigenen Amazon-S3-Bucket (externer Bucket) verwenden, um die Bilder oder die Manifestdatei auf die Konsole hochzuladen. Um einen externen Bucket zu verwenden, fügen Sie der vorherigen Richtlinie den folgenden Richtlinienblock hinzu.

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

Zuweisen von Berechtigungen

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center (Nachfolger von AWS Single Sign-On):

Erstellen Sie einen Berechtigungssatz. Folgen Sie den Anweisungen unter Erstellen eines Berechtigungssatzes im Benutzerhandbuch zu AWS IAM Identity Center (Nachfolger von AWS Single Sign-On).

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.

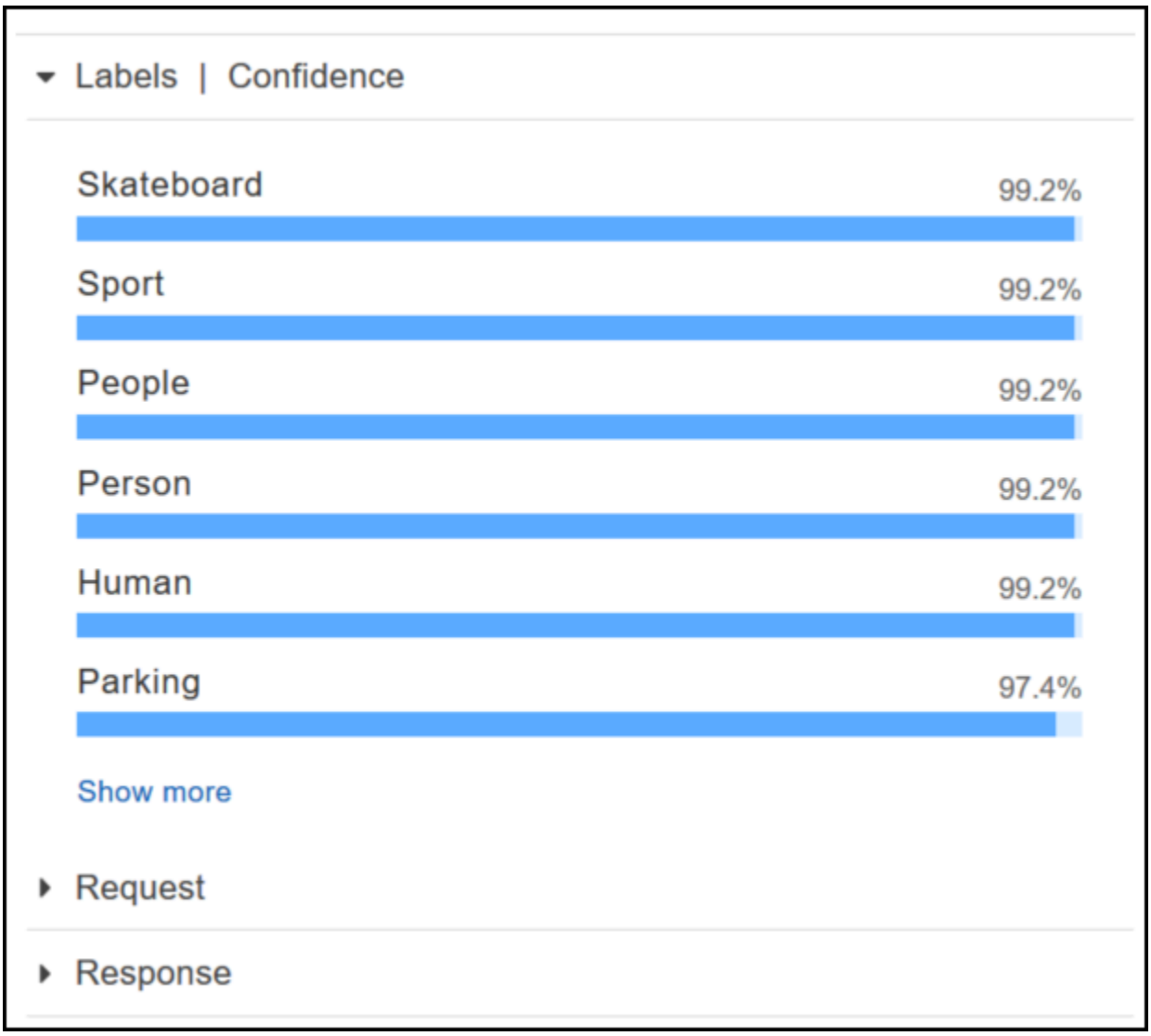
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

Übung 1: Erkennen von Objekten und Szenen (Konsole)

Dieser Abschnitt zeigt, wie die Erkennung von Objekten und Szenen in Amazon Rekognition allgemein funktioniert. Wenn Sie ein Bild als Eingabebild festlegen, erkennt der Dienst die Objekte und Konzepte auf dem Bild und gibt sie zusammen mit einem Zuverlässigkeitswert in Prozent für jedes Objekt und jede Szene zurück.

Auf dem abgebildeten Foto erkennt Amazon Rekognition beispielsweise die folgenden Objekte und Konzepte: Skateboard, Sport, Person, Auto und Fahrzeug.

Amazon Rekognition gibt auch einen Zuverlässigkeitswert für jedes auf dem Beispielbild erkannte Objekt zurück, siehe folgende Beispielantwort.



Wählen Sie Mehr anzeigen im Bereich Labels | Zuverlässigkeit aus, um alle Zuverlässigkeitswerte für die Antwort anzuzeigen.

Als Referenz können Sie auch die Anforderung an die API sowie die Antwort der API ansehen.

Anforderung

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

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

Antwort

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


Weitere Informationen finden Sie unter So funktioniert Amazon Rekognition.

Erkennt Objekte und Konzepte auf einem von Ihnen bereitgestellten Bild

Sie können eines Ihrer eigenen Bilder hochladen oder die URL für ein Bild als Eingabe in der Amazon-Rekognition-Konsole bereitstellen. Amazon Rekognition gibt das Objekt und die Konzepte sowie Zuverlässigkeitswerte für alle Objekte und Konzepte zurück, die es auf Ihrem Bild erkennt.

> **Note**
>
> Das Bild darf höchstens 5MB groß sein und muss im JPEG- oder PNG-Format vorliegen.

Erkennung von Objekten und Konzepten auf einem von Ihnen bereitgestellten Bild

1. Öffnen Sie die Amazon Rekognition Rekognition-Konsole unter. https://console.aws.amazon.com/rekognition/
2. Wählen Sie Labelerkennung.
3. Führen Sie eine der folgenden Aktionen aus:
   - Laden Sie ein Bild hoch – Wählen Sie Hochladen aus, navigieren Sie zum Ort, an dem Sie Ihr Bild gespeichert haben, und wählen Sie das Bild aus.
   - Verwenden Sie eine URL – Geben Sie die URL in das Textfeld ein und wählen Sie anschließend Los aus.
4. Zeigen Sie den Zuverlässigkeitswert jedes erkannten Labels im Bereich Labels | Zuverlässigkeit an.

Weitere Bildanalyseoptionen finden Sie unter the section called “Arbeiten mit Bildern”.

Objekte und Personen in einem von Ihnen bereitgestellten Video erkennen

Sie können ein Video hochladen, das Sie als Eingabe in der Amazon-Rekognition-Konsole bereitstellen. Amazon Rekognition gibt die Personen, Objekte und Labels zurück, die im Video erkannt wurden.

> **Note**
>
> Das Demo-Video darf nicht länger als eine Minute oder größer als 30 MB sein. Sie muss im MP4 Dateiformat vorliegen und mit dem H.264-Codec codiert sein.

Um Objekte und Personen in einem von Ihnen bereitgestellten Video erkennen

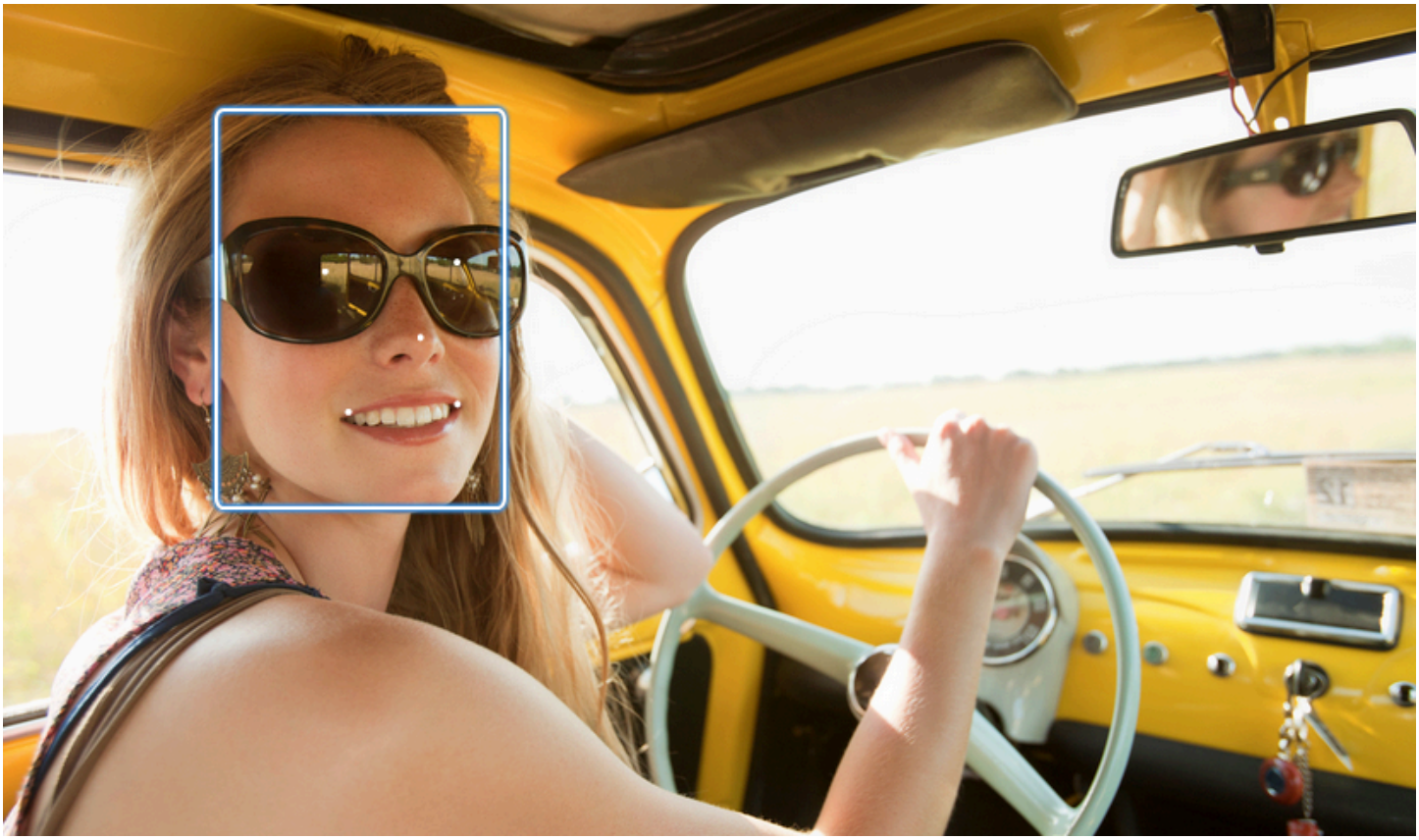
1. Öffnen Sie die Amazon Rekognition Rekognition-Konsole unter. https://console.aws.amazon.com/rekognition/
2. Wählen Sie in der Navigationsleiste die Option Gespeicherte Videoanalyse aus.
3. Wählen Sie unter Wählen Sie ein Beispiel aus oder laden Sie Ihr eigenes Video hoch aus dem Drop-down-Menü aus.
4. Ziehen Sie Ihr Video per Drag-and-Drop oder wählen Sie Ihr Video von dem Ort aus, an dem Sie es gespeichert haben.

Weitere Optionen für die Videoanalyse finden Sie unter the section called “Arbeiten mit gespeicherten Videoanalysevorgängen” oder the section called “Arbeiten mit Streaming-Videoereignissen”.

Übung 2: Analyse von Gesichtern auf einem Bild (Konsole)

In diesem Abschnitt erfahren Sie, wie Sie die Amazon-Rekognition-Konsole nutzen, um Gesichter auf einem Bild zu erkennen und Gesichtsmerkmale zu analysieren. Wenn Sie ein Bild bereitstellen, das ein Gesicht als Eingabe enthält, erkennt der Dienst das Gesicht auf dem Bild. Er analysiert die Gesichtsmerkmale und gibt einen Zuverlässigkeitswert für das auf dem Bild erkannte Gesicht und die Gesichtsmerkmale zurück. Weitere Informationen finden Sie unter So funktioniert Amazon Rekognition.

Beispiel: Wenn Sie das folgende Beispielbild als Eingabe wählen, erkennt es Amazon Rekognition als Gesicht und gibt Zuverlässigkeitswerte für das Gesicht und die erkannten Gesichtsmerkmale zurück.



Nachfolgend ist die Beispielantwort angezeigt.

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

Falls mehrere Gesichter auf dem Eingabebild angezeigt sind, erkennt Amazon Rekognition bis zu 100 Gesichter pro Bild. Jedes erkannte Gesicht wird mit einem Quadrat markiert. Wenn Sie auf den mit einem Quadrat markierten Bereich auf einem Gesicht klicken, zeigt Amazon Rekognition im Bereich Gesichter | Zuverlässigkeit den Zuverlässigkeitswert für dieses Gesicht und dessen Attribute an.

Analyse von Gesichtern auf einem von Ihnen bereitgestellten Bild

Sie können Ihr eigenes Bild hochladen oder die URL für das Bild über die Amazon-Rekognition-Konsole bereitstellen.

> **Note**
>
> Das Bild darf höchstens 5MB groß sein und muss im JPEG- oder PNG-Format vorliegen.

Analyse von Gesichtern auf einem von Ihnen bereitgestellten Bild

1. Öffnen Sie die Amazon Rekognition Rekognition-Konsole unter. https://console.aws.amazon.com/rekognition/
2. Wählen Sie Gesichtsanalyse.
3. Führen Sie eine der folgenden Aktionen aus:
   - Laden Sie ein Bild hoch – Wählen Sie Hochladen aus, navigieren Sie zum Ort, an dem Sie Ihr Bild gespeichert haben, und wählen Sie das Bild aus.
   - Verwenden Sie eine URL – Geben Sie die URL in das Textfeld ein und wählen Sie anschließend Los aus.
4. Den Zuverlässigkeitswert für eines der erkannten Gesichter und dessen Gesichtsmerkmale sehen Sie im Bereich Gesichter | Zuverlässigkeit.
5. Wenn mehrere Gesichter auf einem Bild abgebildet sind, wählen Sie eines der anderen Gesichter, um dessen Attribute und Bewertungen anzuzeigen.

Übung 3: Vergleich von Gesichtern auf Bildern (Konsole)

In diesem Abschnitt erfahren Sie, wie Sie mithilfe der Amazon-Rekognition-Konsole Gesichter auf einer Reihe von Bildern vergleichen, auf denen mehrere Gesichter abgebildet sind. Wenn Sie ein Referenzgesicht als Quelle und ein Gesichtervergleich als Zielbild angeben, vergleicht das größte Gesicht auf dem Quellbild (also das Referenzgesicht) mit bis zu 100 auf dem Zielbild erkannten

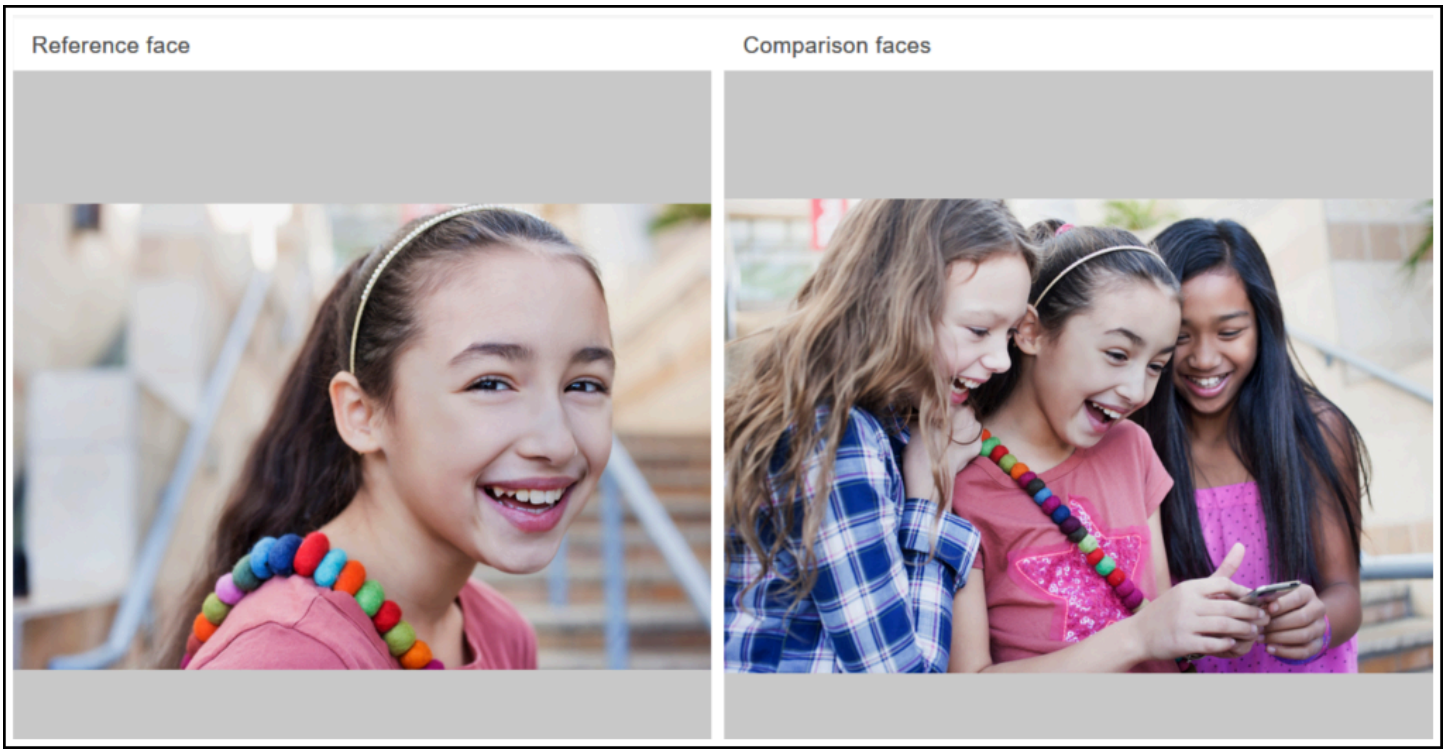
Gesichtern (also der Gesichtervergleich) und gibt an, inwieweit das Gesicht auf dem Quellbild mit den Gesichtern auf dem Zielbild übereinstimmt. Der Ähnlichkeitswert für jeden Vergleich wird im Bereich Results angezeigt.

Falls das Zielbild mehrere Gesichter enthält, vergleicht Rekognition das Gesicht auf dem Quellbild mit bis zu 100 auf dem Zielbild erkannten Gesichtern und weist anschließend jeder Übereinstimmung einen Ähnlichkeitswert zu.

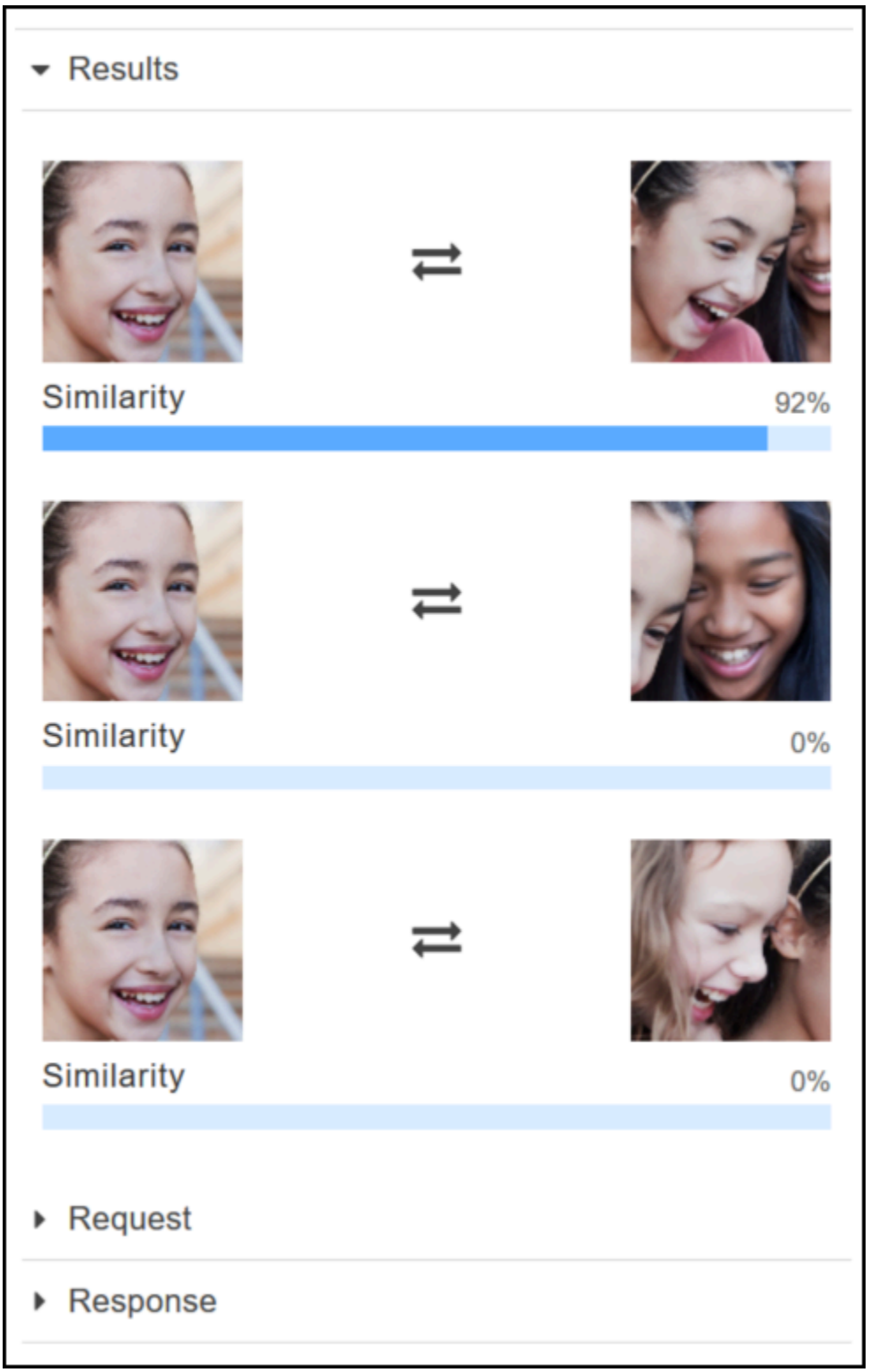
Falls das Quellbild mehrere Gesichter enthält, erkennt der Dienst das größte Gesicht auf dem Quellbild und vergleicht es mit jedem auf dem Zielbild erkannten Gesicht.

Weitere Informationen finden Sie unter Vergleichen von Gesichtern in Bildern.

Ein Beispiel: Mit dem links gezeigten Beispielbild als Quellbild und dem rechts gezeigten Beispielbild als Zielbild erkennt Rekognition das Gesicht im Quellbild, vergleicht es mit jedem im Zielbild erkannten Gesicht und zeigt für jedes Paar einen Ähnlichkeitswert an.



Die folgende Abbildung zeigt die auf dem Zielbild erkannten Gesichter und den Ähnlichkeitswert für jedes Gesicht.



Vergleich von Gesichtern auf einem von Ihnen bereitgestellten Bild

Sie können Ihre eigenen Quell- und Zielbilder für den Amazon-Rekognition-Gesichtsvergleich hochladen oder eine URL für den Standort der Bilder angeben.

> **Note**
>
> Das Bild darf höchstens 5MB groß sein und muss im JPEG- oder PNG-Format vorliegen.

Vergleich von Gesichtern auf Ihren Bildern

1. Öffnen Sie die Amazon Rekognition Rekognition-Konsole unter. https://console.aws.amazon.com/rekognition/
2. Wählen Sie Gesichtsvergleich.
3. Führen Sie für Ihr Quellbild einen der folgenden Schritte aus:
   - Laden Sie ein Bild hoch – Wählen Sie links Hochladen aus, navigieren Sie zum Speicherort Ihres Quellenbildes und wählen Sie das entsprechende Bild aus.
   - Verwenden Sie eine URL – Geben Sie die URL Ihres Quellenbilds in das Textfeld ein und wählen Sie anschließend Los.
4. Führen Sie für Ihr Zielbild einen der folgenden Schritte aus:
   - Laden Sie ein Bild hoch – Wählen Sie auf der rechten Seite Hochladen aus, navigieren Sie zum Speicherort Ihres Quellenbildes und wählen Sie das entsprechende Bild aus.
   - Verwenden Sie eine URL – Geben Sie die URL Ihres Quellenbilds in das Textfeld ein und wählen Sie anschließend Los.
5. Rekognition vergleicht das größte Gesicht auf Ihrem Quellbild mit bis zu 100 Gesichtern auf dem Zielbild und zeigt anschließend den Ähnlichkeitswert für jedes Paar im Bereich Ergebnisse an.

Übung 4: Anzeigen von Gesamtmetriken (Konsole)

Im Amazon-Rekognition-Metrikbereich werden Aktivitätsgraphen für eine Gesamtheit von individuellen Rekognition-Metriken aus einem bestimmten Zeitraum angezeigt. Die `SuccessfulRequestCount`-Gesamtmetrik zeigt beispielsweise die Gesamtanzahl der erfolgreichen Anforderungen an alle Rekognition-API-Operationen der letzten sieben Tage an.

In der folgenden Tabelle finden Sie eine Liste der im Rekognition-Metrikbereich angezeigten Graphen und der dazugehörigen Rekognition-Metrik. Weitere Informationen finden Sie unter CloudWatchMetriken für Rekognition.

Diagramm	Gesamtmetrik
Erfolgreiche Aufrufe	SuccessfulRequestCount
Client-Fehler	UserErrorCount
Serverfehler	ServerErrorCount
Gedrosselt	ThrottledCount
Erkannte Labels	DetectedLabelCount
Erkannte Gesichter	DetectedFaceCount

Jeder Graph zeigt die Gesamtmetrik-Daten an, die über einen bestimmten Zeitraum hinweg gesammelt wurden. Außerdem wird die Gesamtanzahl der Gesamtmetrik-Daten für den Zeitraum angezeigt. Um Metriken für einzelne API-Aufrufe anzuzeigen, wählen Sie den Link unter dem jeweiligen Graphen aus.

Um Benutzern Zugriff auf den Bereich Rekognition-Metriken zu gewähren, stellen Sie sicher, dass der Benutzer über die entsprechenden CloudWatch Rekognition-Berechtigungen verfügt. Es kann zum Beispiel ein Benutzer mit den verwalteten Richtlinien-Berechtigungen `AmazonRekognitionReadOnlyAccess` und `CloudWatchReadOnlyAccess` den Metrikbereich sehen. Wenn ein Benutzer über keine erforderliche Berechtigung verfügt, werden keine Graphen angezeigt, wenn der Benutzer den Metrikbereich öffnet. Weitere Informationen finden Sie unter Identitäts- und Zugriffsverwaltung für Amazon Rekognition.

Weitere Informationen zur Überwachung von Rekognition finden Sie unter CloudWatch . Überwachung von Rekognition mit Amazon CloudWatch

Anzeige von Gesamtmetriken (Konsole)

1. Öffnen Sie die Amazon Rekognition Rekognition-Konsole unter. https://console.aws.amazon.com/rekognition/
2. Wählen Sie im Navigationsbereich Metriken aus.
3. Wählen Sie im Drop-down-Menü den Zeitraum aus, für den Sie Metriken anzeigen möchten.
4. Klicken Sie auf die Schaltfläche Refresh, um die Graphen zu aktualisieren.

5. Um detaillierte CloudWatch Metriken für eine bestimmte aggregierte Metrik anzuzeigen, wählen Sie CloudWatch unter dem Metrikdiagramm die Option Details anzeigen aus.

Mit Bildern und Videos arbeiten

Sie können Amazon Rekognition API-Operationen mit drei verschiedenen Medientypen verwenden: Bilder, gespeicherte Videos und Streaming-Videos. Dieser Abschnitt enthält allgemeine Informationen zum Schreiben von Code, der auf Amazon Rekognition zugreift, um die verschiedenen Medientypen zu verarbeiten. Anleitungen zu bewährten Methoden und Überlegungen finden Sie in den entsprechenden Abschnitten, die je nach Art der Medien, die Sie verarbeiten, unten aufgeführt sind.

Andere Abschnitte in diesem Handbuch enthalten Informationen über bestimmte Arten der Bild- und Videoanalyse, z. B. der Gesichtserkennung.

Themen

- Arbeiten mit Bildern
- Arbeiten mit gespeicherten Videoanalysevorgängen
- Arbeiten mit Streaming-Videoereignissen
- Fehlerbehandlung
- Verwendung von Amazon Rekognition als FedRAMP-autorisierter Service

Arbeiten mit Bildern

In diesem Abschnitt werden die Arten von Analysen beschrieben, die Amazon Rekognition Image für Bilder durchführen kann.

- Objekt- und Szenenerkennung
- Gesichtserkennung und -vergleich
- Gesichtssuche in einer Sammlung
- Prominentenerkennung
- Überwachen von Bildern
- Texterkennung in Bildern

Diese werden durch nicht-speicherbare API-Vorgänge ausgeführt, bei denen Amazon Rekognition Image die durch die Operation ermittelten Informationen nicht speichert. Es werden keine Eingabebild-Bytes durch die nicht speicherbasierten API-Operationen persistent gespeichert. Weitere Informationen finden Sie unter Grundlegendes zu Nicht-Speicher- und Speicher-API-Vorgängen.

Amazon Rekognition Image kann auch Metadaten zu Gesichtern in Sammlungen für den späteren Abruf speichern. Weitere Informationen finden Sie unter Gesichtssuche in einer Sammlung.

In diesem Abschnitt verwenden Sie die Operationen der Amazon Rekognition Image API, um Bilder zu analysieren, die in einem Amazon-S3-Bucket gespeichert sind, sowie Bildbytes, die aus dem lokalen Dateisystem geladen wurden. Dieser Abschnitt behandelt auch das Abrufen von Bildausrichtungsinformationen aus einem JPG-Bild.

Rekognition verwendet nur RGB-Kanäle, um Inferenzen durchzuführen. AWS empfiehlt Benutzern, den Alphakanal zu entfernen, bevor sie ein Display verwenden, um den Vergleich visuell (manuell durch einen Menschen) zu überprüfen.

Themen

- Bildspezifikationen
- Analysieren von Bildern, die in einem Amazon-S3-Bucket gespeichert sind
- Analysieren eines aus einem lokalen Dateisystem geladenen Bildes
- Anzeigen von Begrenzungsrahmen
- Erhalten der Bildausrichtung und der Koordinaten von Begrenzungsrahmen

Bildspezifikationen

Amazon-Rekognition-Image-Operationen können Bilder im .jpg- oder .png-Format analysieren.

Sie übergeben Bild-Bytes an eine Amazon-Rekognition-Image-Operation als Teil des Aufrufs oder Sie verweisen auf ein vorhandenes Amazon-S3-Objekt. Ein Beispiel für die Analyse eines Bildes, das in einem Amazon-S3-Bucket gespeichert ist, finden Sie unter Analysieren von Bildern, die in einem Amazon-S3-Bucket gespeichert sind. Ein Beispiel für die Übergabe von Bild-Bytes an eine Amazon-Rekognition-Image-API-Operation finden Sie unter Analysieren eines aus einem lokalen Dateisystem geladenen Bildes.

Wenn Sie HTTP verwenden und die Bild-Bytes als Teil einer Amazon-Rekognition-Image-Operation übertragen, müssen die Bild-Bytes als base64-kodierte Zeichenfolge vorliegen. Wenn Sie das AWS-SDK verwenden und Bild-Bytes als Teil des API-Operation-Aufrufs übertragen, hängt es von Ihrer verwendeten Sprache ab, ob Sie die Bild-Bytes base64 kodieren müssen.

In den folgenden Fällen AWS SDKs werden Bilder automatisch mit Base64 codiert, und Sie müssen keine Bildbytes codieren, bevor Sie eine Amazon Rekognition Image API-Operation aufrufen.

- Java
- JavaScript
- Python
- PHP

Wenn Sie ein anderes AWS-SDK verwenden und beim Aufruf einer Rekognition-API-Operation einen Bildformatfehler erhalten, versuchen Sie die Codierung in base64 Bild-Bytes, bevor Sie sie an eine Rekognition-API-Operation übergeben.

Wenn Sie Amazon Rekognition Image-Operationen aufrufen, wird die Übergabe von Bildbytes als Teil des Aufrufs nicht unterstützt. AWS CLI Sie müssen das Bild zuerst auf einen Amazon S3-Bucket hochladen und anschließend die Operation mit Verweis auf das hochgeladene Bild aufrufen.

> **Note**
>
> Das Bild muss nicht base64-codiert werden, wenn Sie ein Bild übergeben, das in einem `S3Object` anstatt in Bild-Bytes gespeichert ist.

Informationen zur Sicherstellung der geringstmöglichen Latenz für Amazon-Rekognition-Image-Operationen finden Sie unter Latenz der Amazon-Rekognition-Image-Operation.

Korrigieren der Bildausrichtung

In mehreren Amazon-Rekognition-API-Operationen wird die Ausrichtung eines analysierten Bildes zurückgegeben. Es ist wichtig, die Bildausrichtung zu kennen, da dies Ihnen ermöglicht, Bilder für die Anzeige neu auszurichten. Rekognition-API-Operationen, die Gesichter analysieren, geben auch Begrenzungsrahmen für die Position von Gesichtern innerhalb eines Bilds zurück. Sie können die Begrenzungsrahmen verwenden, um einen Rahmen um ein Gesicht herum auf einem Bild anzuzeigen. Die zurückgegebenen Koordinaten der Begrenzungsrahmen werden von der Bildausrichtung beeinflusst, und Sie müssen die Koordinaten der Begrenzungsrahmen eventuell übertragen, damit der Rahmen um ein Gesicht korrekt angezeigt wird. Weitere Informationen finden Sie unter Erhalten der Bildausrichtung und der Koordinaten von Begrenzungsrahmen.

Größe eines Bilds anpassen

Während der Analyse passt Amazon Rekognition die Größe von Bildern intern anhand einer Reihe vordefinierter Bereiche an, die am besten zu einem bestimmten Modell oder Algorithmus passen. Aus

diesem Grund kann Amazon Rekognition je nach Auflösung des Eingabebilds eine unterschiedliche Anzahl von Objekten erkennen oder unterschiedliche Ergebnisse liefern. Nehmen wir an, Sie haben zwei Bilder. Das erste Bild hat eine Auflösung von 1024x768 Pixeln. Das zweite Bild, eine verkleinerte Version des ersten Bildes, hat eine Auflösung von 640x480 Pixeln. Wenn Sie die Bilder an senden DetectLabels, `DetectLabels` können sich die Antworten der beiden Aufrufe an geringfügig unterscheiden.

Analysieren von Bildern, die in einem Amazon-S3-Bucket gespeichert sind

Amazon Rekognition Image kann Bilder analysieren, die in einem Amazon-S3-Bucket gespeichert sind, oder Bilder, die als Bild-Bytes bereitgestellt werden.

In diesem Thema verwenden Sie den DetectLabelsAPI-Vorgang, um Objekte, Konzepte und Szenen in einem Bild (JPEG oder PNG) zu erkennen, das in einem Amazon S3 S3-Bucket gespeichert ist. Sie übergeben ein Bild an eine Amazon-Rekognition-Image-API-Operation, indem Sie den Eingabeparameter Bild verwenden. Innerhalb von `Image`spezifizieren Sie die S3Object-Objekteigenschaft, um auf ein in einem S3-Bucket gespeichertes Bild zu verweisen. Bild-Bytes für Bilder, die in Amazon-S3-Buckets gespeichert sind, müssen nicht base64-codiert werden. Weitere Informationen finden Sie unter Bildspezifikationen.

Beispielanforderung

In diesem Beispiel einer JSON-Anforderung für `DetectLabels` wird das Quell-Bild (`input.jpg`) aus einem Amazon-S3-Bucket mit dem Namen `amzn-s3-demo-bucket` geladen. Beachten Sie, dass die Region für den S3-Bucket, der das S3-Objekt enthält, mit der Region übereinstimmen muss, die Sie für Amazon-Rekognition-Image-Operationen verwenden.

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

In den folgenden Beispielen werden die AWS SDKs Aufrufe various und AWS CLI to verwendet`DetectLabels`. Informationen über die Antwort auf die Operation `DetectLabels` finden Sie unter DetectLabels Antwort.

So erkennen Sie Labels in einem Bild

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`- und `AmazonS3ReadOnlyAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs. Stellen Sie sicher, dass Sie dem Benutzer, der die API-Operationen aufruft, die richtigen Berechtigungen für den programmatischen Zugriff erteilt haben. Eine Anleitung dazu finden Sie unter Erteilen programmgesteuerten Zugriffs.

2. Laden Sie ein Bild mit einem oder mehreren Objekten wie Bäumen, Häusern und einem Boot auf Ihren S3-Bucket hoch. Das Bild muss entweder im JPG- oder PNG-Format vorliegen.

   Weitere Anleitungen finden Sie unter Upload eines Objekts in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service.

3. Verwenden Sie die folgenden Beispiele zum Aufrufen der `DetectLabels`-Operation.

   Java

   Dieses Beispiel zeigt eine Liste von Labels an, die auf dem Eingabebild erkannt wurden. Ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, das Sie in Schritt 2 verwendet haben.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Dieses Beispiel zeigt die JSON-Ausgabe von der `detect-labels`-CLI-Operation an. Ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, das Sie in Schritt 2 verwendet haben. Ersetzen Sie den Wert von

`profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
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
```

Wenn Sie Windows verwenden, müssen Sie möglicherweise die Escaped Anführungszeichen verwenden, wie im Beispiel unten gezeigt.

```
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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
//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

Dieses Beispiel zeigt die Labels an, die auf dem Eingabebild erkannt wurden. Ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, das Sie in Schritt 2 verwendet haben. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

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

import boto3

def detect_labels(photo, bucket):
```

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

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

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

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

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

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

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

    if "ImageProperties" in str(response):
        print("Background:")
```

```
        print(response["ImageProperties"]["Background"])
        print()
        print("Foreground:")
        print(response["ImageProperties"]["Foreground"])
        print()
        print("Quality:")
        print(response["ImageProperties"]["Quality"])
        print()

    return len(response['Labels'])

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

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

Node.Js

In diesem Beispiel werden Informationen über in einem Bild erkannte Labels angezeigt.

Ändern Sie den Wert von `photo` in den Pfad und Dateinamen einer Bilddatei mit einem oder mehreren prominenten Gesichtern. Ändern Sie den Wert von `bucket` in den Namen des S3-Buckets, der die angegebene Bilddatei enthält. Ändern Sie den Wert von `REGION` in den Namen der Region, die Ihrem Konto zugeordnet ist. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
// 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

Dieses Beispiel zeigt eine Liste von Labels an, die auf dem Eingabebild erkannt wurden. Ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, das Sie in Schritt 2 verwendet haben.

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

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

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

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

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

        try
        {
```

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

Ruby

Dieses Beispiel zeigt eine Liste von Labels an, die auf dem Eingabebild erkannt wurden. Ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, das Sie in Schritt 2 verwendet haben.

```
  # 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
```

Beispielantwort

Die Antwort von `DetectLabels` ist ein Array der erkannten Labels in dem Bild und der Zuverlässigkeitswert, mit dem sie erkannt wurden.

Wenn Sie die `DetectLabels`-Operation mit einem Bild ausführen, gibt Amazon Rekognition eine Ausgabe zurück, die der folgenden Beispielantwort ähnelt.

Die Antwort zeigt, dass die Operation mehrere Labels erkannt hat, einschließlich Person, Fahrzeug und Auto. Jedem Label ist ein Zuverlässigkeitswert zugeordnet. Beispielsweise ist der Erkennungsalgorithmus zu 98,991432 % sicher, dass auf dem Bild eine Person abgebildet ist.

Die Antwort umfasst auch die übergeordneten Vorgängerlabels für ein Label im `Parents`-Array. Beispielsweise hat das Label „Automobil“ die zwei übergeordneten Labels „Fahrzeug“ und „Transport“.

Die Antwort für gängige Objektlabels enthält Informationen zum Begrenzungsrahmen für die Position des Labels auf dem Eingabebild. Beispiel: Das Label „Person“ ist ein Instances-Array mit zwei Begrenzungsrahmen. Dies sind die Positionen von zwei Personen, die im Bild erkannt wurden.

Das Feld `LabelModelVersion` enthält die Versionsnummer des von `DetectLabels` verwendeten Erkennungsmodells.

Weitere Informationen zur Verwendung der `DetectLabels`-Operation finden Sie unter Erkennung von Objekten und Konzepten.


```
{

    {
    "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"
}


}
```

Analysieren eines aus einem lokalen Dateisystem geladenen Bildes

Amazon-Rekognition-Image-Operationen können Bilder analysieren, die als Bild-Bytes oder in einem Amazon-S3-Bucket gespeichert sind.

In diesen Themen finden Sie Beispiele, wie Sie Bild-Bytes an Amazon-Rekognition-Image-API-Operationen übertragen können, indem Sie eine Datei verwenden, die aus einem lokalen Dateisystem geladen wird. Sie übergeben Bild-Bytes an eine Amazon-Rekognition-API-Operation, indem Sie den Eingabeparameter Bild verwenden. Innerhalb von `Image` geben Sie die Eigenschaft `Bytes` zum Übergeben von base64-codierten Bild-Bytes an.

Bild-Bytes, die an eine Amazon-Rekognition-API-Operation mit dem Eingabeparameter `Bytes` übergeben werden, müssen base64-codiert sein. Die AWS SDKs , dass diese Beispiele Bilder automatisch mit Base64 kodieren. Sie müssen vor dem Aufruf einer Amazon-Rekognition-API-Operation keine Bild-Bytes codieren. Weitere Informationen finden Sie unter Bildspezifikationen.

In diesem Beispiel einer JSON-Anforderung für `DetectLabels` werden die Quellbild-Bytes im `Bytes`-Eingabeparameter übergeben.

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

In den folgenden Beispielen werden various AWS SDKs und to call verwendet. AWS CLI `DetectLabels` Informationen über die Antwort auf die Operation `DetectLabels` finden Sie unter DetectLabels Antwort.

Ein clientseitiges JavaScript Beispiel finden Sie unter. Verwenden JavaScript

So erkennen Sie Labels in einem lokalen Bild

1. Wenn Sie dies noch nicht getan haben:

    a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`- und `AmazonS3ReadOnlyAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

b. Installieren und konfigurieren Sie das AWS CLI und das. AWS SDKs Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `DetectLabels`-Operation.

Java

Das folgende Java-Beispiel zeigt, wie Sie ein Bild aus dem lokalen Dateisystem laden und Labels mithilfe der AWS-SDK-Operation detectLabels erkennen. Ändern Sie den Wert von `photo` in den Pfad und Dateinamen einer Bilddatei (JPG- oder 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.)

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

Das folgende AWS-SDK für Python-Beispiel zeigt, wie Sie ein Bild aus dem lokalen Dateisystem laden und die Operation detect_labels aufrufen. Ändern Sie den Wert von `photo` in den Pfad und Dateinamen einer Bilddatei (JPG- oder 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.)

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

Das folgende Beispiel veranschaulicht, wie Sie ein Bild aus dem lokalen Dateisystem laden und Labels mithilfe der Operation `DetectLabels` erkennen. Ändern Sie den Wert von `photo` in den Pfad und Dateinamen einer Bilddatei (JPG- oder 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 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

Das folgende Beispiel für ein AWS SDK for PHP zeigt, wie ein Bild aus dem lokalen Dateisystem geladen und die DetectFacesAPI-Operation aufgerufen wird. Ändern Sie den Wert von `photo` in den Pfad und Dateinamen einer Bilddatei (JPG- oder PNG-Format).

```
<?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

Dieses Beispiel zeigt eine Liste von Labels an, die auf dem Eingabebild erkannt wurden. Ändern Sie den Wert von `photo` in den Pfad und Dateinamen einer Bilddatei (JPG- oder 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.)

    # 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

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

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

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

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

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

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

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

    /**
     * Detects the labels in an image stored in an Amazon S3 bucket using the
 Amazon Rekognition service.
     *
     * @param rekClient     the Amazon Rekognition client used to make the
 detection request
     * @param bucketName    the name of the Amazon S3 bucket where the image is
 stored
     * @param sourceImage   the name of the image file to be analyzed
     */
    public static void detectImageLabels(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
        try {
```

```
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

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

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

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

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

Verwenden JavaScript

Das folgende JavaScript Webseitenbeispiel ermöglicht es einem Benutzer, ein Bild auszuwählen und sich das geschätzte Alter der Gesichter anzusehen, die auf dem Bild erkannt wurden. Die geschätzten Altersgruppen werden über einen Aufruf von DetectFaces zurückgegeben.

Das gewählte Bild wird mithilfe der JavaScript `FileReader.readAsDataURL` Funktion geladen, die das Bild base64-kodiert. Dies ist nützlich, um das Bild auf einer HTML-Zeichenfläche anzuzeigen. Es bedeutet jedoch, dass die Bild-Bytes in ein nicht codiertes Format umgewandelt werden müssen, bevor sie einer Amazon-Rekognition-Image-Operation übergeben werden. In diesem Beispiel wird

gezeigt, wie Sie die geladenen Bildbytes entcodieren. Wenn die entcodierten Bild-Bytes nicht nützlich für Sie sind, verwenden Sie stattdessen `FileReader.readAsArrayBuffer`, da das geladenen Bild nicht kodiert ist. Das bedeutet, dass Amazon-Rekognition-Image-Operationen aufgerufen werden können, ohne dass die Bild-Bytes zuvor in ein nicht codiertes Format umgewandelt werden müssen. Ein Beispiel finden Sie unter readAsArrayBuffer verwenden.

Um das Beispiel auszuführen JavaScript

1. Laden Sie den Beispielquellcode in einen Editor.
2. Rufen Sie die Amazon-Cognito-Identitätspool-ID ab. Weitere Informationen finden Sie unter Abrufen der Amazon-Cognito-Identitätspool-ID.
3. Ändern Sie in der `AnonLog`-Funktion des Beispielcodes `IdentityPoolIdToUse` und `RegionToUse` in die Werte, die Sie in Schritt 9 von Abrufen der Amazon-Cognito-Identitätspool-ID notiert haben.
4. Ändern Sie in der Funktion `DetectFaces` `RegionToUse` in den im vorigen Schritt verwendeten Wert.
5. Speichern Sie den Beispielquellcode als `.html`-Datei.
6. Laden Sie die Datei in Ihren Browser.
7. Wählen Sie die Schaltfläche Durchsuchen... und wählen Sie ein Bild aus, das ein oder mehrere Gesichter enthält. Eine Tabelle wird mit den geschätzten Altersgruppen für jedes im Bild erkannt Gesicht wird angezeigt.

> **Note**
>
> Im folgenden Codebeispiel werden zwei Skripts verwendet, die nicht mehr Teil von Amazon Cognito sind. Um diese Dateien abzurufen, folgen Sie den Links für aws-cognito-sdk.min.js und amazon-cognito-identity.min.js und speichern Sie dann den Text aus jeder Datei als separate `.js` Dateien.

JavaScript Beispielcode

Das folgende Codebeispiel verwendet JavaScript V2. Ein Beispiel für JavaScript V3 finden Sie in dem Beispiel im AWS Documentation SDK Examples GitHub Repository.

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

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

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

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

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

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

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

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

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

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

readAsArrayBuffer verwenden

Der folgende Codeausschnitt ist eine alternative Implementierung der `ProcessImage` Funktion im Beispielcode unter Verwendung JavaScript von V2. Es verwendet `readAsArrayBuffer`, um ein Bild zu laden und `DetectFaces` aufzurufen. Da `readAsArrayBuffer` die Datei nicht mit base64 codiert, ist es nicht notwendig, die Bild-Bytes zu entcodieren, bevor eine Amazon-Rekognition-Image-Operation aufgerufen wird.

```
//Copyright 2018 Amazon.com, Inc. or its 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);
  }
```

Abrufen der Amazon-Cognito-Identitätspool-ID

Der Einfachheit halber wird in diesem Beispiel ein anonymer Amazon-Cognito-Identitätspool verwendet, um einen nicht authentifizierten Zugriff auf die Amazon-Rekognition-Image-API zu ermöglichen. Dies könnte für Ihre Anforderungen nützlich sein. Sie können den nicht authentifizierten Zugriff beispielsweise verwenden, um kostenlosen Zugriff auf Ihre Website oder einen Testzugang zu bieten, bevor sich Benutzer anmelden. Um authentifizierten Zugriff bereitzustellen, verwenden Sie einen Amazon-Cognito-Benutzerpool. Weitere Informationen finden Sie unter Amazon-Cognito-Benutzerpool.

Das folgende Verfahren zeigt, wie Sie einen Identitätspool erstellen, der den Zugriff auf nicht authentifizierte Identitäten ermöglicht, und wie Sie die Identitätspool-ID abrufen, die im Beispielcode benötigt wird.

Abrufen der Identitätspool-ID

1. Öffnen Sie die Amazon-Cognito-Konsole.
2. Wählen Sie Neuen Identitätspool erstellen.
3. Geben Sie für Identitätspoolname* einen Namen für Ihren Identitätspool ein.
4. Wählen Sie in Nicht authentifizierte Identitäten die Option Zugriff auf nicht authentifizierte Identitäten aktivieren.
5. Wählen Sie Pool erstellen.
6. Wählen Sie Details anzeigen und notieren Sie den Rollennamen für nicht authentifizierte Identitäten.
7. Wählen Sie Zulassen.
8. Wählen Sie unter Plattform die Option. JavaScript
9. Notieren Sie in AWS-Anmeldeinformationen abrufen die Werte von `AWS.config.region` und `IdentityPooldId`, die im Codeabschnitt angezeigt werden.
10. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
11. Wählen Sie im Navigationsbereich Rollen aus.
12. Wählen Sie den Rollennamen, den Sie in Schritt 6 notiert haben.
13. Wählen Sie in der Registerkarte Berechtigungen die Option Richtlinien anfügen aus.
14. Wählen Sie AmazonRekognitionReadOnlyAccess.
15. Wählen Sie Richtlinie anfügen aus.

Anzeigen von Begrenzungsrahmen

Amazon-Rekognition-Image-Operationen können Koordinaten der Begrenzungsrahmen für Elemente zurückgeben, die in Bildern erkannt werden. Beispiel: Die DetectFaces-Operation gibt für jedes Gesicht, das in einem Bild erkannt wird, einen Begrenzungsrahmen (BoundingBox) zurück. Sie können die Koordinaten des Begrenzungsrahmens verwenden, um einen Rahmen um die erkannten Elemente anzuzeigen. Beispiel: In der folgenden Abbildung wird ein Begrenzungsrahmen angezeigt, der ein Gesicht umgibt.

Eine `BoundingBox` besitzt die folgenden Eigenschaften:

- Höhe: Höhe des Begrenzungsrahmens als Verhältnis der gesamten Bildhöhe.
- Links: Linke Koordinate des Begrenzungsrahmens als Verhältnis zur Gesamtbildbreite.
- Oben: Obere Koordinate des Begrenzungsrahmens als Verhältnis zur Gesamtbildbreite.
- Breite: Breite des Begrenzungsrahmens als Verhältnis der gesamten Bildbreite.

Jede BoundingBox Eigenschaft hat einen Wert zwischen 0 und 1. Jeder Eigenschaftswert ist ein Verhältnis der gesamten Breite (`Left` und `Width`) oder der gesamten Höhe (`Height` und `Top`) des Bildes. Beispiel: Bei einem Eingabebild mit 700 x 200 Pixeln und einer linken oberen Koordinate des Begrenzungsrahmens von 350 x 50 Pixeln gibt die API einen `Left`-Wert von 0,5 (350/700) und einen `Top`-Wert von 0,25 (50/200) zurück.

Das folgende Diagramm zeigt den Bereich eines Bildes, der von jeder Begrenzungsrahmen-Eigenschaft abgedeckt wird.

Um den Begrenzungsrahmen mit der richtigen Position und Größe anzuzeigen, müssen Sie die BoundingBox Werte mit der Bildbreite oder -höhe (abhängig vom gewünschten Wert) multiplizieren, um die Pixelwerte zu erhalten. Sie verwenden die Pixelwerte, um den Begrenzungsrahmen anzuzeigen. Beispiel: Die Pixelmaße des vorherigen Bildes waren Breite 608 x Höhe 588. Die Werte des Begrenzungsrahmens für das Gesicht sind:

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

Der Speicherort des Gesichts-Begrenzungsrahmens in Pixeln wird wie folgt berechnet:

```
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
```

Sie können mit diesen Werten einen Begrenzungsrahmen um das Gesicht herum anzeigen.

> **Note**
>
> Ein Bild kann auf verschiedene Weise ausgerichtet werden. Ihre Anwendung muss das Bild möglicherweise rotieren, um es in der richtigen Ausrichtung anzuzeigen. Die Koordinaten des Begrenzungsrahmens sind von der Ausrichtung des Bildes betroffen. Sie müssen die Koordinaten möglicherweise übersetzen, bevor Sie einen Begrenzungsrahmen an der

richtigen Stelle anzeigen können. Weitere Informationen finden Sie unter Erhalten der Bildausrichtung und der Koordinaten von Begrenzungsrahmen.

Die folgenden Beispiele zeigen, wie Sie einen Begrenzungsrahmen um Gesichter herum anzeigen, die durch Aufrufen von DetectFaces erkannt werden. Die Beispiele gehen davon aus, dass die Bilder auf 0 Grad ausgerichtet sind. Die Beispiele zeigen auch, wie das Bild aus einem Amazon-S3-Bucket heruntergeladen wird.

So zeigen Sie einen Begrenzungsrahmen an

1. Wenn Sie dies noch nicht getan haben:
   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`- und `AmazonS3ReadOnlyAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.
   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.
2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `DetectFaces`-Operation.

   Java

   Ändern Sie den Wert von `bucket` in den Amazon-S3-Bucket mit der Bilddatei. Ändern Sie den Wert von `photo` in den Dateinamen einer Bilddatei (JPG- oder PNG-Format).

```
//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

Ändern Sie den Wert von `bucket` in den Amazon-S3-Bucket mit der Bilddatei. Ändern Sie den Wert von `photo` in den Dateinamen einer Bilddatei (JPG- oder PNG-Format). Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

Beachten Sie, dass `s3` sich auf den AWS-SDK-Amazon-S3-Client bezieht und `rekClient` sich auf den AWS-SDK-Amazon-Rekognition-Client bezieht.

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

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

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

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

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

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

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

        String sourceImage = args[0];
        String bucketName = args[1];
```

```
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

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

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

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

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

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

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

```
            result = rekClient.detectFaces(facesRequest);

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

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

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

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

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

        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
```

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

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

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

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

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

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

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

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

Erhalten der Bildausrichtung und der Koordinaten von Begrenzungsrahmen

Anwendungen, die Amazon Rekognition Image verwenden, müssen in der Regel die von Amazon Rekognition Image erkannten Bilder und die Rahmen um die erkannten Gesichter anzeigen. Um ein Bild in Ihrer Anwendung korrekt anzuzeigen, müssen Sie die Ausrichtung des Bildes kennen. Möglicherweise müssen Sie diese Ausrichtung anpassen. Bei einigen JPG-Dateien ist die Orientierung des Bildes in den Metadaten des Exchangeable Image File Format (Exif) des Bildes enthalten.

Um einen Rahmen um ein Gesicht herum anzuzeigen, benötigen Sie die Koordinaten für den Begrenzungsrahmen des Gesichts. Wenn das Feld nicht richtig ausgerichtet ist, müssen Sie diese Koordinaten möglicherweise anpassen. Die Gesichtserkennungsoperationen von Amazon Rekognition Image geben die Bounding-Box-Koordinaten für jedes erkannte Gesicht zurück, aber die Koordinaten für JPG-Dateien ohne Exif-Metadaten werden nicht geschätzt.

Die folgenden Beispiele zeigen, wie Sie die Bounding-Box-Koordinaten für die in einem Bild erkannten Gesichter erhalten.

Verwenden Sie die Informationen in diesem Beispiel, um sicherzustellen, dass Ihre Bilder richtig ausgerichtet sind, und dass die Begrenzungsrahmen an der richtigen Stelle in Ihrer Anwendung angezeigt werden.

Da der Code, der zum Drehen und Anzeigen von Bildern und Begrenzungsrahmen verwendet wird, von der Sprache und Umgebung abhängt, die Sie verwenden, erklären wir nicht, wie Sie Bilder und Begrenzungsrahmen in Ihrem Code darstellen oder wie Sie Orientierungsinformationen aus den Exif-Metadaten erhalten.

Ermitteln der Ausrichtung eines Bildes

Um ein Bild in Ihrer Anwendung korrekt darzustellen, müssen Sie es möglicherweise drehen. Das folgende Bild ist auf 0 Grad ausgerichtet und wird korrekt dargestellt.

Das folgende Bild ist jedoch um 90 Grad gegen den Uhrzeigersinn gedreht. Um es korrekt darzustellen, müssen Sie die Ausrichtung des Bildes ermitteln und diese Information in Ihrem Code verwenden, um das Bild auf 0 Grad zu drehen.

Einige Bilder im JPG-Format enthalten Orientierungsinformationen in den Exif-Metadaten. Falls verfügbar, enthalten die Exif-Metadaten für das Bild die Ausrichtung. In den Exif-Metadaten finden Sie die Ausrichtung des Bildes im Feld `orientation`. Obwohl Amazon Rekognition Image das Vorhandensein von Bildausrichtungsinformationen in Exif-Metadaten identifiziert, bietet es keinen Zugriff darauf. Um auf die Exif-Metadaten in einem Bild zuzugreifen, verwenden Sie eine Bibliothek eines Drittanbieters oder schreiben Sie Ihren eigenen Code. Weitere Informationen finden Sie unter Exif Version 2.32.

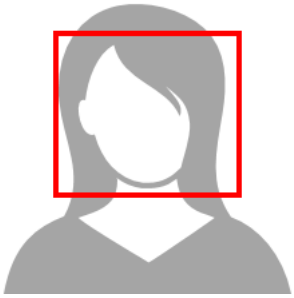
Wenn Sie die Ausrichtung eines Bildes kennen, können Sie Code schreiben, um es zu drehen und korrekt darzustellen.

Anzeigen von Begrenzungsrahmen

Die Amazon-Rekognition-Image-Operationen, die Gesichter in einem Bild analysieren, geben auch die Koordinaten der Begrenzungsrahmen zurück, die die Gesichter umgeben. Weitere Informationen finden Sie unter BoundingBox.

Um in Ihrer Anwendung einen Begrenzungsrahmen um ein Gesicht herum anzuzeigen, das dem im folgenden Bild ähnlich ist, verwenden Sie die Begrenzungsrahmenkoordinaten in Ihrem Code. Die

Koordinaten des Begrenzungsrahmens, die von einer Operation zurückgegeben werden, spiegeln die Ausrichtung des Bildes wider. Wenn Sie das Bild drehen müssen, um es korrekt darzustellen, müssen Sie die Koordinaten des Begrenzungsrahmens entsprechend übertragen.



Anzeigen von Begrenzungsrahmen, wenn die Orientierungsinformationen in Exif-Metadaten vorhanden sind

Wenn die Ausrichtung eines Bildes in den Exif-Metadaten enthalten ist, machen die Amazon-Rekognition-Image-Operationen folgendes:

- Geben Null im Feld für die Orientierungskorrektur in der Antwort der Operation zurück. Um das Bild zu drehen, verwenden Sie die Orientierung, die in den Exif-Metadaten in Ihrem Code angegeben ist.
- Geben Koordinaten für Begrenzungsrahmen bereits auf 0 Grad orientiert zurück. Um den Begrenzungsrahmen an der richtigen Position anzuzeigen, verwenden Sie die zurückgegebenen Koordinaten. Sie müssen sie nicht übertragen.

Beispiel: Erhalten der Bildausrichtung und der Koordinaten von Begrenzungsrahmen für ein Bild

Die folgenden Beispiele zeigen, wie Sie das AWS-SDK verwenden, um die Exif-Bildausrichtungsdaten und die Bounding-Box-Koordinaten für die von der `RecognizeCelebrities`-Operation erkannten Berühmtheiten zu erhalten.

> **Note**
>
> Die Unterstützung für die Schätzung der Bildausrichtung anhand des `OrientationCorrection`-Feldes wurde im August 2021 eingestellt. Alle zurückgegebenen Werte für dieses Feld, die in einer API-Antwort enthalten sind, sind immer NULL.

Java

Dieses Beispiel lädt ein Bild aus dem lokalen Dateisystem, ruft die `RecognizeCelebrities`-Operation auf, bestimmt die Höhe und Breite des Bildes und berechnet die Koordinaten des Begrenzungsrahmens des Gesichts für das gedrehte Bild. Das Beispiel zeigt nicht, wie Orientierungsinformationen, die in Exif-Metadaten gespeichert sind, verarbeitet werden können.

Ersetzen Sie in der `main`-Funktion den Wert von `photo` durch den Namen und Pfad eines Bildes, das lokal im PNG- oder JPG-Format gespeichert ist.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Dieses Beispiel verwendet die PIL/Pillow-Bildbibliothek, um Bildbreite und -höhe zu erhalten. Weitere Informationen finden Sie unter Pillow. Dieses Beispiel bewahrt die Exif-Metadaten auf, die Sie möglicherweise an anderer Stelle in Ihrer Anwendung benötigen.

Ersetzen Sie in der `main`-Funktion den Wert von `photo` durch den Namen und Pfad eines Bildes, das lokal im PNG- oder JPG-Format gespeichert ist.

```
#Copyright 2018 Amazon.com, Inc. or its 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

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

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

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

                Usage:    <sourceImage>

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

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

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

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

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

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

            Image souImage = Image.builder()
```

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

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

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

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

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

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

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

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

Arbeiten mit gespeicherten Videoanalysevorgängen

Amazon Rekognition Video ist eine API, mit der Sie Videos analysieren können. Mit Amazon Rekognition Video können Sie Labels, Gesichter, Personen, Prominente und nicht jugendfreie (anzügliche und explizite) Inhalte in Videos erkennen, die in einem Bucket von Amazon Simple Storage Service (Amazon S3) gespeichert sind. Sie können Amazon Rekognition Video in Kategorien wie Medien/Unterhaltung und öffentliche Sicherheit verwenden. Früher hätte das Durchsuchen von Videos nach Objekten oder Personen viele Stunden einer fehleranfälligen Betrachtung durch den Menschen in Anspruch genommen. Amazon Rekognition Video automatisiert die Erkennung von Objekten und deren Auftreten in einem Video.

Dieser Abschnitt behandelt die Arten von Analysen, die Amazon Rekognition Video durchführen kann, einen Überblick über die API und Beispiele für die Verwendung von Amazon Rekognition Video.

Themen

- Analysearten
- Überblick für die Amazon-Rekognition-Video-API
- Amazon-Rekognition-Video-Operationen aufrufen
- Amazon Rekognition Video konfigurieren
- Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK)
- Analysieren eines Videos mit dem AWS Command Line Interface
- Referenz: Videoanalyse-Ergebnisbenachrichtigung
- Fehlerbehebung für Amazon Rekognition Video

Analysearten

Sie können Amazon Rekognition Video verwenden, um Videos auf die folgenden Informationen zu analysieren:

- Video-Segmente
- Labels
- Suggestive und explizite Inhalte für Erwachsene
- Text
- Prominente
- Gesichter
- Personen

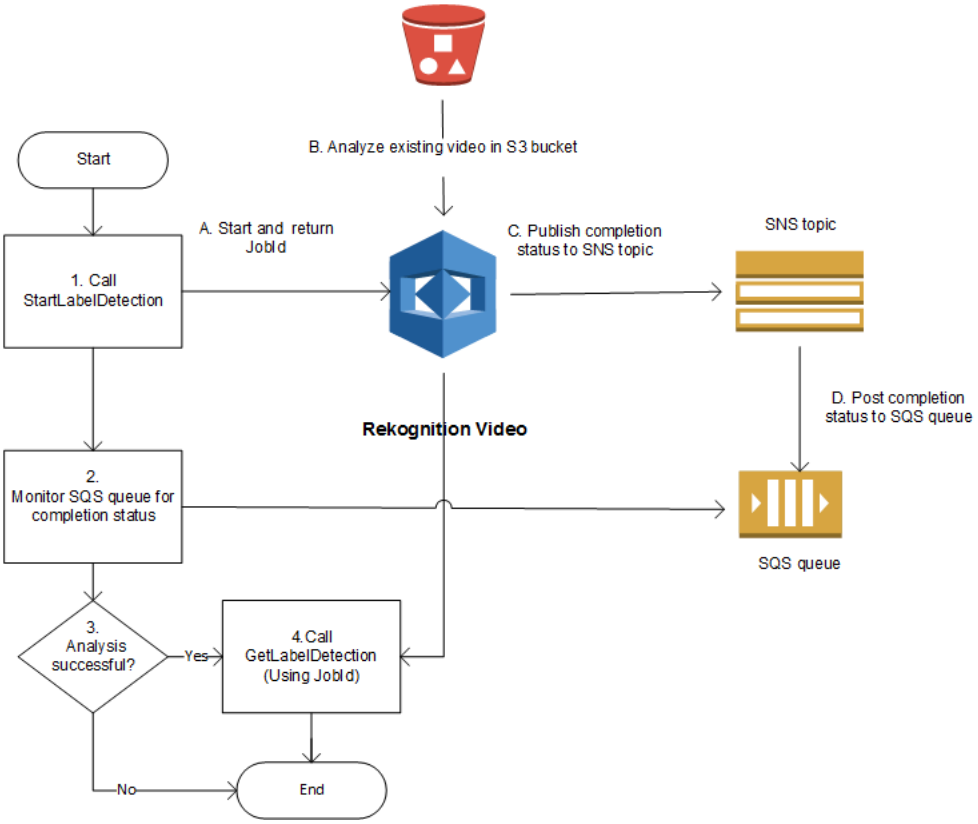
Weitere Informationen finden Sie unter So funktioniert Amazon Rekognition.

Überblick für die Amazon-Rekognition-Video-API

Amazon Rekognition Video verarbeitet ein Video, das in einem Amazon-S3-Bucket gespeichert ist. Das Design-Pattern ist ein asynchrones Set von Operationen. Sie starten die Videoanalyse durch Aufruf einer `Start`-Operation wie StartLabelDetection. Der Abschlussstatus der Anforderung

wird in einem Amazon-Simple-Notification-Service-Thema (Amazon SNS) veröffentlicht. Um den Abschlussstatus des Amazon SNS SNS-Themas abzurufen, können Sie eine Amazon Simple Queue Service (Amazon SQS) -Warteschlange oder eine AWS Lambda Funktion verwenden. Sobald Sie den Erledigungsstatus bekommen haben, rufen Sie eine `Get`-Operation auf, wie z. B. GetLabelDetection, um die Ergebnisse der Anfrage zu erhalten.

Das folgende Diagramm zeigt den Prozess zur Erkennung von Label in einem Video, das in einem Amazon-S3-Bucket gespeichert ist. Im Diagramm erhält eine Amazon-SQS-Warteschlange den Abschlussstatus aus dem Amazon-SNS-Thema. Alternativ können Sie eine AWS Lambda Funktion verwenden.



Der Prozess ist für andere Amazon-Rekognition-Video-Operationen identisch. Die folgende Tabelle listet die `Start`- und `Get`-Operationen für jede der nicht speichernden Amazon-Rekognition-Operationen auf.

Erkennung	Start-Operation	Get-Operation
Video-Segmente	StartSegmentDetection	GetSegmentDetection
Labels	StartLabelDetection	GetLabelDetection

Erkennung	Start-Operation	Get-Operation
Explizite oder suggestive Inhalte für Erwachsene	StartContentModeration	GetContentModeration
Text	StartTextDetection	GetTextDetection
Prominente	StartCelebrityRecognition	GetCelebrityRecognition
Gesichter	StartFaceDetection	GetFaceDetection
Personen	StartPersonTracking	GetPersonTracking

Für andere `Get`-Operationen als `GetCelebrityRecognition`, gibt Amazon Rekognition Video Trackinginformationen zurück, wenn Entitäten im Verlauf eines Eingabevideos erkannt werden.

Weitere Informationen zur Verwendung von Amazon Rekognition Video finden Sie unter Amazon-Rekognition-Video-Operationen aufrufen. Ein Beispiel für die Videoanalyse mit Hilfe von Amazon SQS finden Sie unter Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK). Beispiele finden Sie unter. AWS CLI Analysieren eines Videos mit dem AWS Command Line Interface

Video-Formate und -Speicher

Amazon-Rekognition-Operationen können Videos analysieren, die in Amazon-S3-Buckets gespeichert sind. Eine Liste aller Beschränkungen für Videoanalysenoperationen finden Sie unter Richtlinien und Kontingente.

Das Video muss mit dem H.264-Codec kodiert sein. Die unterstützten Dateiformate sind MPEG-4 und MOV.

Ein Codec ist eine Software oder Hardware, die Daten für eine schnellere Bereitstellung komprimiert und empfangene Daten in ihre ursprüngliche Form dekomprimiert. Der H.264-Codec wird häufig zum Aufnehmen, Komprimieren und Verteilen von Videoinhalten verwendet. Ein Videodateiformat kann einen oder mehrere Codecs enthalten. Wenn Ihre MOV- oder MPEG-4-Videodatei nicht mit Amazon Rekognition Video funktioniert, überprüfen Sie, ob der Codec, der für die Codierung des Videos verwendet wird, H.264 ist.

Jede Amazon-Rekognition-Video-API, die Audiodaten analysiert, unterstützt nur AAC-Audiocodecs.

Die maximale Dateigröße für ein gespeichertes Video beträgt 10 GB.

Suchen nach Personen

Sie können in einer Sammlung gespeicherte Gesichtsmetadaten verwenden, um ein Video nach Personen zu durchsuchen. Beispielsweise können Sie ein archiviertes Video nach einer bestimmten Person oder nach mehreren Personen durchsuchen. Sie speichern Gesichtsmetadaten aus Quellbildern in einer Sammlung, indem Sie die Operation IndexFaces verwenden. Sie können dann StartFaceSearch verwenden, um asynchron nach Gesichtern in der Sammlung zu suchen. Sie verwenden GetFaceSearch, um die Suchergebnisse zu erhalten. Weitere Informationen finden Sie unter Suche nach Gesichtern in gespeicherten Videos. Die Suche nach Personen ist ein Beispiel für eine speicherbasierte Amazon-Rekognition-Operation. Weitere Informationen finden Sie unter Speicherbasierte API-Operationen.

Sie können auch in einem Streaming-Video nach Personen suchen. Weitere Informationen finden Sie unter Arbeiten mit Streaming-Videoereignissen.

Amazon-Rekognition-Video-Operationen aufrufen

Amazon Rekognition Video ist eine asynchrone API, die Sie verwenden können, um Videos zu analysieren, die in einem Amazon-Simple-Storage-Service-Bucket (Amazon S3) gespeichert sind. Sie starten die Analyse eines Videos, indem Sie einen Amazon Rekognition Video `Start` Video-Vorgang aufrufen, z. B. StartPersonTracking Amazon Rekognition Video veröffentlicht das Ergebnis der Analyseanforderung in einem Amazon-Simple-Notification-Service-Thema (Amazon SNS). Sie können eine Amazon Simple Queue Service (Amazon SQS) -Warteschlange oder eine AWS Lambda Funktion verwenden, um den Abschlussstatus der Videoanalyseanfrage aus dem Amazon SNS SNS-Thema abzurufen. Schließlich erhalten Sie die Ergebnisse der Videoanalyseanfrage, indem Sie einen Amazon Rekognition `Get` Rekognition-Vorgang aufrufen, z. B. GetPersonTracking

Die Informationen in den folgenden Abschnitten zeigen anhand von Operationen zur Erkennung von Labels, wie Amazon-Rekognition-Video-Labels (Objekte, Ereignisse, Konzepte und Aktivitäten) in einem Video erkennt, das in einem Amazon-S3-Bucket gespeichert ist. Derselbe Ansatz funktioniert auch für die anderen Amazon Rekognition Video Video-Operationen, z. B. und. StartFaceDetectionStartPersonTracking Das Beispiel Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) zeigt das Analysieren eines Videos unter Verwendung einer Amazon-SQS-Warteschlange, um den Abschlussstatus aus dem Amazon-SNS-Thema abzurufen. Es wird auch als Grundlage für andere Amazon-Rekognition-Video-Beispiele verwendet, wie z. B. Pfade von Personen. Beispiele finden Sie unter. AWS CLI Analysieren eines Videos mit dem AWS Command Line Interface

Themen

- Starten der Videoanalyse
- Abrufen des Abschlussstatus einer Amazon-Rekognition-Video-Analyseanforderungs
- Analyseergebnisse von Amazon Rekognition Video abrufen

Starten der Videoanalyse

Sie starten eine Anfrage zur Erkennung von Amazon Rekognition Video Video-Etiketten, indem Sie anrufen. StartLabelDetection Im Folgenden sehen Sie ein Beispiel für eine JSON-Anforderung, die von `StartLabelDetection` übergeben wird.

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

Der Eingabeparameter `Video` gibt den Namen der Videodatei und den Amazon-S3-Bucket an, aus dem sie abgerufen werden soll. `NotificationChannel` enthält den Amazon-Ressourcennamen (ARN) (ARN) des Amazon-SNS-Themas, das Amazon Rekognition Video benachrichtigt, wenn die Videoanalyseanforderung abgeschlossen ist. Das Amazon-SNS-Thema muss sich in derselben AWS-Region befinden wie der Amazon-Rekognition-Video-Endpunkt, den Sie aufrufen. `NotificationChannel` enthält auch den ARN für eine Rolle, die es Amazon Rekognition Video erlaubt, auf dem Amazon-SNS-Thema zu veröffentlichen. Sie erteilen Amazon Rekognition Veröffentlichungsberechtigungen für Ihre Amazon-SNS-Themen, indem Sie eine IAM-Servicerolle erstellen. Weitere Informationen finden Sie unter Amazon Rekognition Video konfigurieren.

Sie können auch einen optionalen Eingabeparameter, `JobTag`, angeben, der es Ihnen ermöglicht, den Auftrag im Abschlussstatus zu identifizieren, der im Amazon-SNS-Thema veröffentlicht wird.

Um ein versehentliches doppeltes Ausführen von Analyseaufträgen zu vermeiden, können Sie optional ein idempotentes Token, `ClientRequestToken`, bereitstellen. Wenn Sie einen Wert für `ClientRequestToken` angeben, liefert die Operation `Start` die gleiche `JobId` für mehrere identische Aufrufe zur Startoperation, wie z. B. `StartLabelDetection`. Ein Token `ClientRequestToken` hat eine Lebensdauer von 7 Tagen. Nach 7 Tagen können Sie es wiederverwenden. Wenn Sie das Token während der Token-Lebensdauer wiederverwenden, geschieht folgendes:

- Wenn Sie das Token mit der gleichen `Start`-Operation und den gleichen Eingabeparametern wiederverwenden, wird dieselbe `JobId` zurückgegeben. Der Auftrag wird nicht erneut ausgeführt und Amazon Rekognition Video sendet keinen Abschlussstatus an das registrierte Amazon-SNS-Thema.
- Wenn Sie das Token mit der gleichen `Start`-Operation und einer geringfügigen Änderung der Eingabeparameter wiederverwenden, wird eine `IdempotentParameterMismatchException`-Ausnahme (HTTP-Statuscode: 400) ausgelöst.
- Sie sollten ein Token nicht wiederholt bei verschiedenen `Start`-Operationen verwenden, da Sie unvorhersehbare Ergebnisse von Amazon Rekognition erhalten.

Die Antwort auf die `StartLabelDetection`-Operation ist eine Auftrags-ID (`JobId`). Verwenden Sie `JobId`, um Anforderungen zu verfolgen und die Analyseergebnisse zu erhalten, nachdem Amazon Rekognition Video den Abschlussstatus im Amazon-SNS-Thema veröffentlicht hat. Zum Beispiel:

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

Wenn Sie zu viele Aufträge gleichzeitig starten, lösen Aufrufe von `StartLabelDetection` eine `LimitExceededException` (HTTP-Statuscode: 400) aus, bis die Anzahl der gleichzeitig ausgeführten Aufträge unter dem Amazon-Rekognition-Servicelimit liegt.

Wenn Sie feststellen, dass `LimitExceededException`-Ausnahmen bei Spitzenaktivitäten ausgelöst werden, empfiehlt sich für die Verwaltung eingehender Anforderungen die Verwendung einer Amazon-SQS-Warteschlange. Wenden Sie sich an den AWS Support, wenn Sie feststellen, dass Ihre durchschnittliche Anzahl gleichzeitiger Anfragen nicht von einer Amazon SQS SQS-Warteschlange verwaltet werden kann und Sie immer noch Ausnahmen erhaltenLimitExceededException.

Abrufen des Abschlussstatus einer Amazon-Rekognition-Video-Analyseanforderungs

Amazon Rekognition Video sendet eine Benachrichtigung über den Abschluss der Analyse an das registrierte Amazon-SNS-Thema. Die Benachrichtigung enthält die Auftrags-ID und den Erledigungsstatus der Operation in einer JSON-Zeichenfolge. Eine erfolgreiche Videoanalyseanforderung hat einen Status SUCCEEDED. Das folgende Ergebnis zeigt z. B. die erfolgreiche Abarbeitung eines Label-Erkennungsauftrags.

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

Weitere Informationen finden Sie unter Referenz: Videoanalyse-Ergebnisbenachrichtigung.

Um die Statusinformationen zu erhalten, die von Amazon Rekognition Video im Amazon-SNS-Thema veröffentlicht werden, verwenden Sie eine der folgenden Optionen:

- AWS Lambda – Sie können eine AWS Lambda -Funktion abonnieren, die Sie in ein Amazon-SNS-Thema schreiben. Die Funktion wird aufgerufen, wenn Amazon Rekognition das Amazon-SNS-Thema benachrichtigt, dass die Anforderung abgeschlossen ist. Verwenden Sie eine Lambda-Funktion, wenn Sie serverseitigen Code zur Verarbeitung der Ergebnisse einer Videoanalyseanforderung benötigen. Sie können beispielsweise serverseitigen Code verwenden, um das Video mit Anmerkungen zu versehen oder um einen Bericht über die Videoinhalte zu erstellen, bevor die Informationen an eine Client-Anwendung zurückgegeben werden. Wir empfehlen auch die serverseitige Verarbeitung von großen Videos, da die Amazon-Rekognition-API große Datenmengen zurückliefern kann.
- Amazon Simple Queue Service – Sie können eine Amazon SQS-Warteschlange für ein Amazon-SNS-Thema abonnieren. Sie fragen dann die Amazon-Simple-Queue-Service-Warteschlange ab, um den von Amazon Rekognition veröffentlichten Abschlussstatus abzurufen, wenn eine Videoanalyseanforderung abgeschlossen ist. Weitere Informationen finden Sie unter Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK). Verwenden

Sie eine Amazon-SQS-Warteschlange, wenn Sie Amazon-Rekognition-Video-Operationen nur von einer Client-Anwendung aus aufrufen wollen.

> **Important**
>
> Wir raten davon ab, den Status der Auftragserfüllung durch wiederholtes Aufrufen der Amazon-Rekognition-Video-`Get`-Operation zu ermitteln. Der Grund hierfür ist, dass Amazon Rekognition Video die `Get`-Operation drosselt, wenn zu viele Anforderungen generiert werden. Wenn Sie mehrere Videos gleichzeitig verarbeiten, ist es einfacher und effizienter, eine SQS-Warteschlange für die Benachrichtigung über die Fertigstellung zu überwachen, als Amazon Rekognition Video für den Status jedes einzelnen Videos abzufragen.

Analyseergebnisse von Amazon Rekognition Video abrufen

Um die Ergebnisse einer Videoanalyseanforderung zu erhalten, stellen Sie zunächst sicher, dass der Abschlussstatus, der aus dem Amazon-SNS-Thema abgerufen wird, `SUCCEEDED` lautet. Rufen Sie dann `GetLabelDetection` auf, wodurch der Wert `JobId` übergeben wird, den `StartLabelDetection` zurückgibt. Die JSON-Ausgabe sieht folgendermaßen oder ähnlich aus:

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

JobId ist die Kennung für den Videoanalysevorgang. Da die Videoanalyse große Datenmengen erzeugen kann, geben Sie mit `MaxResults` die maximale Anzahl der Ergebnisse an, die in einer einzigen Get-Operation zurückgegeben werden soll. Der Standardwert von `MaxResults` beträgt 1000. Wenn Sie einen größeren Wert als 1.000 angeben, wird die maximale Anzahl von 1.000 Ergebnissen zurückgegeben. Wenn die Operation nicht die gesamte Ergebnismenge zurückgibt, wird in der Antwort der Operation ein Paginierungs-Token für die nächste Seite zurückgegeben. Wenn Sie ein Paginierung-Token aus einer vorherigen Get-Anforderung haben, verwenden Sie es mit `NextToken` um die nächste Seite der Ergebnisse zu erhalten.

> **Note**
>
> Amazon Rekognition behält die Ergebnisse einer Videoanalyse für 7 Tage bei. Nach dieser Zeit können Sie die Analyseergebnisse nicht mehr abrufen.

Die `GetLabelDetection` Operation Response JSON ist ähnlich wie folgt:

```
{
    "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
    }
}
```

Mit den `GetLabelDetection`- und `GetContentModeration`-Operationen können Sie die Analyseergebnisse nach Zeitstempel oder Labelnamen sortieren. Sie können die Ergebnisse auch nach Videosegment oder nach Zeitstempel zusammenfassen.

Sie können die Ergebnisse nach der Erkennungszeit (Millisekunden vom Anfang des Videos) oder alphabetisch nach der erkannten Entität (Objekt, Gesicht, Prominenter, Moderationslabel oder Person) sortieren. Um nach der Zeit zu sortieren, setzen Sie den Wert des Eingabeparameters `SortBy` auf `TIMESTAMP`. Wenn `SortBy` nicht angegeben ist, wird standardmäßig nach der Zeit sortiert. Das vorhergehende Beispiel ist nach der Zeit sortiert. Um nach Entität zu sortieren, verwenden Sie den Eingabeparameter `SortBy` mit dem Wert, der für die von Ihnen durchgeführte Operation geeignet ist. Um z. B. bei einem Aufruf von `GetLabelDetection` nach erkanntem Label zu sortieren, verwenden Sie den Wert `NAME`.

Um Ergebnisse nach Zeitstempel zu aggregieren, setzen Sie den Wert des `AggregateBy`-Parameters auf `TIMESTAMPS`. Um nach Videosegmenten zu aggregieren, legen Sie den Wert von `AggregateBy` auf `SEGMENTS` fest. Der `SEGMENTS`-Aggregationsmodus aggregiert die Labels im Laufe der Zeit und `TIMESTAMPS` gibt gleichzeitig den Zeitstempel an, bei dem ein Label erkannt wurde. Dabei werden 2-FPS-Sampling und Frame-Ausgabe verwendet (Hinweis: Diese aktuelle Sampling-Rate kann sich ändern, es sollten keine Annahmen über die aktuelle Sampling-Rate getroffen werden). Wenn kein Wert angegeben wird, ist die Standardaggregationsmethode `TIMESTAMPS`.

Amazon Rekognition Video konfigurieren

Um die Amazon-Rekognition-Video-API mit gespeicherten Videos zu verwenden, müssen Sie den Benutzer und eine IAM-Servicerolle für den Zugriff auf Ihre Amazon-SNS-Themen konfigurieren. Sie müssen auch eine Amazon-SQS-Warteschlange für Ihre Amazon-SNS-Themen abonnieren.

> **Note**
>
> Wenn Sie anhand dieser Anweisungen das Beispiel Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) einrichten, müssen Sie die Schritte 3, 4, 5 und 6 nicht ausführen. Das Beispiel enthält Code zum Erstellen und Konfigurieren des Amazon-SNS-Themas und der Amazon-SQS-Warteschlange.

Die Beispiele in diesem Abschnitt erstellen ein neues Amazon-SNS-Thema durch Verwendung der Anweisungen, die Amazon-Rekognition-Video-Zugriff auf mehrere Themen gewähren. Wenn Sie ein

vorhandenes Amazon-SNS-Thema verwenden möchten, verwenden Sie für Schritt 3 Der Zugriff auf ein bestehendes Amazon-SNS-Thema ermöglichen.

So konfigurieren Sie Amazon Rekognition Video

1. Richten Sie ein AWS Konto für den Zugriff auf Amazon Rekognition Video ein. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.
2. Installieren und konfigurieren Sie das erforderliche SDK. AWS Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.
3. Um die Codebeispiele in diesem Entwicklerhandbuch auszuführen, stellen Sie sicher, dass der von Ihnen gewählte Benutzer programmgesteuerten Zugriff hat. Weitere Informationen finden Sie unter Erteilen programmgesteuerten Zugriffs.

   Ihr Benutzer benötigt außerdem mindestens folgende Berechtigungen:

   - SQSFullZugriff auf Amazon
   - AmazonRekognitionFullAccess
   - Amazon S3 FullAccess
   - SNSFullZugriff auf Amazon

   Wenn Sie IAM Identity Center zur Authentifizierung verwenden, fügen Sie die Berechtigungen dem Berechtigungssatz für Ihre Rolle hinzu. Andernfalls fügen Sie die Berechtigungen zu Ihrer IAM-Rolle hinzu.
4. Erstellen Sie ein Amazon-SNS-Thema mit der Amazon-SNS-Konsole. Stellen Sie dem Namen des Themas Folgendes voran AmazonRekognition. Notieren Sie den ARN (Amazon-Ressourcenname). Stellen Sie sicher, dass das Thema sich in derselben Region befindet wie der verwendete AWS -Endpunkt.
5. Mit der Amazon-SQS-Konsole erstellen Sie eine Amazon-SQS-Standardwarteschlange. Notieren Sie den ARN der Warteschlange.
6. Abonnieren Sie die Warteschlange zum Thema, das Sie in Schritt 3 erstellt haben.
7. Erteilen der Berechtigung für das Amazon-SNS-Thema zum Senden von Nachrichten an die Amazon-SQS-Warteschlange.
8. Erstellen Sie eine IAM-Servicerolle, um Amazon Rekognition Video Zugriff auf Ihre Amazon-SNS-Themen zu gewähren. Notieren Sie sich den Amazon-Ressourcennamen (ARN) der

Service-Rolle. Weitere Informationen finden Sie unter Den Zugriff auf mehrfache Amazon-SNS-Themen ermöglichen.

9. Um sicherzustellen, dass Ihr Konto sicher ist, sollten Sie den Zugriff von Rekognition auf die Ressourcen beschränken, die Sie verwenden. Dies kann erreicht werden, indem Sie Ihrer IAM-Servicerolle eine Vertrauensrichtlinie hinzufügen. Weitere Informationen hierzu finden Sie unter Serviceübergreifende Confused-Deputy-Prävention.
10. Fügen Sie die folgende eingebundene Richtlinie an den Benutzer an, den Sie in Schritt 1 erstellt haben:

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

Geben Sie der eingebundenen Richtlinie einen beliebigen Namen.

11. Wenn Sie einen vom Kunden verwalteten AWS Key Management Service Schlüssel verwenden, um die Videos in Ihrem Amazon S3 S3-Bucket zu verschlüsseln, fügen Sie dem Schlüssel Berechtigungen hinzu, die es der in Schritt 7 erstellten Servicerolle ermöglichen, die Videos zu entschlüsseln. Für die Servicerolle sind mindestens Berechtigungen für `kms:GenerateDataKey`- und `kms:Decrypt`-Aktionen erforderlich. Zum Beispiel:

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

```
}
```

Weitere Informationen finden Sie unter Mein Amazon-S3-Bucket hat eine Standardverschlüsselung mit einem benutzerdefinierten AWS-KMS-Schlüssel. Wie kann ich Benutzern erlauben, Daten aus dem Bucket herunterzuladen und in ihn hochzuladen? und Schutz von Daten mit serverseitiger Verschlüsselung mit KMS-Schlüssel, die im AWS Key Management Service (SSE-KMS) gespeichert sind.

12. Jetzt können Sie die Beispiele in Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) und Analysieren eines Videos mit dem AWS Command Line Interface ausführen.

Den Zugriff auf mehrfache Amazon-SNS-Themen ermöglichen

Sie verwenden eine IAM-Servicerolle, um Amazon Rekognition Video Zugriff auf die von Ihnen erstellten Amazon-SNS-Themen zu geben. IAM stellt den Anwendungsfall Rekognition für die Erstellung einer Amazon-Rekognition-Video-Servicerolle zur Verfügung.

Sie können Amazon Rekognition Video Zugriff auf mehrere Amazon SNS SNS-Themen gewähren, indem Sie die `AmazonRekognitionServiceRole` Berechtigungsrichtlinie verwenden und den Themennamen Folgendes voranstellen — zum Beispiel. AmazonRekognition`AmazonRekognitionMyTopicName`

Um Amazon Rekognition Video Zugriff auf mehrere Amazon-SNS-Themen zu gewähren

1. Erstellen Sie eine IAM-Servicerolle. Verwenden Sie die folgenden Informationen, um die IAM-Servicerolle zu erstellen:
   1. Wählen Sie Rekognition als Servicenamen.
   2. Wählen Sie Rekognition für den Anwendungsfall der Service-Rolle. Die Berechtigungsrichtlinie sollte aufgeführt sein. AmazonRekognitionServiceRole AmazonRekognitionServiceRolegewährt Amazon Rekognition Video Zugriff auf Amazon SNS SNS-Themen, denen ein Präfix vorangestellt ist. AmazonRekognition
   3. Geben Sie der Servicerolle einen beliebigen Namen.
2. Notieren Sie den ARN der Service-Rolle. Sie benötigen ihn, um Videoanalyse-Operationen zu starten.

Der Zugriff auf ein bestehendes Amazon-SNS-Thema ermöglichen

Sie können eine Berechtigungsrichtlinie erstellen, die Amazon Rekognition Video Zugriff auf ein bestehendes Amazon-SNS-Thema erlaubt.

So gewähren Sie Amazon Rekognition Video Zugriff auf ein vorhandenes Amazon-SNS-Thema

1. Erstellen Sie mit dem IAM JSON Policy Editor eine neue Berechtigungsrichtlinie und verwenden Sie die folgende Richtlinie. Ersetzen Sie `topicarn` mit dem Amazon-Ressourcenname (ARN) des gewünschten Amazon-SNS-Themas.

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

2. Erstellen Sie ein IAM-Servicerolle oder aktualisieren Sie eine bestehende IAM-Servicerolle. Verwenden Sie die folgenden Informationen, um die IAM-Servicerolle zu erstellen:

   1. Wählen Sie Rekognition als Servicenamen.
   2. Wählen Sie Rekognition für den Anwendungsfall der Service-Rolle.
   3. Fügen Sie die Berechtigungsrichtlinie hinzu, die Sie in Schritt 1 erstellt haben.

3. Notieren Sie den ARN der Service-Rolle. Sie benötigen ihn, um Videoanalyse-Operationen zu starten.

Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK)

Dieses Verfahren zeigt Ihnen, wie Sie mit Hilfe von Amazon-Rekognition-Video-Labelerkennungsoperationen, einem in einem Amazon-S3-Bucket gespeicherten Video und einem Amazon-SNS-Thema Labels in einem Video erkennen. Die Prozedur zeigt auch, wie Sie eine

Amazon-SQS-Warteschlange verwenden, um den Abschlussstatus vom Amazon-SNS-Thema zu erhalten. Weitere Informationen finden Sie unter Amazon-Rekognition-Video-Operationen aufrufen. Sie sind nicht auf die Verwendung einer Amazon-SQS-Warteschlange beschränkt. Sie können beispielsweise eine AWS Lambda Funktion verwenden, um den Abschlussstatus abzurufen. Weitere Informationen finden Sie unter Aufrufen von Lambda-Funktionen unter Verwendung von Amazon SNS-Benachrichtigungen.

Der Beispielcode in dieser Prozedur zeigt Ihnen, wie Sie die folgenden Schritte durchführen können:

1. Amazon-SNS-Thema erstellen.
2. Amazon-SQS-Warteschlange erstellen.
3. Geben Sie Amazon Rekognition Video die Berechtigung, den Abschlussstatus einer Videoanalyseoperation im Amazon-SNS-Thema zu veröffentlichen.
4. Abonnieren der Amazon-SQS-Warteschlange für das Amazon-SNS-Thema.
5. Starten Sie die Videoanalyse-Anfrage durch Aufruf von StartLabelDetection.
6. Rufen Sie den Status der Erledigung aus der Amazon-SQS-Warteschlange ab. Das Beispiel verfolgt die Auftragskennung (`JobId`), die in `StartLabelDetection` zurückgegeben wird, und liefert nur die Ergebnisse für übereinstimmende Auftragskennungen, die aus dem Erledigungsstatus gelesen werden. Dies ist ein wichtiger Aspekt, wenn andere Anwendungen die gleiche Warteschlange und das gleiche Thema verwenden. Der Einfachheit halber werden in diesem Beispiel Aufträge gelöscht, die nicht übereinstimmen. Ziehen Sie in Erwägung, sie zur weiteren Untersuchung zu einer Amazon-SQS-Warteschlange für unzustellbare Nachrichten hinzufügen.
7. Rufen Sie GetLabelDetection auf und zeigen Sie die Ergebnisse der Videoanalyse an.

Voraussetzungen

Der Beispielcode für dieses Verfahren wird in Java und Python bereitgestellt. Sie müssen das entsprechende AWS SDK installiert haben. Weitere Informationen finden Sie unter Erste Schritte mit Amazon Rekognition. Das AWS-Konto, das Sie verwenden, muss über die Zugriffsberechtigungen für die Amazon-Rekognition-API verfügen. Weitere Informationen finden Sie unter Von Amazon Rekognition definierte Aktionen.

Um Labels in einem Video zu erkennen

1. Konfigurieren Sie den Benutzerzugriff auf Amazon Rekognition Video und den Zugriff auf Amazon Rekognition Video auf Amazon SNS. Weitere Informationen finden Sie unter Amazon

Rekognition Video konfigurieren. Sie müssen die Schritte 3, 4, 5 und 6 nicht ausführen, da der Beispielcode das Amazon SNS-Thema und die Amazon SQS-Warteschlange erstellt und konfiguriert.

2. Laden Sie eine Videodatei im MOV- oder MPEG-4-Format in einen Amazon-S3-Bucket hoch. Laden Sie zu Testzwecken ein Video hoch, das nicht länger als 30 Sekunden ist.

   Weitere Anleitungen finden Sie unter Upload eines Objekts in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service.

3. Verwenden Sie die folgenden Codebeispiele, um Labels in einem Video zu erkennen.

   Java

   In der `main`-Funktion:

   - Ersetzen Sie `roleArn` durch den ARN der IAM-Servicerolle, die Sie in Schritt 7 von So konfigurieren Sie Amazon Rekognition Video erstellt haben.
   - Ersetzen Sie die Werte von `amzn-s3-demo-bucket` und `video` durch den Bucket-Namen und den Namen der Videodatei, die Sie in Schritt 2 angegeben haben.

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

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

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

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

public class VideoDetect {


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

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


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

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

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

        CreateTopicandQueue();

        //=================================================
```

```
        StartLabelDetection(bucket, video);

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

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


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

    }


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


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

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

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

                    // Get status and job id from notification.
                    ObjectMapper mapper = new ObjectMapper();
```

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


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

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

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

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

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

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


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

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

   }

   private static void GetLabelDetectionResults() throws Exception{

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

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

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


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

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

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


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

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

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

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

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

    }

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

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

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

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

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


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

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

Python

In der `main`-Funktion:

- Ersetzen Sie `roleArn` durch den ARN der IAM-Servicerolle, die Sie in Schritt 7 von So konfigurieren Sie Amazon Rekognition Video erstellt haben.
- Ersetzen Sie die Werte von `amzn-s3-demo-bucket` und `video` durch den Bucket-Namen und den Namen der Videodatei, die Sie in Schritt 2 angegeben haben.
- Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.
- Sie können auch Filterkriterien in den Einstellungsparameter aufnehmen. Sie können beispielsweise einen `LabelsInclusionFilter` oder einen `LabelsExclusionFilter` zusammen mit einer Liste gewünschter Werte verwenden. Im folgenden Code können

Sie den `Features`- und `Settings`-Abschnitt auskommentieren und Ihre eigenen Werte angeben, um die zurückgegebenen Ergebnisse auf die Labels zu beschränken, an denen Sie interessiert sind.

- Im Aufruf an `GetLabelDetection` können Sie Werte für die `SortBy`- und `AggregateBy`-Argumente angeben. Um nach der Zeit zu sortieren, setzen Sie den Wert des Eingabeparameters `SortBy` auf `TIMESTAMP`. Um nach Entität zu sortieren, verwenden Sie den Eingabeparameter `SortBy` mit dem Wert, der für die von Ihnen durchgeführte Operation geeignet ist. Um Ergebnisse nach Zeitstempel zu aggregieren, setzen Sie den Wert des `AggregateBy`-Parameters auf `TIMESTAMPS`. Um nach Videosegmenten zu aggregieren, verwenden Sie `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,
MessageAttributeNames=['ALL'],
                                                   MaxNumberOfMessages=10)

            if sqsResponse:

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

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

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

ReceiptHandle=message['ReceiptHandle'])
                    else:
                        print("Job didn't match:" +
                              str(rekMessage['JobId']) + ' : ' +
self.startJobId)
```

```
                    # Delete the unknown message. Consider sending to dead
letter queue
                    self.sqs.delete_message(QueueUrl=self.sqsQueueUrl,

ReceiptHandle=message['ReceiptHandle'])

        return succeeded

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

NotificationChannel={'RoleArn': self.roleArn,

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

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

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

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

            print('Codec: ' + response['VideoMetadata']['Codec'])
            print('Duration: ' + str(response['VideoMetadata']
['DurationMillis']))
```

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

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

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

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

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

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

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

    def CreateTopicandQueue(self):
```

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

        # Create SNS topic

        snsTopicName = "AmazonRekognitionExample" + millis

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

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

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

        sqsQueueArn = attribs['QueueArn']

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

        # Authorize SNS to write SQS queue
        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

Im folgenden Beispielcode:

- Ersetzen Sie den Wert von `REGION` durch den Namen der Betriebsregion Ihres Kontos.

- Ersetzen Sie den Wert von `amzn-s3-demo-bucket` durch den Namen des Amazon-S3-Buckets, der Ihre Videodatei enthält.
- Ersetzen Sie den Wert von `videoName` durch den Namen der Videodatei in Ihrem Amazon-S3-Bucket.
- Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.
- Ersetzen Sie `roleArn` durch den ARN der IAM-Servicerolle, die Sie in Schritt 7 von So konfigurieren Sie Amazon Rekognition Video erstellt haben.

```
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()
     // Searh for pagination token, if found, set variable to next token
     if (String(response).includes("NextToken")){
       paginationToken = response.NextToken

     }else{
       finished = true
     }

     })
```

```
  }
}

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

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

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

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

```
  }
};

runLabelDetectionAndGetResults()
```

Java V2

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

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

/**
```

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

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

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

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

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

    SqsClient sqs = SqsClient.builder()
        .region(Region.US_WEST_2)
```

```
            .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
            .build();

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

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

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

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

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

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

            boolean ans = true;
            String status = "";
```

```
        int yy = 0;
        while (ans) {

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

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

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

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

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

    } catch(RekognitionException | InterruptedException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {

    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message: messages) {
                String notification = message.body();
```

```
                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonMessageTree = mapper.readTree(notification);
                JsonNode messageBodyText = jsonMessageTree.get("Message");
                ObjectMapper operationResultMapper = new ObjectMapper();
                JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                JsonNode operationJobId = jsonResultTree.get("JobId");
                JsonNode operationStatus = jsonResultTree.get("Status");
                System.out.println("Job found in JSON is " + operationJobId);

                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

                String jobId = operationJobId.textValue();
                if (startJobId.compareTo(jobId)==0) {
                    System.out.println("Job id: " + operationJobId );
                    System.out.println("Status : " +
operationStatus.toString());

                    if (operationStatus.asText().equals("SUCCEEDED"))
                        GetResultsLabels(rekClient);
                    else
                        System.out.println("Video analysis failed");

                    sqs.deleteMessage(deleteMessageRequest);
                }

                else{
                    System.out.println("Job received was not job " +
startJobId);
                    sqs.deleteMessage(deleteMessageRequest);
                }
            }
        }

    } catch(RekognitionException e) {
        e.getMessage();
        System.exit(1);
    } catch (JsonMappingException e) {
        e.printStackTrace();
```

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

// Gets the job results by calling GetLabelDetection
private static void GetResultsLabels(RekognitionClient rekClient) {

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

    try {
        do {
            if (labelDetectionResult !=null)
                paginationToken = labelDetectionResult.nextToken();


            GetLabelDetectionRequest labelDetectionRequest=
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
                .maxResults(maxResults)
                .nextToken(paginationToken)
                .build();

            labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
            VideoMetadata videoMetaData=labelDetectionResult.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());

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

                System.out.println("   Label:" + label.name());
                System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());
```

```
                List<Instance> instances = label.instances();
                System.out.println("   Instances of " + label.name());

                if (instances.isEmpty()) {
                    System.out.println("        " + "None");
                } else {
                    for (Instance instance : instances) {
                        System.out.println("        Confidence: " +
instance.confidence().toString());
                        System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                    }
                }
                System.out.println("   Parent labels for " + label.name() +
":");
                List<Parent> parents = label.parents();

                if (parents.isEmpty()) {
                    System.out.println("        None");
                } else {
                    for (Parent parent : parents) {
                        System.out.println("   " + parent.name());
                    }
                }
                System.out.println();
            }
        } while (labelDetectionResult !=null &&
labelDetectionResult.nextToken() != null);

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

4. Erstellen Sie den Code und führen Sie ihn aus. Die Operation kann einige Zeit in Anspruch nehmen. Nach Abschluss des Vorgangs wird eine Liste der im Video erkannten Labels angezeigt. Weitere Informationen finden Sie unter Erkennen von Labels in einem Video.

Analysieren eines Videos mit dem AWS Command Line Interface

Sie können die AWS Command Line Interface (AWS CLI) verwenden, um Amazon Rekognition Video Video-Operationen aufzurufen. Das Entwurfsmuster entspricht der Verwendung der Amazon Rekognition Video API mit dem AWS SDK für Java oder einem anderen AWS. SDKs Weitere Informationen finden Sie unter Überblick für die Amazon-Rekognition-Video-API. Die folgenden Verfahren zeigen, wie Sie mit dem AWS CLI Labels in einem Video erkennen können.

Sie starten, Labels in einem Video zu erkennen, indem Sie `start-label-detection` aufrufen. Wenn Amazon Rekognition mit der Analyse des Videos fertig ist, wird der Fertigstellungsfortschritt an das im `--notification-channel`-Parameter von `start-label-detection` angegebenen Amazon-SNS-Thema gesendet. Sie können den Abschlussstatus erhalten, indem Sie eine Amazon-Simple-Queue-Service-Warteschlange (Amazon SQS) zum Amazon-SNS-Thema abonnieren. Sie befragen dann receive-message, um den Abschlussstatus aus der Amazon-SQS-Warteschlange abzurufen.

Beim Aufrufen von `StartLabelDetection` können Sie Ihre Ergebnisse filtern, indem Sie Filterargumente für die Argumente `LabelsInclusionFilter` und/oder `LabelsExclusionFilter` angeben. Weitere Informationen finden Sie unter Erkennen von Labels in einem Video.

Die Statusmeldung der Erledigung ist eine JSON-Struktur innerhalb der Antwort `receive-message`. Sie müssen den JSON aus der Antwort extrahieren. Informationen über den JSON-Erledigungsstatus finden Sie unter Referenz: Videoanalyse-Ergebnisbenachrichtigung. Wenn der Wert des Feldes `Status` des JSON-Status `SUCCEEDED` ist, können Sie die Ergebnisse der Videoanalyseanforderung erhalten, indem Sie `get-label-detection` aufrufen. Beim Aufrufen von `GetLabelDetection` können Sie die zurückgegebenen Ergebnisse mithilfe der Argumente `SortBy` und `AggregateBy` sortieren und aggregieren.

Die folgenden Prozeduren enthalten keinen Code zum Abfragen der Amazon-SQS-Warteschlange. Außerdem enthalten sie keinen Code, um den JSON zu analysieren, der von der Amazon-SQS-Warteschlange zurückgegeben wird. Ein Beispiel in Java finden Sie unter Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK).

Voraussetzungen

Um dieses Verfahren ausführen zu können, muss das AWS CLI installiert sein. Weitere Informationen finden Sie unter Erste Schritte mit Amazon Rekognition. Das AWS-Konto, das Sie verwenden, muss

über die Zugriffsberechtigungen für die Amazon-Rekognition-API verfügen. Weitere Informationen finden Sie unter Von Amazon Rekognition definierte Aktionen.

Um Amazon Rekognition Video zu konfigurieren und ein Video hochzuladen

1. Konfigurieren Sie den Benutzerzugriff auf Amazon Rekognition Video und den Zugriff auf Amazon Rekognition Video auf Amazon SNS. Weitere Informationen finden Sie unter Amazon Rekognition Video konfigurieren.
2. Laden Sie eine Videodatei im MOV- oder MPEG-4-Format in Ihren S3-Bucket hoch. Bei der Entwicklung und beim Testen empfehlen wir, kurze Videos mit einer Länge von maximal 30 Sekunden zu verwenden.

   Weitere Anleitungen finden Sie unter Upload eines Objekts in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service.

Um Labels in einem Video zu erkennen

1. Führen Sie den folgenden AWS CLI Befehl aus, um mit der Erkennung von Labels in einem Video zu beginnen.

```
aws rekognition start-label-detection --video '{"S3Object":{"Bucket":"amzn-s3-demo-
bucket","Name":"video-name"}}' \
 --notification-channel '{"SNSTopicArn":"TopicARN","RoleArn":"RoleARN"}' \
--region region-name  \
--features GENERAL_LABELS \
--profile profile-name \
--settings "{"GeneralLabels":{"LabelInclusionFilters":["Car"]}}
```

   Aktualisieren Sie die folgenden Werte:

   - Ändern Sie `amzn-s3-demo-bucket` und `videofile` in den Amazon-S3-Bucket-Namen und den Dateinamen, die Sie in Schritt 2 angegeben haben.
   - Ändern Sie `us-east-1` in die von Ihnen verwendete AWS-Region.
   - Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.
   - Ändern Sie `TopicARN` in den ARN des Amazon-SNS-Themas, das Sie in Schritt 3 von Amazon Rekognition Video konfigurieren erstellt haben.

- Ändern Sie `RoleARN` in den ARN der IAM-Servicerolle, die Sie in Schritt 7 von Amazon Rekognition Video konfigurieren erstellt haben.
- Falls erforderlich, können Sie die `endpoint-url` angeben. Die AWS-CLI sollte anhand der angegebenen Region automatisch die richtige Endpunkt-URL ermitteln. Wenn Sie jedoch einen Endpunkt von Ihrer privaten VPC aus verwenden, müssen Sie möglicherweise die `endpoint-url` angeben. Die Ressource AWS-Service-Endpunkte listet die Syntax für die Angabe von Endpunkt-URLs sowie die Namen und Codes für jede Region auf.
- Sie können auch Filterkriterien in den Einstellungsparameter aufnehmen. Sie können beispielsweise einen `LabelsInclusionFilter` oder einen `LabelsExclusionFilter` zusammen mit einer Liste gewünschter Werte verwenden.

Wenn Sie auf einem Windows-Gerät auf die CLI zugreifen, verwenden Sie doppelte Anführungszeichen anstelle von einfachen Anführungszeichen und maskieren Sie die inneren doppelten Anführungszeichen durch einen Backslash (d. h. \), um eventuell auftretende Parserfehler zu beheben. Ein Beispiel finden Sie unten:

```
aws rekognition start-label-detection --video "{\"S3Object\":{\"Bucket\":\"amzn-s3-
demo-bucket",\"Name\":\"video-name\"}}" --notification-channel "{\"SNSTopicArn\":
\"TopicARN\",\"RoleArn\":\"RoleARN\"}" \
--region us-east-1 --features GENERAL_LABELS --settings "{\"GeneralLabels\":
{\"LabelInclusionFilters\":[\"Car\"]}}" --profile profile-name
```

2. Notieren Sie den `JobId`-Wert in der Antwort. Die Antwort sollte dem folgenden JSON-Beispiel ähnlich sein.

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

3. Schreiben Sie Code, um die Amazon-SQS-Warteschlange nach dem JSON-Abschlussstatus abzufragen (unter Verwendung von receive-message).
4. Schreiben Sie Code, um das Feld `Status` aus dem JSON-Erledigungsstatus zu extrahieren.
5. Wenn der Wert von `Status` ist`SUCCEEDED`, führen Sie den folgenden AWS CLI Befehl aus, um die Ergebnisse der Etikettenerkennung anzuzeigen.

```
aws rekognition get-label-detection  --job-id JobId \
```

```
--region us-east-1 --sort-by TIMESTAMP aggregate-by TIMESTAMPS
```

Aktualisieren Sie die folgenden Werte:

- Ändern Sie `JobId` in die Auftrags-ID, die Sie in Schritt 2 notiert haben.
- Ändern Sie `Endpoint` und `us-east-1` in den AWS-Endpunkt und die Region, die Sie verwenden.

Das Ergebnis sollte dem folgenden JSON-Beispiel ähnlich sein:

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

Referenz: Videoanalyse-Ergebnisbenachrichtigung

Amazon Rekognition Video veröffentlicht die Ergebnisse einer Analyseanforderung von Amazon Rekognition Video, einschließlich des Abschlussstatus, zu einem Amazon-Simple-Notification-Service-Thema (Amazon SNS). Um die Benachrichtigung von einem Amazon SNS SNS-Thema zu erhalten, verwenden Sie eine Amazon Simple Queue Service-Warteschlange oder eine AWS Lambda Funktion. Weitere Informationen finden Sie unter the section called “Amazon-Rekognition-Video-Operationen aufrufen”. Ein Beispiel finden Sie unter Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK).

Die Nutzlast liegt im folgenden JSON-Format vor:

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

Name	Beschreibung
JobId	Die Auftragskennung. Entspricht einer Auftragskennung, die von einer `Start`-Operation zurückgegeben wird, wie z. B. StartPersonTracking.
Status	Der Status des Auftrags. Gültige Werte sind SUCCEEDED, FAILED oder ERROR.
API	Die Amazon-Rekognition-Video-Operation, die zur Analyse des Eingabevideos verwendet wird.

Name	Beschreibung
JobTag	Bezeichner für den Auftrag. Sie geben `JobTag` im Aufruf einer Start-Operation an, an wie z. B. StartLabelDetection.
Zeitstempel	Der Unix-Zeitstempel für das Ende des Auftrags.
Video	Details zum Video, das verarbeitet wurde. Umfasst den Dateinamen und den Amazon-S3-Bucket, in dem die Datei gespeichert ist.

Im Folgenden sehen Sie ein Beispiel für eine erfolgreiche Benachrichtigung, die an ein Amazon-SNS-Thema gesendet wurde.

```
{
  "JobId": "6de014b0-2121-4bf0-9e31-856a18719e22",
  "Status": "SUCCEEDED",
  "API": "LABEL_DETECTION",
  "Message": "",
  "Timestamp": 1502230160926,
  "Video": {
    "S3ObjectName": "video.mpg",
    "S3Bucket": "amzn-s3-demo-bucket"
  }
}
```

Fehlerbehebung für Amazon Rekognition Video

Nachfolgend werden Informationen zur Fehlerbehebung bei der Arbeit mit Amazon Rekognition Video und gespeicherten Videos behandelt.

Ich erhalte nie den Abschlussstatus, der an das Amazon-SNS-Thema gesendet wird.

Amazon Rekognition Video veröffentlicht Statusinformationen an ein Amazon-SNS-Thema, wenn die Videoanalyse abgeschlossen ist. In der Regel erhalten Sie Nachrichten zum Abschlussstatus, indem Sie das Thema mit einer Amazon-SQS-Warteschlange oder Lambda-Funktion abonnieren. Abonnieren Sie das Amazon-SNS-Thema per E-Mail, um die Nachrichten in Ihrem E-Mail-

Posteingang zu erhalten, die an Ihr Amazon-SNS-Thema gesendet werden. Weitere Informationen finden Sie unter Tutorial: Abonnieren eines Endpunkts für ein Amazon-SNS-Thema.

Wenn Sie die Nachricht in Ihrer Anwendung nicht erhalten, gehen Sie wie folgt vor:

- Stellen Sie sicher, dass die Analyse abgeschlossen ist. Überprüfen Sie den `JobStatus`-Wert in der Antwort der Get-Operation (z. B. `GetLabelDetection`). Wenn der Wert `IN_PROGRESS` ist, ist die Analyse noch nicht abgeschlossen und der Abschlussstatus wurde noch nicht an das Amazon-SNS-Thema veröffentlicht.
- Stellen Sie sicher, dass Sie über eine IAM-Servicerolle verfügen, die Amazon Rekognition Video die Berechtigung zum Veröffentlichen an Ihr Amazon-SNS-Thema gewährt. Weitere Informationen finden Sie unter Amazon Rekognition Video konfigurieren.
- Vergewissern Sie sich, dass die von Ihnen verwendete IAM-Servicerolle mithilfe von Rollenanmeldeinformationen im Amazon-SNS-Thema veröffentlichen kann und dass die Berechtigungen Ihrer Servicerolle sicher auf die von Ihnen verwendeten Ressourcen beschränkt sind. Führen Sie die folgenden Schritte aus:
  - Rufen Sie den Amazon-Ressourcennamen (ARN) des Benutzers ab:

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

  - Fügen Sie den ARN des Benutzers der Vertrauensbeziehung zur Rolle hinzu. Weitere Informationen finden Sie unter Ändern einer Rolle. In der folgenden Beispiel-Vertrauensrichtlinie werden die Anmeldeinformationen des Benutzers für die Rolle angegeben und die Berechtigungen der Servicerolle auf die Ressourcen beschränkt, die Sie verwenden (weitere Informationen zur sicheren Einschränkung des Umfangs der Berechtigungen einer Servicerolle finden Sie unter Serviceübergreifende Confused-Deputy-Prävention):

    ```
    {
      "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/*"
                }
      }
    }
  ]
}
```

- Nehmen Sie die Rolle an: `aws sts assume-role --role-arn arn:Role ARN --role-session-name SessionName --profile RekognitionUser`
- Veröffentlichen im Amazon-SNS-Thema: `aws sns publish --topic-arn arn:Topic ARN --message "Hello World!" --region us-east-1 --profile RekognitionUser`

Wenn der AWS CLI-Befehl funktioniert, erhalten Sie die Nachricht (in Ihrem E-Mail-Posteingang, wenn Sie das Thema per E-Mail abonniert haben). Wenn Sie die Nachricht nicht erhalten:

- Überprüfen Sie, ob Amazon Rekognition Video konfiguriert ist. Weitere Informationen finden Sie unter Amazon Rekognition Video konfigurieren.
- Lesen Sie die anderen Tipps für diese Frage zur Problembehebung.

- Stellen Sie sicher, dass Sie das korrekte Amazon-SNS-Thema verwenden:
  - Wenn Sie Amazon Rekognition Video mithilfe einer IAM-Servicerolle Zugriff auf ein einzelnes Amazon-SNS-Thema gewähren, überprüfen Sie, dass Sie die Berechtigung für das korrekte Amazon-SNS-Thema gewährt haben. Weitere Informationen finden Sie unter Der Zugriff auf ein bestehendes Amazon-SNS-Thema ermöglichen.
  - Wenn Sie eine IAM-Servicerolle verwenden, um Amazon Rekognition Video Zugriff auf mehrere SNS-Themen zu gewähren, stellen Sie sicher, dass Sie das richtige Thema verwenden und dass dem Themennamen Folgendes vorangestellt wird. AmazonRekognition Weitere Informationen finden Sie unter Den Zugriff auf mehrfache Amazon-SNS-Themen ermöglichen.
  - Wenn Sie eine AWS Lambda Funktion verwenden, vergewissern Sie sich, dass Ihre Lambda-Funktion das richtige Amazon SNS SNS-Thema abonniert hat. Weitere Informationen finden Sie unter Fanout zu Lambda-Funktionen.
- Wenn Sie eine Amazon-SQS-Warteschlange für Ihr Amazon-SNS-Thema abonnieren, bestätigen Sie, dass Ihr Amazon-SNS-Thema über die Berechtigung zum Senden von Nachrichten an die Amazon-SQS-Warteschlange verfügt. Weitere Informationen finden Sie unter Erteilen der

Berechtigung für das Amazon-SNS-Thema zum Senden von Nachrichten an die Amazon-SQS-Warteschlange.

Ich benötige zusätzliche Hilfe zur Fehlerbehebung für das Amazon-SNS-Thema

Sie können Amazon SNS verwenden AWS X-Ray , um die Nachrichten, die Ihre Anwendung durchlaufen, zu verfolgen und zu analysieren. Weitere Informationen finden Sie unter Amazon SNS und AWS X-Ray.

Wenn Sie weitere Hilfe benötigen, können Sie Ihre Frage im Amazon-Rekognition-Forum stellen oder erwägen, sich für technischen AWS -Support anzumelden.

Arbeiten mit Streaming-Videoereignissen

Sie können Amazon Rekognition Video verwenden, um Gesichter oder Objekte in Streaming-Videos zu erkennen und zu identifizieren. Amazon Rekognition Video verwendet Amazon Kinesis Video Streams, um einen Videostrom zu empfangen und zu verarbeiten. Sie erstellen einen Stromprozessor mit Parametern, die angeben, was der Stromprozessor aus dem Videostrom erkennen soll. Rekognition sendet Labelerkennungsergebnisse von Streaming-Videoereignissen als Amazon-SNS- und Amazon-S3-Benachrichtigungen. Rekognition gibt die Ergebnisse der Gesichtssuche in einen Kinesis-Datenstrom aus.

Die Stromprozessoren für die Gesichtssuche verwenden `FaceSearchSettings`, um nach Gesichtern aus einer Sammlung zu suchen. Weitere Informationen zur Implementierung von Stromprozessoren für die Gesichtssuche zur Analyse von Gesichtern in Streaming-Videos finden Sie unter the section called “Suchen nach Gesichtern in einer Sammlung im Streaming-Video”.

Stromprozessoren zur Labelerkennung verwenden `ConnectedHomeSettings`, um bei Streaming-Videoereignissen nach Personen, Paketen und Haustieren zu suchen. Weitere Informationen zum Implementieren von Stromprozessoren zur Labelerkennung finden Sie unter the section called “Erkennung von Labels in Streaming-Videoereignissen”.

Überblick über den Betrieb des Amazon-Rekognition-Video-Stromprozessors

Sie beginnen die Analyse eines Streaming-Videos, indem Sie einen Amazon-Rekognition-Videostromprozessor starten und das Video in Amazon Rekognition Video streamen. Mit einem

Amazon-Rekognition-Video-Stromprozessor können Sie Stromprozessoren starten, beenden und verwalten. Sie erstellen einen Stream-Prozessor, indem Sie CreateStreamProcessor aufrufen. Zu den Anforderungsparametern für die Erstellung eines Streamprozessors für die Gesichtssuche gehören die Amazon-Ressourcennamen (ARNs) für den Kinesis-Videostream, den Kinesis-Datenstream und die Kennung für die Sammlung, die zur Erkennung von Gesichtern im Streaming-Video verwendet wird. Zu den Anforderungsparametern für die Erstellung eines Streamprozessors für die Sicherheitsüberwachung gehören die Amazon-Ressourcennamen (ARNs) für den Kinesis-Videostream und das Amazon SNS-Thema, die Objekttypen, die Sie im Videostream erkennen möchten, und Informationen für einen Amazon S3 S3-Bucket für die Ausgabeergebnisse. Auch ein Name, den Sie für den Stromprozessor angeben, gehört dazu.

Sie starten die Verarbeitung eines Videos durch Aufrufen der StartStreamProcessor-Operation. Um Statusinformationen für einen Stream-Prozessor zu erhalten, machen Sie den Aufruf DescribeStreamProcessor. Andere Operationen, die Sie aufrufen können TagResource, sind das Markieren eines Stream-Prozessors und DeleteStreamProcessordas Löschen eines Stream-Prozessors. Wenn Sie einen Stream-Prozessor für die Gesichtssuche verwenden, können Sie ihn auch StopStreamProcessorzum Stoppen eines Stream-Prozessors verwenden. Um eine Liste der Stream-Prozessoren in Ihrem Konto zu erhalten, rufen Sie ListStreamProcessors auf.

Nachdem der Stromprozessor gestartet wurde, streamen Sie das Video über den Kinesis-Videostrom, den Sie in `CreateStreamProcessor` angegeben haben, in Amazon Rekognition Video. Sie können den Kinesis Video Streams PutMediaSDK-Vorgang verwenden, um Video in den Kinesis-Videostream zu übertragen. Ein Beispiel finden Sie unter PutMediaAPI-Beispiel.

Informationen darüber, wie Ihre Anwendung Amazon-Rekognition-Video-Analyseergebnisse von einem Gesichtssuche-Stromprozessor verarbeiten kann, finden Sie unter Lesen von Streaming-Video-Analyseergebnissen.

Hinzufügen von Tags zu einem neuen Stromprozessor

Sie können Amazon-Rekognition-Stromprozessoren mithilfe von Tags identifizieren, organisieren, suchen und filtern. Jedes Tag ist ein Label, das aus einem benutzerdefinierten Schlüssel und Wert besteht.

Sie können einem Stromprozessor hinzufügen, während Sie ihn mit der `CreateStreamProcessor`-Operation erstellen. Geben Sie ein oder mehrere Tags im `Tags`-Array-Eingabeparameter an. Im Folgenden sehen Sie ein JSON-Beispiel für die Anforderung von `CreateStreamProcessor` mit Tags.

```
{
       "Name": "streamProcessorForCam",
       "Input": {
              "KinesisVideoStream": {
                     "Arn": "arn:aws:kinesisvideo:us-east-1:nnnnnnnnnnnn:stream/
inputVideo"
              }
       },
       "Output": {
              "KinesisDataStream": {
                     "Arn": "arn:aws:kinesis:us-east-1:nnnnnnnnnnnn:stream/outputData"
              }
       },
       "RoleArn": "arn:aws:iam::nnnnnnnnnnn:role/roleWithKinesisPermission",
       "Settings": {
              "FaceSearch": {
                     "CollectionId": "collection-with-100-faces",
                     "FaceMatchThreshold": 85.5
              },
              "Tags": {
      "Dept": "Engineering",
        "Name": "Ana Silva Carolina",
        "Role": "Developer"
       }
}
```

Hinzufügen von Tags zu einem bestehenden Stromprozessor

Sie können Amazon-Rekognition-Stromprozessoren mithilfe von Tags identifizieren, organisieren, suchen und filtern. Jedes Tag ist ein Label, das aus einem benutzerdefinierten Schlüssel und Wert besteht.

Verwenden Sie die `TagResource`-Operation, um einem vorhandenen Stromprozessor ein oder mehr Tags hinzuzufügen. Geben Sie den Amazon-Ressourcennamen (ARN) (`ResourceArn`) des Stromprozessors an und die Tags (`Tags`), die Sie hinzufügen möchten. Das folgende Beispiel zeigt, wie zwei Tags hinzugefügt werden.

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

> **Note**
>
> Wenn Sie den Amazon-Ressourcennamen des Stromprozessors nicht kennen, können Sie die `DescribeStreamProcessor`-Operation verwenden.

Hinzufügen von Tags in einem Stromprozessor

Sie können Amazon-Rekognition-Stromprozessoren mithilfe von Tags identifizieren, organisieren, suchen und filtern. Jedes Tag ist ein Label, das aus einem benutzerdefinierten Schlüssel und Wert besteht.

Um die an einen Stromprozessor angehängten Tags aufzulisten, verwenden Sie die `ListTagsForResource`-Operation und geben Sie den ARN des Stromprozessors an (`ResourceArn`). Die Antwort ist eine Zuordnung von Tag-Schlüsseln und -Werten, die an den angegebenen Stromprozessor angehängt sind.

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

In der Ausgabe wird eine Liste von Tags angezeigt, die an den Stromprozessor angehängt sind:

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

Löschen von Tags von einem Stromprozessor

Sie können Amazon-Rekognition-Stromprozessoren mithilfe von Tags identifizieren, organisieren, suchen und filtern. Jedes Tag ist ein Label, das aus einem benutzerdefinierten Schlüssel und Wert besteht.

Verwenden Sie die `UntagResource`-Operation, um ein oder mehr Tags von einem Stromprozessor zu entfernen. Geben Sie den ARN des Modells (`ResourceArn`) und die Tag-Schlüssel (`Tag-Keys`) an, die Sie entfernen möchten.

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

Alternativ können Sie Tag-Schlüssel in diesem Format angeben:

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

Fehlerbehandlung

Dieser Abschnitt beschreibt Laufzeitfehler und wie sie gehandhabt werden können. Es werden auch Fehlermeldungen und Codes beschrieben, die Amazon-Rekognition-spezifisch sind.

Themen

- Fehlerkomponenten
- Fehlermeldungen und Codes
- Fehlerbehandlung in Ihrer Anwendung

Fehlerkomponenten

Wenn das Programm eine Anforderung sendet, versucht Amazon Rekognition diese zu verarbeiten. Bei einer erfolgreichen Anforderung gibt Amazon Rekognition einen HTTP-Erfolgscode (`200 OK`) sowie die Ergebnisse der angeforderten Operation zurück.

Wenn die Anforderung nicht erfolgreich ist, gibt Amazon Rekognition einen Fehler zurück. Jeder Fehler hat drei Komponenten:

- Einen HTTP-Statuscode (z. B. `400`).
- Einen Ausnahmenamen (z. B. `InvalidS3ObjectException`).
- Eine Fehlermeldung (z. B. `Unable to get object metadata from S3. Check object key, region and/or access permissions.`).

AWS SDKs kümmert sich darum, Fehler an Ihre Anwendung weiterzugeben, sodass Sie entsprechende Maßnahmen ergreifen können. Sie können beispielsweise in einem Java-Programm eine `try-catch`-Logik schreiben, um eine `ResourceNotFoundException` zu verarbeiten.

Wenn Sie keinen AWS-SDK verwenden, müssen Sie den Inhalt der Low-Level-Antwort von Amazon Rekognition analysieren. Nachfolgend finden Sie ein Beispiel einer solchen Antwort:

```
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."}
```

Fehlermeldungen und Codes

Die folgende Liste enthält von Amazon Rekognition zurückgegebene Ausnahmen, gruppiert nach HTTP-Statuscode. Wenn OK, um es erneut zu versuchen? Ja ist, können Sie die gleiche Anforderung erneut senden. Wenn OK, um es erneut zu versuchen? auf Nein eingestellt ist, müssen Sie das Problem clientseitig lösen, bevor Sie eine neue Anforderung absenden.

HTTP-Statuscode 400

Der HTTP-Statuscode `400` weist auf ein Problem mit der Anforderung hin. Einige Beispiele für Probleme sind Authentifizierungsfehler, wobei es sich um erforderliche Parameter handelt, die fehlen oder den bereitgestellten Durchsatz der Operation überschreiten. Sie müssen das Problem zuerst in der Anwendung beheben, bevor Sie die Anforderung erneut senden.

AccessDeniedException

Meldung: Beim Aufrufen der Operation ist ein Fehler aufgetreten (AccessDeniedException): User: is not authorized to perform: on resource:. <Operation><User ARN><Operation><Resource ARN>

Sie sind nicht berechtigt, die Aktion auszuführen. Verwenden Sie den Amazon-Ressourcennamen (ARN) der IAM-Rolle oder eines autorisierten Benutzers, um die Operation auszuführen.

Erneut versuchen? Nein

GroupFacesInProgressException

Meldung: GroupFaces Job konnte nicht geplant werden. Es gibt einen bestehenden Auftrag für Gruppengesichter für diese Sammlung.

Wiederholen Sie die Operation, nachdem der vorhandene Auftrag ausgeführt wurde.

Erneut versuchen? Nein

IdempotentParameterMismatchException

Nachricht ClientRequestToken: Die von Ihnen angegebene Datei wird bereits verwendet. <Token>

Ein ClientRequestToken Eingabeparameter wurde bei einer Operation wiederverwendet, aber mindestens einer der anderen Eingabeparameter unterscheidet sich vom vorherigen Aufruf der Operation.

Erneut versuchen? Nein

ImageTooLargeException

Meldung: Image size is too large

Die Größe des Eingabebildes überschreitet die zulässige Grenze. Wenn Sie anrufen DetectProtectiveEquipment, überschreitet die Bildgröße oder Auflösung den zulässigen Grenzwert. Weitere Informationen finden Sie unter Richtlinien und Kontingente in Amazon Rekognition.

Erneut versuchen? Nein

InvalidImageFormatException

Meldung: Request has invalid image format

Das angegebene Bildformat wird nicht unterstützt. Verwenden Sie ein unterstütztes Bildformat (.JPEG oder .PNG). Weitere Informationen finden Sie unter Richtlinien und Kontingente in Amazon Rekognition.

Erneut versuchen? Nein

InvalidPaginationTokenException

Nachrichten

- Ungültiges Token
- Ungültiges Paginierungs-Token

Das Paginierungs-Token in der Anforderung ist ungültig. Das Token ist möglicherweise abgelaufen.

Erneut versuchen? Nein

InvalidParameterException

Meldung: Request has invalid parameters.

Ein Eingabeparameter verletzt eine Beschränkung. Validieren Sie die Parameter, bevor Sie die API-Operation erneut aufrufen.

Erneut versuchen? Nein

Ungültig (S3). ObjectException

Meldungen:

- Request has invalid S3 object.
- Objekt-Metadaten können nicht von S3 abgerufen werden. Prüfen Sie Objektschlüssel, Region und/oder Zugriffsberechtigungen.

Amazon Rekognition kann auf das in der Anforderung angegebene S3-Objekt nicht zugreifen. Weitere Informationen finden Sie unter Übersicht über die Verwaltung von Zugriffsberechtigungen für Ihre Amazon S3-Ressourcen. Informationen zur Fehlerbehebung finden Sie unter Amazon S3-Fehlerbehebung.

Erneut versuchen? Nein

LimitExceededException

Meldungen:

- Stream processor limit exceeded for account, limit - <Current Limit>.
- <Anzahl ausstehender Aufträge> open Jobs for User <Benutzer-ARN> Maximum limit: <Maximum>

Ein Servicelimit von Amazon Rekognition wurde überschritten. Wenn Sie beispielsweise zu viele Amazon-Rekognition-Video-Aufträge gleichzeitig starten, lösen Aufrufe zum Starten von Operationen (z. B. `StartLabelDetection`) eine `LimitExceededException`-Ausnahme (HTTP-Statuscode 400) aus, bis die Anzahl der gleichzeitig ausgeführten Aufträge unter dem Amazon-Rekognition-Service-Limit liegt.

Erneut versuchen? Nein

ProvisionedThroughputExceededException

Meldungen:

- Provisioned Rate exceeded.
- S3 download limit exceeded.

Die Anzahl der Anforderungen hat das Durchsatzlimit überschritten. Weitere Informationen finden Sie unter Amazon-Rekognition-Services-Limits.

Um eine Erhöhung des Limits zu beantragen, folgen Sie den Anweisungen unter the section called “Erstellen Sie einen Fall, um TPS-Kontingente zu ändern”.

Erneut versuchen? Ja

ResourceAlreadyExistsException

Meldung: The collection id: <Collection Id> already exists.

Eine Sammlung mit der angegebenen ID ist bereits vorhanden.

Erneut versuchen? Nein

ResourceInUseException

Meldungen:

- Stream processor name already in use.
- Specified resource is in use.
- Processor not available for stopping stream.
- Cannot delete stream processor.

Wiederholen Sie die Operation, wenn die Ressource verfügbar ist.

Erneut versuchen? Nein

ResourceNotFoundException

Meldung: Various messages depending on the API call.

Die angegebene Ressource ist nicht vorhanden.

Erneut versuchen? Nein

ThrottlingException

Meldung: Slow down; sudden increase in rate of requests.

Die Zunahme der Anforderungsrate ist zu hoch. Reduzieren Sie die Anforderungsrate und steigern Sie sie dann langsam. Wir empfehlen, die Anforderungsrate exponentiell zu reduzieren und es dann erneut zu versuchen. Standardmäßig AWS SDKs verwenden sie automatische Wiederholungslogik und exponentielles Backoff. Weitere Informationen finden Sie unter Wiederholversuche bei Fehlern und exponentielles Backoff in AWS und Exponentielles Backoff und Jitter.

Erneut versuchen? Ja

VideoTooLargeException

Meldung: Video size in bytes: <Video-Größe> is more than the maximum limit of: <Maximalgröße> bytes.

Die Dateigröße oder die Dauer der bereitgestellten Medien ist zu groß. Weitere Informationen finden Sie unter Richtlinien und Kontingente in Amazon Rekognition.

Erneut versuchen? Nein

HTTP-Statuscode 5xx

Der HTTP-Statuscode `5xx` weist auf ein Problem hin, das von AWS behoben werden muss. Dies kann ein vorübergehender Fehler sein. Wenn dies der Fall ist, können Sie Ihre Anforderung wiederholen, bis sie korrekt ausgeführt wird. Andernfalls rufen Sie das AWS Service Health Dashboard auf, um zu sehen, ob es Betriebsprobleme mit dem Service gibt.

InternalServerError (HTTP 500)

Meldung: Internal server error

Amazon Rekognition hat ein Service-Problem festgestellt. Wiederholen Sie den Aufruf. Reduzieren Sie die Anforderungsrate exponentiell und versuchen Sie es erneut. Standardmäßig AWS SDKs verwenden sie automatische Wiederholungslogik und exponentielles Backoff. Weitere Informationen finden Sie unter Wiederholversuche bei Fehlern und exponentielles Backoff in AWS und Exponentielles Backoff und Jitter.

Erneut versuchen? Ja

ThrottlingException (HTTP 500)

Meldung: Service Unavailable

Amazon Rekognition kann die Anforderung vorübergehend nicht verarbeiten. Wiederholen Sie den Aufruf. Wir empfehlen, die Anforderungsrate exponentiell zu reduzieren und es dann erneut zu versuchen. Standardmäßig AWS SDKs verwenden sie automatische Wiederholungslogik und exponentielles Backoff. Weitere Informationen finden Sie unter Wiederholversuche bei Fehlern und exponentielles Backoff in AWS und Exponentielles Backoff und Jitter.

Erneut versuchen? Ja

Fehlerbehandlung in Ihrer Anwendung

Damit Ihre Anwendung reibungslos ausgeführt wird, müssen Sie Logik zum Erfassen und Behandeln von Fehlern integrieren. Typische Ansätze umfassen die Verwendung von `try-catch`-Blöcken oder `if-then`-Anweisungen.

Die AWS SDKs führt ihre eigenen Wiederholungen und Fehlerprüfungen durch. Wenn Sie bei der Verwendung eines der AWS auf einen Fehler stoßen SDKs, können Ihnen der Fehlercode und die Beschreibung bei der Behebung helfen.

Sie sollten auch eine `Request ID` in der Antwort sehen. Die `Request ID` kann hilfreich sein, wenn Sie mit AWS Support ein Problem diagnostizieren müssen.

Der folgende Java-Codeausschnitt versucht, Objekte in einem Bild zu erkennen. Er führt eine rudimentäre Fehlerbehandlung durch. (In diesem Fall wird der Benutzer darüber informiert, dass die Anforderung fehlgeschlagen ist.)

```
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());
}
```

In diesem Codeausschnitt behandelt das `try-catch`-Konstrukt zwei unterschiedliche Arten von Ausnahmen:

- `AmazonRekognitionException` – Diese Ausnahme tritt auf, wenn die Client-Anforderung korrekt an Amazon Rekognition übermittelt wurde, aber Amazon Rekognition die Anforderung nicht verarbeiten konnte und stattdessen eine Fehlerantwort zurückgegeben hat.
- `AmazonClientException` – Diese Ausnahme wird ausgegeben, wenn der Client keine Antwort von einem Service erhalten oder die Antwort von einem Service nicht analysieren konnte.

Verwendung von Amazon Rekognition als FedRAMP-autorisierter Service

Das AWS FedRAMP-Compliance-Programm umfasst Amazon Rekognition als von FedRAMP autorisierten Service. Als Bundeskunde oder gewerblicher Kunde können Sie den Service nutzen, um sensible Workloads in den Regionen AWS USA Ost und USA West zu verarbeiten und zu speichern, wobei die Daten bis zur Stufe mit moderaten Auswirkungen reichen. Sie können den Service für sensible Workloads innerhalb der Autorisierungsgrenzen der Region AWS GovCloud (USA) verwenden, bei denen Daten bis zur höchsten Auswirkungsstufe verarbeitet werden. Weitere Informationen zur FedRAMP Compliance finden Sie unter AWS FedRAMP Compliance.

Um mit FedRAMP kompatibel zu sein, können Sie einen FIPS-Endpunkt (Federal Information Processing Standard) verwenden. Auf diese Weise erhalten Sie Zugriff auf für FIPS 140-2 validierte kryptographische Module, wenn Sie mit vertraulichen Informationen arbeiten. Weitere Informationen zu FIPS-Endpunkten finden Sie unter FIPS 140-2-Übersicht.

Sie können die AWS Command Line Interface (AWS CLI) oder eine der AWS-Optionen verwenden SDKs , um den Endpunkt anzugeben, der von Amazon Rekognition verwendet wird.

Informationen zu Endpunkten, die mit Amazon Rekognition verwendet werden können, finden Sie unter Amazon-Rekognition-Regionen und -Endpunkte.

Im Folgenden finden Sie Beispiele aus dem Thema Auflistung von Sammlungen im Amazon-Rekognition-Entwicklerhandbuch. Sie werden geändert, um die Region und den FIPS-Endpunkt anzugeben, über den auf Amazon Rekognition zugegriffen wird.

Java

Verwenden Sie für Java die `withEndpointConfiguration`-Methode, wenn Sie den Amazon-Rekognition-Client erstellen. In diesem Beispiel werden die Sammlungen gezeigt, die den FIPS-Endpunkt in der Region USA Ost (N.Virginia) verwenden:

```
//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

Verwenden Sie für die das `--endpoint-url` Argument AWS CLI, um den Endpunkt anzugeben, über den auf Amazon Rekognition zugegriffen wird. In diesem Beispiel werden die Sammlungen gezeigt, die den FIPS-Endpunkt in der Region USA Ost (Ohio) verwenden:

```
aws rekognition list-collections --endpoint-url https://rekognition-fips.us-
east-2.amazonaws.com --region us-east-2
```

Python

Verwenden Sie für Python das `endpoint_url`-Argument in der Funktion boto3.client. Legen Sie es auf den Endpunkt fest, den Sie angeben möchten. In diesem Beispiel werden die Sammlungen gezeigt, die den FIPS-Endpunkt in der Region USA West (Oregon) verwenden:

```
#Copyright 2018 Amazon.com, Inc. or its 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()
```

Bewährte Methoden für Sensoren, Eingabebilder und Videos

Dieser Abschnitt enthält Informationen zu bewährten Methoden für die Verwendung von Amazon Rekognition. Diese bewährten Methoden helfen Ihnen dabei, die von Ihnen aufgerufenen Operationen optimal zu nutzen. Wenn Sie Schwierigkeiten haben, mit einem Vorgang die erwarteten Ergebnisse zu erzielen, stellen Sie sicher, dass Sie die hier dokumentierten bewährten Methoden befolgen.

Informationen zur Latenz von Image-Vorgängen finden Sie im Folgenden:

- the section called “Latenz der Amazon-Rekognition-Image-Operation”

Für den Gesichtsvergleich und die Suche nach Gesichtern müssen Sie bestimmte bewährte Methoden befolgen, um Gesichter in einem Bild zu finden. Diese Anforderungen wurden auch unter dem folgenden Link dokumentiert:

- the section called “Empfehlungen zu Eingabebildern für den Gesichtsvergleich”
- the section called “Empfehlungen für die Suche nach Gesichtern in einer Sammlung”

In den folgenden Abschnitten wird beschrieben, wie Sie Ihre Kamera für jeden Medientyp einrichten, den Amazon Rekognition analysieren kann:

- the section called “Empfehlungen für die Kameraeinrichtung (Bild und Video)”
- the section called “Empfehlungen für die Kameraeinrichtung (Gespeicherte Bilder und Videostreaming)”
- the section called “Empfehlungen für die Kameraeinrichtung (Videostreaming)”

Für die Operationen „Face Liveness“ gelten auch eigene bewährte Methoden, die befolgt werden sollten, um die optimale Leistung aus dem Tool „Liveness Check“ herauszuholen:

- the section called “Empfehlungen für die Verwendung von Face Liveness”

Latenz der Amazon-Rekognition-Image-Operation

Um die geringstmögliche Latenzzeit für Amazon-Rekognition-Image-Operationen zu gewährleisten, sollten Sie Folgendes beachten:

- Die Region für den Amazon-S3-Bucket, der Ihre Bilder enthält, muss mit der Region übereinstimmen, die Sie für Amazon-Rekognition-Image-API-Operationen nutzen.
- Der Aufruf einer Amazon-Rekognition-Image-Operation mit Bildbytes ist schneller als das Hochladen des Bildes in einen Amazon-S3-Bucket und das anschließende Referenzieren des hochgeladenen Bilds in einer Amazon-Rekognition-Image-Operation. Beachten Sie diesen Ansatz, wenn Sie Bilder zu Amazon Rekognition Image hochladen, um eine Verarbeitung in nahezu Echtzeit durchzuführen. Beispielsweise bei Bildern, die von einer IP-Kamera oder über ein Webportal hochgeladen wurden.
- Wenn sich das Bild bereits in einem Amazon-S3-Bucket befindet, wird es wahrscheinlich schneller gehen, wenn Sie das Bild in einer Amazon-Rekognition-Image-Operation referenzieren, als wenn Sie Bild-Bytes an die Operation übergeben.

Empfehlungen zu Eingabebildern für den Gesichtsvergleich

Die Modelle, die für den Gesichtsvergleich verwendet werden, funktionieren für eine Vielzahl von Posen, Gesichtsausdrücken, Altersbereichen, Rotationen, Lichtverhältnissen und Größen. Es wird empfohlen, bei der Auswahl von Referenzfotos für CompareFaces oder beim Hinzufügen von Gesichtern zu einer Sammlung mit IndexFaces die folgenden Richtlinien zu beachten.

Allgemeine Empfehlungen für Eingabebilder für Gesichtsoperationen

- Die Aufnahmen sollten hell und scharf genug sein. Vermeiden Sie nach Möglichkeit verschwommene Bilder, bei denen sich die aufgenommene Person oder die Kamera bewegt hat. Mit DetectFaces können Helligkeit und Schärfe eines Gesichts bestimmt werden.
- Für die Blickerkennung wird empfohlen, das Originalbild in Originalgröße und -qualität hochzuladen.
- Verwenden Sie ein Bild, auf dem sich das Gesicht innerhalb des empfohlenen Bereichs von Winkeln befindet. Die Neigung (Pitch) sollte nach unten weniger 30 Grad und nach oben weniger als 45 Grad betragen. Die seitliche Ausschwenkung (Yaw) sollte in jeder Richtung weniger als 45 Grad betragen. Es gibt keine Beschränkungen bezüglich der Drehung um die Längsachse (Roll).
- Auf dem Bild von dem Gesicht sollten beide Augen offen und sichtbar sein.
- Das Gesicht sollte nicht verdeckt oder stark zugeschnitten sein. Das Bild sollte Kopf und Schultern der Person vollständig enthalten. Es sollte nicht auf den Begrenzungsrahmen zugeschnitten werden.

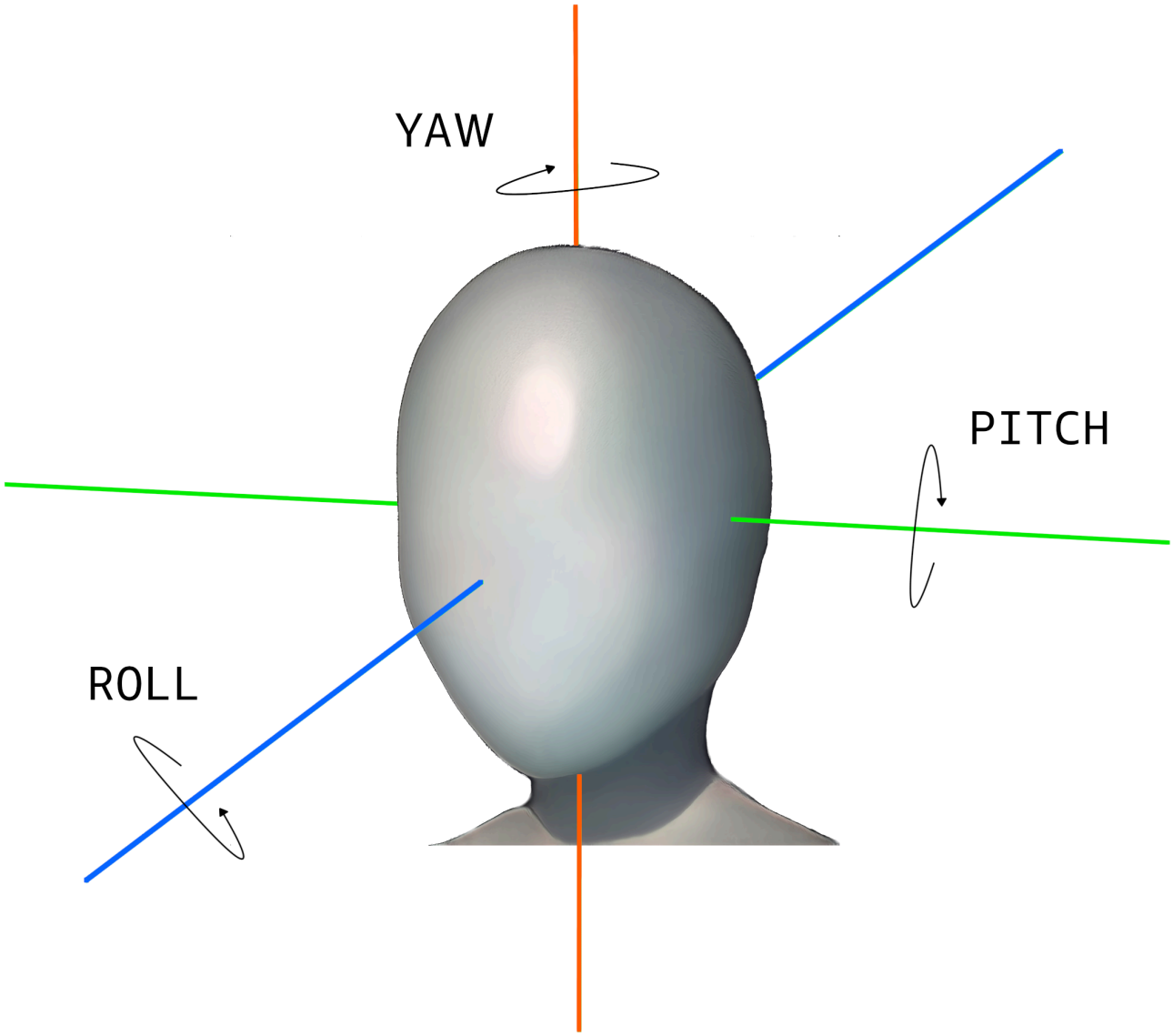
- Vermeiden Sie Gegenstände, die das Gesicht verbergen, wie z. B. Stirnbänder und Masken.
- Das Gesicht sollte einen großen Teil des Bildes abdecken. Bilder, bei denen das Gesicht einen größeren Teil ausmacht, werden mit größerer Genauigkeit abgeglichen.
- Stellen Sie sicher, dass die Auflösung der Bilder ausreichend groß ist. Amazon Rekognition kann Gesichter mit einer Größe von nur 50 x 50 Pixeln in Bildauflösungen von bis zu 1920 x 1080 erkennen. Bei Bildern mit höherer Auflösung ist die erforderliche minimale Gesichtsgröße größer. Auf Bildern, die größer als die Mindestgröße sind, lassen sich bessere Ergebnisse beim Gesichtsvergleich erzielen.
- Verwenden Sie Farbbilder.
- Das Gesicht sollte gleichmäßig ausgeleuchtet sein, d. h. es sollten keine Partien im Schatten liegen.
- Es sollte ein ausreichender Kontrast zum Hintergrund bestehen. Gut geeignet ist ein kontrastreicher monochromer Hintergrund.
- Für Verwendungszwecke, bei denen eine hohe Genauigkeit wichtig ist, sollten die Gesichtsausdrücke der Gesichter mit geschlossenem Mund und schwachem bis keinem Lächeln neutral sein.

Empfehlungen für die Suche nach Gesichtern in einer Sammlung

- Achten Sie bei der Suche nach Gesichtern in einer Sammlung darauf, dass aktuelle Gesichtsbilder indexiert sind.
- Wenn mit `IndexFaces` eine Sammlung erstellt wird, verwenden Sie mehrere Bilder des Gesichts einer Person mit verschiedenen Neigungen und seitlichen Ausschwenkungen (innerhalb des empfohlenen Bereichs von Winkeln). Wir empfehlen, dass mindestens fünf Bilder der Person indiziert werden – geradeaus, mit dem Gesicht nach links gedreht mit einem Gierwinkel von 45 Grad oder weniger, mit dem Gesicht nach rechts gedreht mit einem Gierwinkel von 45 Grad oder weniger, mit dem Gesicht nach unten gekippt mit einer Neigung von 30 Grad oder weniger und mit dem Gesicht nach oben gekippt mit einer Neigung von 45 Grad oder weniger. Wenn Sie diese Gesichts-Instances als zu demselben Gesicht gehörend nachverfolgen möchten, wird zur Verwendung des externen Bild-ID-Attributs geraten, wenn sich in dem zu indizierenden Bild nur ein Gesicht befindet. Zum Beispiel können fünf Bilder von John Doe in der Sammlung mit externen Bildern IDs als nachverfolgt werden. `John_Doe_1.jpg, … John_Doe_5.jpg`

Empfehlungen für die Kameraeinrichtung (Bild und Video)

Die folgenden Empfehlungen gelten zusätzlich zu Empfehlungen zu Eingabebildern für den Gesichtsvergleich.



- Bildauflösung: Es gibt keine Mindestanforderung für die Bildauflösung, solange die Gesichtsauflösung 50 x 50 Pixel für Bilder mit einer Gesamtauflösung von bis zu 1920 x 1080 beträgt. Bei Bildern mit höherer Auflösung ist die erforderliche minimale Gesichtsgröße größer.

> **Note**
>
> Die vorangegangene Empfehlung basiert auf der nativen Auflösung der Kamera. Wenn aus einem Bild geringer Auflösung ein Bild hoher Auflösung generiert wird, werden (aufgrund der beim Upsampling des Bildes entstehenden Artefakte) nicht die für eine Gesichtssuche benötigten Ergebnisse erzielt.

- Kamerawinkel: Es gibt drei Metriken für den Kamerawinkel: Neigungswinkel, Rollwinkel und Gierwinkel.
  - Neigung: Wir empfehlen eine Neigung von weniger als 30 Grad, wenn die Kamera nach unten zeigt, und weniger als 45 Grad, wenn die Kamera nach oben zeigt.
  - Rolle: Für diesen Parameter gibt es keine Mindestanforderung. Amazon Rekognition kann jede beliebige Menge Rollen verarbeiten.
  - Gieren: Wir empfehlen ein Gieren von weniger als 45 Grad in beide Richtungen.

  Der Gesichtswinkel entlang einer beliebigen Achse, der von der Kamera aufgenommen wird, ist eine Kombination daraus, in welchem Winkel die Kamera auf die Szene gerichtet wird und in welchem Winkel die aufgenommene Person in der Szene den Kopf hält. Beispiel: Wenn die Kamera in einem Winkel von 30 Grad nach unten zeigt und die Person den Kopf in einem Winkel von weiteren 30 Grad nach unten hält, beträgt die tatsächliche Neigung des Gesichts aus der Sicht der Kamera 60 Grad. In diesem Fall könnte Amazon Rekognition das Gesicht nicht erkennen. Es wird empfohlen, beim Einrichten der Kamera die Kamerawinkel so einzustellen, dass davon auszugehen ist, dass Personen gewöhnlich mit einer Gesamtneigung (Kombination von Gesicht und Kamera) von maximal 30 Grad in die Kamera schauen.
- Kamerazoom: Die empfohlene minimale Gesichtsauflösung von 50 x 50 Pixeln sollte diese Kameraeinstellung beeinflussen. Es wird empfohlen, die Zoomeinstellung einer Kamera so zu nutzen, dass sich für die gewünschten Gesichter eine Auflösung von mindestens 50 x 50 Pixeln ergibt.
- Kamerahöhe: Für diesen Parameter sollte der empfohlene Kameraabstand maßgeblich sein.

Empfehlungen für die Kameraeinrichtung (Gespeicherte Bilder und Videostreaming)

Die folgenden Empfehlungen gelten zusätzlich zu Empfehlungen für die Kameraeinrichtung (Bild und Video).

- Der Codec sollte in H.264 kodiert sein.
- Die empfohlene Bildrate liegt bei 30 fps. (Sie sollte nicht kleiner als 5 fps sein.)
- Die empfohlene Encoder-Bitrate sollte 3 Mbit/s betragen. (Sie sollte nicht kleiner als 1,5 Mbit/s sein.)
- Bildrate im Vergleich zur Bildauflösung: Wenn die Encoder-Bitrate eine Einschränkung darstellt, empfehlen wir, eine höhere Bildauflösung einer höheren Bildrate vorzuziehen, um bessere Ergebnisse bei der Gesichtssuche zu erzielen. Auf diese Weise wird sichergestellt, dass Amazon Rekognition ein Bild mit der bestmöglichen Qualität innerhalb der zugewiesenen Bitrate erhält. Einen Nachteil hat dies jedoch. Aufgrund der niedrigen Bildrate kann die Kamera schnelle Bewegungen in einer Szene nicht erfassen. Es ist wichtig, die Kompromisse zwischen diesen beiden Parametern bei einer bestimmten Einrichtung zu verstehen. Beispiel: Wenn die maximal mögliche Bitrate 1,5 Mbit/s beträgt, sind Aufnahmen mit der Kamera von 1080p bei 5 fps oder 720p bei 15 fps möglich. Die Wahl zwischen den beiden möglichen Einstellungen ist anwendungsabhängig, solange die empfohlene Gesichtsauflösung von 50 x 50 Pixeln erzielt wird.

Empfehlungen für die Kameraeinrichtung (Videostreaming)

Die folgende Empfehlung gilt zusätzlich zu Empfehlungen für die Kameraeinrichtung (Gespeicherte Bilder und Videostreaming).

Eine zusätzliche Einschränkung bei Streaming-Anwendungen ist die Internetbandbreite. Für Live-Video akzeptiert Amazon Rekognition nur Amazon Kinesis Video Streams als Eingabe. Sie sollten sich über die Abhängigkeit zwischen der Encoder-Bitrate und der verfügbaren Netzwerkbandbreite im Klaren sein. Die verfügbare Bandbreite sollte mindestens die Bitrate unterstützen, die von der Kamera zur Kodierung des Live-Streams verwendet wird. Auf diese Weise wird sichergestellt, dass alle Kameraaufnahmen über Amazon Kinesis Video Streams weitergeleitet werden. Wenn die verfügbare Bandbreite kleiner als die Encoder-Bitrate ist, gehen Amazon Kinesis Video Streams je nach Netzwerkbandbreite Bits verloren. Dies führt zu niedrigerer Videoqualität.

Eine typische Streaming-Einrichtung umfasst den Anschluss mehrerer Kameras an einen Netzwerk-Hub, der die Datenströme weiterleitet. In diesem Fall sollte die Bandbreite groß genug für die Gesamtsumme der Datenströme von allen am Hub angeschlossenen Kameras sein. Beispiel: Wenn der Hub mit fünf Kameras mit einer Kodierung von 1,5 Mbit/s verbunden ist, sollte die verfügbare Netzwerkbandbreite mindestens 7,5 Mbit/s betragen. Um sicherzustellen, dass keine Pakete verloren gehen, sollten Sie versuchen, eine Netzwerkbandbreite höher als 7,5 Mbit/s aufrechtzuerhalten, um Jitter aufgrund von Verbindungsunterbrechungen zwischen einer Kamera und dem Hub bewältigen zu können. Der tatsächliche Wert ist von der Zuverlässigkeit des internen Netzwerks abhängig.

Empfehlungen für die Verwendung von Face Liveness

Wir empfehlen Ihnen, bei der Verwendung von Rekognition Face Liveness die folgenden bewährten Methoden anzuwenden:

- Benutzer sollten den Face Liveness Check in Umgebungen durchführen, die nicht zu dunkel oder zu hell sind und eine ziemlich gleichmäßige Beleuchtung bieten.
- Benutzer sollten bei der Überprüfung in Webbrowsern die Helligkeit ihres Bildschirms auf die maximale Helligkeit erhöhen. Mobile Native SDKs passt die Displayhelligkeit automatisch an.
- Wählen Sie einen Schwellenwert für den Zuverlässigkeitswert, der der Art Ihres Anwendungsfalls entspricht. Verwenden Sie für Anwendungsfälle mit größeren Sicherheitsbedenken einen hohen Schwellenwert.
- Überprüfen Sie die Auditbilder regelmäßig von Mitarbeitern, um sicherzustellen, dass gefälschte Angriffe bei Erreichen der von Ihnen festgelegten Vertrauensschwelle abgewehrt werden.
- Bieten Sie Ihren Benutzern einen alternativen Weg zur Verifizierung der Gesichtserkennung an, wenn sie fotoempfindlich sind oder ihre Gesichtserkennung nicht mit Rekognition verifizieren möchten.
- Senden Sie das Ergebnis der Echtheitsprüfung nicht und zeigen Sie es auch nicht in der Benutzeranwendung an. Senden Sie nur ein Bestanden- oder Fehlgeschlagen-Signal.
- Lassen Sie von einem einzigen Gerät aus nur fünf fehlgeschlagene Verfügbarkeitsprüfungen innerhalb von drei Minuten zu. Nach fünf Fehlschlägen tritt für den Benutzer ein Timeout von 30—60 Minuten ein. Wenn das Muster drei- bis fünfmal wiederholt auftritt, blockieren Sie das Benutzergerät daran, weitere Aufrufe zu tätigen.
- Implementieren Sie den Startbildschirm in Ihren Workflow, damit Benutzer die Face-Liveness-Kontrollen leichter bestehen können.

- Sie sind dafür verantwortlich, Ihren Endnutzern rechtlich angemessene Datenschutzhinweise für die Verarbeitung, Speicherung, Nutzung und Übertragung von Inhalten durch Face Liveness bereitzustellen und die erforderliche Zustimmung von ihnen einzuholen.

Erkennung von Objekten und Konzepten

Dieser Abschnitt enthält Informationen zur Erkennung von Labels in Bildern und Videos mit Amazon Rekognition Image und Amazon Rekognition Video.

Ein Label oder ein Tag ist ein Objekt oder ein Konzept (einschließlich Szenen und Aktionen), das in einem Bild oder Video anhand seines Inhalts gefunden wird. Ein Bild von Menschen an einem tropischen Strand kann zum Beispiel Etiketten wie Palme (Objekt), Strand (Szene), Laufen (Aktion) und Draußen (Konzept) enthalten.

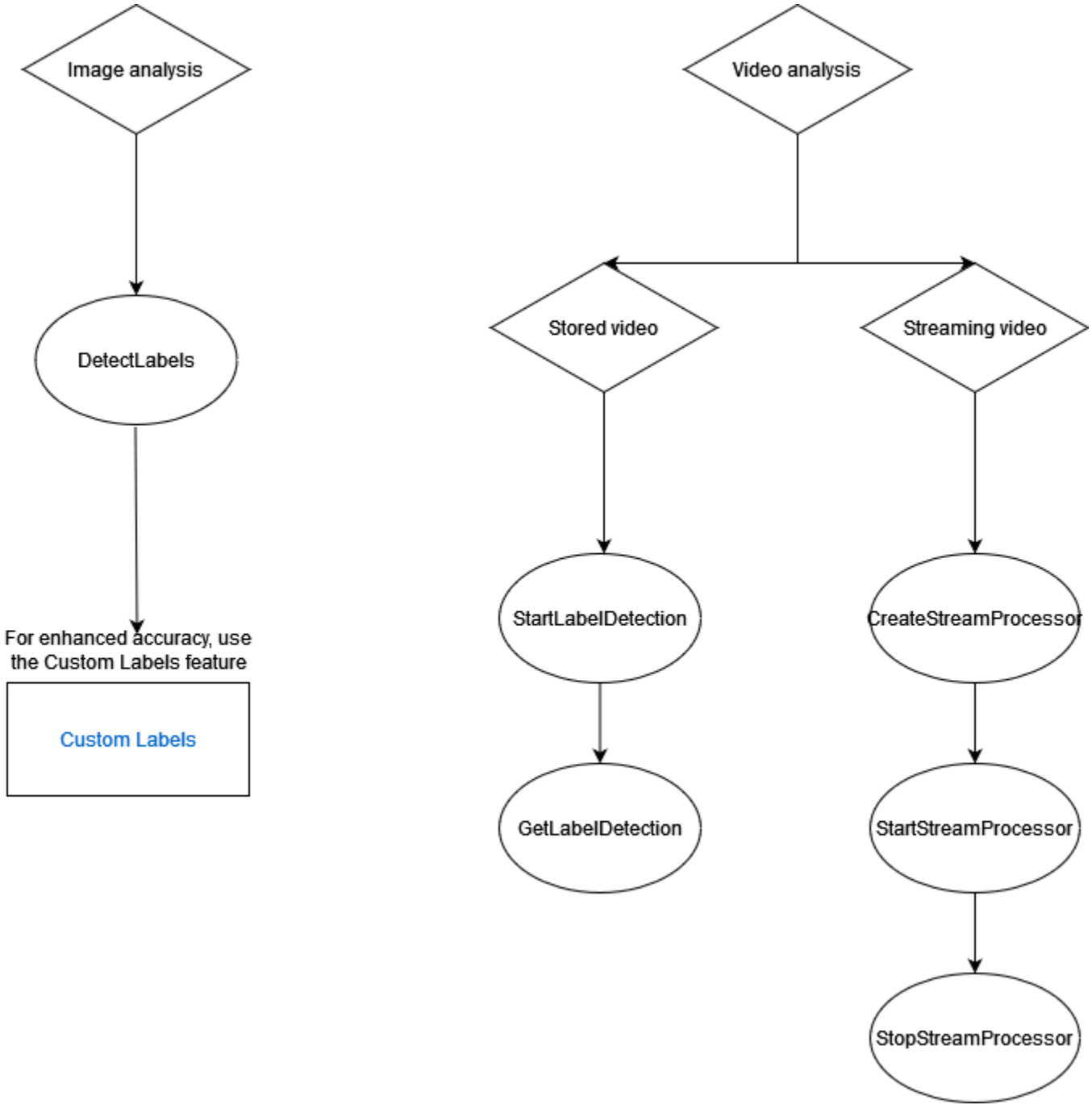
Etiketten, die von Rekognition-Etikettenerkennungsoperationen unterstützt werden

- Um die neueste Liste der von Amazon Rekognition unterstützten Labels und Objektbegrenzungsrahmen herunterzuladen, klicken Sie hier.
- Um die vorherige Liste der Labels und Objektbegrenzungsrahmen herunterzuladen, klicken Sie hier.

> **Note**
>
> Amazon Rekognition macht binäre geschlechtsspezifische Vorhersagen (Mann, Frau, Mädchen usw.), die auf dem physischen Erscheinungsbild einer Person in einem bestimmten Bild basieren. Diese Art von Vorhersage dient nicht dazu, die Geschlechtsidentität einer Person zu kategorisieren, und Sie sollten Amazon Rekognition nicht verwenden, um eine solche Entscheidung zu treffen. Beispielsweise könnte ein männlicher Schauspieler, der für eine Rolle eine langhaarige Perücke und Ohrringe trägt, als weiblich eingestuft werden.
> Die Verwendung von Amazon Rekognition für binäre geschlechtsspezifische Vorhersagen eignet sich am besten für Anwendungsfälle, in denen aggregierte Statistiken zur Geschlechterverteilung analysiert werden müssen, ohne bestimmte Benutzer zu identifizieren. Zum Beispiel der Prozentsatz weiblicher Benutzer im Vergleich zu männlichen Benutzern auf einer Social-Media-Plattform.
> Wir raten davon ab, anhand von binären Geschlechtervoraussagen Entscheidungen zu treffen, durch sich auf die Rechte, Datenschutz oder Zugriff auf Services von Einzelpersonen auswirken.

Amazon Rekognition sendet Label in englischer Sprache zurück. Sie können Amazon Translate verwenden, um englische Label in andere Sprachen zu übersetzen.

Das folgende Diagramm zeigt die Reihenfolge der Anrufvorgänge, abhängig von Ihren Zielen für die Nutzung der Amazon Rekognition Image- oder Amazon Rekognition Video Video-Operationen:



Label-Antwortobjekte

Begrenzungsrahmen

Amazon Rekognition Image und Amazon Rekognition Video können die Begrenzungsrahmen für gängige Objektlabels wie z. B. Personen, Fahrzeuge, Möbel, Bekleidung oder Haustiere zurückgeben. Für weniger gängige Objektlabels werden keine Informationen zum Begrenzungsrahmen zurückgegeben. Sie können die Begrenzungsrahmen verwenden, um die genaue Position von Objekten in einem Bild zu finden, Instanzen erkannter Objekte zu zählen oder die Größe eines Objekts mit Begrenzungsrahmen-Dimensionen zu messen.

Zum Beispiel kann Amazon Rekognition Image im folgenden Bild die Anwesenheit einer Person, eines Skateboards, geparkter Autos und andere Informationen erkennen. Amazon Rekognition Image gibt auch den Begrenzungsrahmen für eine erkannte Person und andere erkannte Objekte wie Autos und Räder zurück.

Zuverlässigkeitswert

Amazon Rekognition Video und Amazon Rekognition Image geben in Prozent an, wie viel Zuverlässigkeit Amazon Rekognition in die Richtigkeit jedes erkannten Labels hat.

Übergeordnete Kategorien

Amazon Rekognition Image und Amazon Rekognition Video verwenden eine hierarchische Taxonomie von Ahnen-Labels, um Label zu kategorisieren. Beispiel: Eine Person, die eine Straße überquert, könnte als Fußgänger erkannt werden. Das übergeordnete Label für Fußgänger ist Person. Beide diese Labels werden in der Antwort zurückgegeben. Alle Vorgängerlabels werden zurückgegeben. Jedes Label enthält eine Liste der übergeordneten Labels und andere Vorgängerlabels. Beispiel: Großeltern- und Urgroßeltern-Labels, sofern vorhanden. Sie können die übergeordneten Labels zum Erstellen von Gruppen zusammengehöriger Labels verwenden, und um Abfragen ähnlicher Labels in einem oder mehreren Bildern zuzulassen. Beispiel: Eine Abfrage für alle Fahrzeuge könnte ein Auto von einem Bild und ein Motorrad von einem anderen zurückgeben.

Kategorien

Amazon Rekognition Image und Amazon Rekognition Video geben Informationen zu Labelkategorien zurück. Label sind Teil von Kategorien, in denen einzelne Label auf der Grundlage gemeinsamer Funktionen und Kontexte gruppiert werden, z. B. „Fahrzeuge und Autos“ und „Lebensmittel und Getränke“. Eine Labelkategorie kann eine Unterkategorie einer übergeordneten Kategorie sein.

Aliasnamen

Amazon Rekognition Image und Amazon Rekognition Video senden nicht nur Label sondern geben auch alle Aliase zurück, die mit dem Label verknüpft sind. Aliase sind Label mit derselben Bedeutung oder Label, die visuell mit dem zurückgesandten Hauptlabel austauschbar sind. Beispielsweise ist „Handy“ ein Alias für „Mobiltelefon“.

In früheren Versionen gab Amazon Rekognition Image Aliase wie „Handy“ in derselben Liste von Hauptlabelnamen zurück, die „Mobiltelefon“ enthielten. Amazon Rekognition Image gibt jetzt „Handy“ in einem Feld mit dem Namen „Aliase“ und „Mobiltelefon“ in der Liste der Hauptlabelnamen zurück. Wenn Ihre Anwendung auf den Strukturen basiert, die von einer früheren Version von Rekognition zurückgegeben wurden, müssen Sie möglicherweise die aktuelle Antwort, die von den Bild- oder Videolabelerkennungsoperationen zurückgegeben wurde, in die vorherige Antwortstruktur umwandeln, in der alle Label und Aliase als Hauptlabel zurückgegeben werden.

Wenn Sie die aktuelle Antwort von der DetectLabels API (für die Erkennung von Etiketten in Bildern) in die vorherige Antwortstruktur umwandeln müssen, finden Sie das Codebeispiel unter. Transformation der DetectLabels Antwort

Wenn Sie die aktuelle Antwort von der GetLabelDetection API (zur Erkennung von Bezeichnungen in gespeicherten Videos) in die vorherige Antwortstruktur umwandeln müssen, finden Sie das Codebeispiel unterTransformation der Antwort GetLabelDetection .

Bildeigenschaften

Amazon Rekognition Image gibt Informationen zur Bildqualität (Schärfe, Helligkeit und Kontrast) für das gesamte Bild zurück. Schärfe und Helligkeit werden auch für den Vordergrund und den Hintergrund des Bildes zurückgegeben. Bildeigenschaften können auch verwendet werden, um dominante Farben des gesamten Bildes, des Vordergrunds, des Hintergrunds und von Objekten mit Begrenzungsrahmen zu erkennen.

Das Folgende ist ein Beispiel für die ImageProperties Daten, die in der Antwort auf eine DetectLabels Operation für das nachfolgende Bild enthalten sind:

Image Properties	Dominant Colors Examples and Pixel Percentage	Image Quality
Entire Image	Hex code #808080, RGB (128, 128, 128), 15.72 Hex code #000000, RGB (0, 0, 0), 15.10 Hex code #696969, RGB (105, 105, 105), 14.02 Hex code #8fbc8f, RGB (143, 188, 143), 12.70 Hex code #5f9ea0, RGB (95, 158, 160), 11.92	Brightness: 76.08 Sharpness: 89.72 Contrast: 88.42
Foreground	Hex code #8fbc8f, RGB (143, 188, 143), 30.18 Hex code #5f9ea0, RGB (95, 158, 160), 24.29 Hex code #000000, RGB (0, 0, 0), 12.02 Hex code #2f4f4f, RGB (47, 79, 79), 9.20 Hex code #696969, RGB (105, 105, 105), 8.95	Brightness: 79.48 Sharpness: 93.47
Background	Hex code #808080, RGB (128, 128, 128), 21.16 Hex code #2f4f4f, RGB (47, 79, 79), 14.61 Hex code #000000, RGB (0, 0, 0), 14.23 Hex code #696969, RGB (105, 105, 105), 13.19 Hex code #ffebcd, RGB (255, 235, 205), 12.80	Brightness: 74.42 Sharpness: 87.84
Car (example of objects with bounding boxes)	Hex code #5f9ea0, RGB (95, 158, 160), 29.18 Hex code #8fbc8f, RGB (143, 188, 143), 14.39 Hex code #000000, RGB (0, 0, 0), 11.76 Hex code #808080, RGB (128, 128, 128), 11.38 Hex code #2f4f4f, RGB (47, 79, 79), 9.44	Not applicable

Bildeigenschaften sind für Amazon Rekognition Video nicht verfügbar.

Modellversion

Amazon Rekognition Image und Amazon Rekognition Video geben beide die Version des Label-Erkennungsmodells zurück, mit dem Labels in einem Bild oder einem gespeicherten Video erkannt wurden.

Einschluss- und Ausschlussfilter

Sie können die Ergebnisse filtern, die von Amazon-Rekognition-Image- und Amazon-Rekognition-Video-Labelerkennungsoperationen zurückgegeben werden. Filtern Sie die Ergebnisse, indem Sie Filterkriterien für Label und Kategorien angeben. Labelfilter können inklusiv oder exklusiv sein.

Weitere Informationen zur Filterung der mit `DetectLabels` erzielten Ergebnisse finden Sie unter Erkennen von Labels in einem Bild.

Weitere Informationen zur Filterung der durch `GetLabelDetection` erzielten Ergebnisse finden Sie unter Erkennen von Labels in einem Video.

Ergebnisse sortieren und aggregieren

Die Ergebnisse bestimmter Amazon-Rekognition-Video-Operationen können nach Zeitstempeln und Videosegmenten sortiert und aggregiert werden. Beim Abrufen der Ergebnisse eines Auftrags zur Labelerkennung oder Inhaltsmoderation können Sie mit `GetLabelDetection` bzw. `GetContentModeration` die `SortBy`- und `AggregateBy`-Argumente verwenden, um anzugeben, wie Ihre Ergebnisse zurückgegeben werden sollen. Sie können `SortBy` mit `TIMESTAMP` oder `NAME` (Labelnamen) und `TIMESTAMPS` oder `SEGMENTS` mit dem AggregateBy Argument verwenden.

Erkennen von Labels in einem Bild

Sie können die DetectLabelsOperation verwenden, um Beschriftungen (Objekte und Konzepte) in einem Bild zu erkennen und Informationen über die Eigenschaften eines Bilds abzurufen. Zu den Bildeigenschaften gehören Attribute wie die Farbe des Vordergrunds und des Hintergrunds sowie die Schärfe, Helligkeit und der Kontrast des Bildes. Sie können nur die Labels in einem Bild, nur die Eigenschaften des Bildes oder beides abrufen. Ein Beispiel finden Sie unter Analysieren von Bildern, die in einem Amazon-S3-Bucket gespeichert sind.

In den folgenden Beispielen werden die AWS SDKs Befehle various und AWS CLI to verwendet`DetectLabels`. Informationen über die Antwort auf die Operation `DetectLabels` finden Sie unter DetectLabels Antwort.

So erkennen Sie Labels in einem Bild

1. Wenn Sie dies noch nicht getan haben:

a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`- und `AmazonS3ReadOnlyAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Laden Sie ein Bild mit einem oder mehreren Objekten wie Bäumen, Häusern und einem Boot auf Ihren S3-Bucket hoch. Das Bild muss entweder im JPG- oder PNG-Format vorliegen.

   Weitere Anleitungen finden Sie unter Upload eines Objekts in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service.

3. Verwenden Sie die folgenden Beispiele zum Aufrufen der `DetectLabels`-Operation.

   Java

   Dieses Beispiel zeigt eine Liste von Labels an, die auf dem Eingabebild erkannt wurden. Ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, das Sie in Schritt 2 verwendet haben.

```
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

Dieses Beispiel zeigt die JSON-Ausgabe von der `detect-labels`-CLI-Operation an. Ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, das Sie in Schritt 2 verwendet haben. Ersetzen Sie den Wert von `profile-name` mit dem Namen Ihres Entwicklerprofils.

```
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
```

Wenn Sie auf einem Windows-Gerät auf die CLI zugreifen, verwenden Sie doppelte Anführungszeichen anstelle von einfachen Anführungszeichen und maskieren Sie die inneren doppelten Anführungszeichen durch einen Backslash (d. h. \), um eventuell auftretende Parserfehler zu beheben. Sehen Sie sich zum Beispiel Folgendes an:

```
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

Dieses Beispiel zeigt die Labels an, die auf dem Eingabebild erkannt wurden. Ersetzen Sie in der `main`-Funktion die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bildes, die Sie in Schritt 2 verwendet haben. Ersetzen Sie den Wert von

`profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

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

import boto3

def detect_labels(photo, bucket):

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

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

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

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

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

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

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

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

        return len(response['Labels'])

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

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

.NET

Dieses Beispiel zeigt eine Liste von Labels an, die auf dem Eingabebild erkannt wurden. Ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, das Sie in Schritt 2 verwendet haben.

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

using System;
```

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

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

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

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

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

Ruby

Dieses Beispiel zeigt eine Liste von Labels an, die auf dem Eingabebild erkannt wurden. Ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, das Sie in Schritt 2 verwendet haben.

```
 # 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

Dieses Beispiel zeigt eine Liste von Labels an, die auf dem Eingabebild erkannt wurden. Ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, das Sie in Schritt 2 verwendet haben. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

Wenn Sie TypeScript Definitionen verwenden, müssen Sie möglicherweise `import AWS from 'aws-sdk'` anstelle von`const AWS = require('aws-sdk')`, verwenden, um das Programm mit Node.js auszuführen. Weitere Informationen finden Sie im AWS - SDK für Javascript. Je nachdem, wie Sie Ihre Konfigurationen eingerichtet haben, müssen Sie möglicherweise auch Ihre Region mit `AWS.config.update({region:`*`region`*`});` angeben.

```
// 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
//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 Operationsanfrage

Die Eingabe in `DetectLabel` ist ein Bild. In dieser Beispiel-JSON-Eingabe wird das Quellbild aus einem Amazon-S3-Bucket geladen. `MaxLabels` gibt die maximale Anzahl der Labels an, die zurückzugeben sind. `MinConfidence` ist das minimale Vertrauen, das Amazon Rekognition Image in die Genauigkeit des erkannten Labels haben muss, damit es in der Antwort zurückgegeben wird.

Mit Eigenschaften können Sie ein oder mehrere Eigenschaften des Bilds angeben, die zurückgegeben werden sollen, sodass Sie `GENERAL_LABELS` und `IMAGE_PROPERTIES` auswählen können. Der Einschluss von `GENERAL_LABELS` wird die im Eingabebild erkannten Label zurückgeben, während Sie mit dem Einschluss von `IMAGE_PROPERTIES` auf Farbe und Qualität des Bildes zugreifen können.

Mit den Einstellungen können Sie die zurückgegebenen Artikel sowohl nach den `GENERAL_LABELS`- als auch nach den `IMAGE_PROPERTIES`-Merkmalen filtern. Für Label können Sie inklusive und exklusive Filter verwenden. Sie können auch nach labelspezifischen, individuellen Labels oder Nach-Label-Kategorien filtern:

- LabelInclusionFilters - Ermöglicht es Ihnen, anzugeben, welche Labels in die Antwort aufgenommen werden sollen.
- LabelExclusionFilters - Ermöglicht es Ihnen, anzugeben, welche Labels aus der Antwort ausgeschlossen werden sollen.
- LabelCategoryInclusionFilters - Ermöglicht es Ihnen, anzugeben, welche Labelkategorien in die Antwort aufgenommen werden sollen.
- LabelCategoryExclusionFilters - Ermöglicht es Ihnen, anzugeben, welche Labelkategorien aus der Antwort ausgeschlossen werden sollen.

Sie können je nach Bedarf auch inklusive und exklusive Filter kombinieren, wobei Sie einige Label oder Kategorien ausschließen und andere einbeziehen.

`IMAGE_PROPERTIES` beziehen sich auf die dominierenden Farben und Qualitätsmerkmale eines Bildes wie Schärfe, Helligkeit und Kontrast. Bei der Erkennung von `IMAGE_PROPERTIES` können Sie mithilfe des `MaxDominantColors`-Parameters die maximale Anzahl dominanter Farben angeben, die zurückgegeben werden sollen (Standard ist 10).

```
{
    "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 Antwort

Die Antwort von `DetectLabels` ist ein Array der erkannten Labels in dem Bild und der Zuverlässigkeitswert, mit dem sie erkannt wurden.

Das folgende Beispiel ist eine Antwort von `DetectLabels`. Die folgende Beispielantwort enthält eine Vielzahl von Attributen, die für GENERAL_Label zurückgegeben wurden, darunter:

- Name: Der Name des erkannten Labels. In diesem Beispiel wurde bei der Operation ein Objekt mit dem Label „Handy“ erkannt.
- Zuverlässigkeit: Jedem Label ist ein Zuverlässigkeitswert, d. h. ein Zuverlässigkeitswert, zugeordnet. In diesem Beispiel lag die Zuverlässigkeit für das Label bei 99,36 %.
- Übergeordnete Kategorien: Die Vorfahren-Labels für ein erkanntes Label. In diesem Beispiel hat das Label „Handy“ ein übergeordnetes Label namens „Telefon“.
- Aliase: Informationen über mögliche Aliase für das Label. In diesem Beispiel hat das Label „Mobiltelefon“ das mögliche Alias „Handy“.
- Kategorien: Die Labelkategorie, zu der das erkannte Label gehört. In diesem Beispiel handelt es sich um Technologie und Informatik.

Die Antwort für gängige Objektlabels enthält Informationen zum Begrenzungsrahmen für die Position des Labels auf dem Eingabebild. Beispiel: Das Label „Person“ ist ein Instances-Array mit zwei Begrenzungsrahmen. Dies sind die Positionen von zwei Personen, die im Bild erkannt wurden.

Die Antwort enthält auch Attribute zu IMAGE_PROPERTIES. Die Funktion IMAGE_PROPERTIES bietet folgende Attribute:

- Qualität: Informationen über Schärfe, Helligkeit und Kontrast des Eingabebilds, bewertet zwischen 0 und 100. Die Qualität wird für das gesamte Bild sowie für den Hintergrund und Vordergrund des Bildes angegeben, sofern verfügbar. Der Kontrast wird jedoch nur für das gesamte Bild gemeldet, während Schärfe und Helligkeit auch für Hintergrund und Vordergrund gemeldet werden.
- Dominante Farbe: Eine Reihe der dominanten Farben im Bild. Jede dominante Farbe wird mit einem vereinfachten Farbnamen, einer CSS-Farbpalette, RGB-Werten und einem Hex-Code beschrieben.
- Vordergrund: Informationen über die dominanten Farben, Schärfe und Helligkeit des Vordergrunds des Eingabebilds.
- Hintergrund: Informationen über die dominierenden Farben, Schärfe und Helligkeit des Hintergrunds des Eingabebilds.

Wenn GENERAL_LABELS und IMAGE_PROPERTIES zusammen als Eingabeparameter verwendet werden, gibt Amazon Rekognition Image auch die dominanten Farben von Objekten mit Begrenzungsrahmen zurück.

Das Feld `LabelModelVersion` enthält die Versionsnummer des von `DetectLabels` verwendeten Erkennungsmodells.

```
{

   "Labels": [
        {
            "Name": "Mobile Phone",
            "Parents": [
              {
                "Name": "Phone"
              }
            ],
            "Aliases": [
              {
                "Name": "Cell Phone"
```

```
            }
          ],
          "Categories": [
            {
              "Name": "Technology and Computing"
            }
          ],
          "Confidence": 99.9364013671875,
          "Instances": [
              {
                  "BoundingBox": {
                      "Width": 0.26779675483703613,
                      "Height": 0.8562285900115967,
                      "Left": 0.3604024350643158,
                      "Top": 0.09245597571134567,
                  }
                  "Confidence": 99.9364013671875,
                  "DominantColors": [
                  {
              "Red": 120,
              "Green": 137,
              "Blue": 132,
              "HexCode": "3A7432",
              "SimplifiedColor": "red",
              "CssColor": "fuscia",
              "PixelPercentage": 40.10
                  }
                      ],
              }
          ]
      }
  ],
  "ImageProperties": {
      "Quality": {
          "Brightness": 40,
          "Sharpness": 40,
          "Contrast": 24,
      },
      "DominantColors": [
          {
              "Red": 120,
              "Green": 137,
              "Blue": 132,
              "HexCode": "3A7432",
```

```
                "SimplifiedColor": "red",
                "CssColor": "fuscia",
                "PixelPercentage": 40.10
            }
        ],
        "Foreground": {
            "Quality": {
                "Brightness": 40,
                "Sharpness": 40,
            },
            "DominantColors": [
                {
                    "Red": 200,
                    "Green": 137,
                    "Blue": 132,
                    "HexCode": "3A7432",
                    "CSSColor": "",
                    "SimplifiedColor": "red",
                    "PixelPercentage": 30.70
                }
            ],
        }
        "Background": {
            "Quality": {
                "Brightness": 40,
                "Sharpness": 40,
            },
            "DominantColors": [
                {
                    "Red": 200,
                    "Green": 137,
                    "Blue": 132,
                    "HexCode": "3A7432",
                    "CSSColor": "",
                    "SimplifiedColor": "Red",
                    "PixelPercentage": 10.20
                }
            ],
        },
    },
    "LabelModelVersion": "3.0"
}
```

Transformation der DetectLabels Antwort

Wenn Sie die DetectLabels API verwenden, muss die Antwortstruktur möglicherweise die ältere API-Antwortstruktur nachahmen, bei der sowohl primäre Labels als auch Aliase in derselben Liste enthalten waren.

Im Folgenden finden Sie ein Beispiel für die aktuelle API-Antwort von: DetectLabels

```
"Labels": [
        {
            "Name": "Mobile Phone",
            "Confidence": 99.99717712402344,
            "Instances": [],
            "Parents": [
                {
                "Name": "Phone"
                }
             ],
            "Aliases": [
                {
                "Name": "Cell Phone"
                }
             ]
        }
 ]
```

Das folgende Beispiel zeigt die vorherige Antwort der DetectLabelsAPI:

```
"Labels": [
        {
            "Name": "Mobile Phone",
            "Confidence": 99.99717712402344,
            "Instances": [],
            "Parents": [
                {
                "Name": "Phone"
                }
             ]
         },
         {
            "Name": "Cell Phone",
            "Confidence": 99.99717712402344,
```

```
            "Instances": [],
            "Parents": [
                {
                "Name": "Phone"
                }
             ]
        },
]
```

Bei Bedarf können Sie die aktuelle Antwort so transformieren, dass sie dem Format der älteren Antwort folgt. Sie können den folgenden Beispielcode verwenden, um die neueste API-Antwort in die vorherige API-Antwortstruktur umzuwandeln:

Python

Das folgende Codebeispiel zeigt, wie die aktuelle Antwort von der DetectLabels API transformiert wird. Im folgenden Codebeispiel können Sie den Wert von *EXAMPLE_INFERENCE_OUTPUT* durch die Ausgabe einer DetectLabels Operation ersetzen, die Sie ausgeführt haben.

```
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)
```

Hier ist ein Beispiel für die transformierte Antwort:

```
#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"
            ]
        }
    ]
}
```

Erkennen von Labels in einem Video

Amazon Rekognition Video kann Label (Objekte und Konzepte) und den Zeitpunkt, zu dem ein Label erkannt wird, in einem Video erkennen. Ein SDK-Codebeispiel finden Sie unter Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK). Ein AWS CLI Beispiel finden Sie unterAnalysieren eines Videos mit dem AWS Command Line Interface.

Die Amazon-Rekognition-Video-Labelerkennung ist eine asynchrone Operation. Rufen Sie an, um die Erkennung von Etiketten in einem Video zu starten StartLabelDetection.

Das Amazon-Simple-Notification-Service-Thema, zu dem Amazon Rekognition Video die Ergebnisse der Objekterkennung und den Abschlussstatus einer Videoanalyse-Operation veröffentlicht. Wenn die Videoanalyse erfolgreich ist, rufen Sie GetLabelDetection auf, um die erkannten Labels zu erhalten. Informationen zum Aufruf von API-Operationen der Videoanalyse finden Sie unter Amazon-Rekognition-Video-Operationen aufrufen.

StartLabelDetectionAnfrage

Das folgende Beispiel ist eine Anforderung für die `StartLabelDetection`-Operation. Sie stellen für die `StartLabelDetection`-Operation ein Video bereit, das in einem Amazon-S3-Bucket gespeichert ist. In der JSON-Beispielanforderung werden der Amazon-S3-Bucket und der Videoname zusammen mit `MinConfidence`, `Features`, `Settings` und `NotificationChannel` angegeben.

`MinConfidence` ist das Mindestvertrauen, das Amazon Rekognition Video in die Genauigkeit des erkannten Labels oder eines Instance-Begrenzungsrahmens (falls erkannt) haben muss, damit es in der Antwort zurückgegeben wird.

Mit `Features` können Sie angeben, dass GENERAL_LABELS als Teil der Antwort zurückgegeben werden soll.

Mit `Settings` können Sie die zurückgegebenen Artikel nach GENERAL_LABELS filtern. Für Label können Sie inklusive und exklusive Filter verwenden. Sie können auch nach labelspezifischen, individuellen Labels oder Nach-Label-Kategorien filtern:

- `LabelInclusionFilters` – Wird verwendet, um anzugeben, welche Label in der Antwort enthalten sein sollen
- `LabelExclusionFilters` – Wird verwendet, um anzugeben, welche Label aus der Antwort ausgeschlossen werden sollen.
- `LabelCategoryInclusionFilters` – Wird verwendet, um anzugeben, welche Labelkategorien in die Antwort aufgenommen werden sollen.
- `LabelCategoryExclusionFilters` – Wird verwendet, um anzugeben, welche Labelkategorien aus der Antwort ausgeschlossen werden sollen.

Sie können je nach Bedarf auch inklusive und exklusive Filter kombinieren, wobei Sie einige Label oder Kategorien ausschließen und andere einbeziehen.

`NotificationChannel` ist der ARN des Amazon-SNS-Themas, für das Amazon Rekognition Video den Abschlussstatus des Labelerkennungsvorgangs veröffentlichen soll. Wenn Sie die `AmazonRekognitionServiceRole`-Berechtigungsrichtlinie verwenden, muss das Amazon-SNS-Thema einen Themennamen haben, der mit Rekognition beginnt.

Im Folgenden finden Sie eine `StartLabelDetection`-Beispielanforderung in JSON-Form, einschließlich Filtern:

```
{
    "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 Antwort auf die Operation

`GetLabelDetection` gibt ein Array (`Labels`) zurück, das Informationen über die erkannten Labels im Video enthält. Das Array kann entweder nach Zeit oder nach dem Label sortiert werden, das bei der Angabe des `SortBy`-Parameters erkannt wurde. Mithilfe des Parameters können Sie auch auswählen, wie Antwortelemente mit `AggregateBy` aggregiert werden.

Nachfolgend finden Sie ein Beispiel einer JSON-Antwort von `GetLabelDetection`. In der Antwort ist Folgendes zu beachten:

- Sortierreihenfolge – Das Array der zurückgegebenen Labels wird nach der Zeit sortiert. Um nach dem Label zu sortieren, geben Sie `NAME` im Eingabeparameter `SortBy` für `GetLabelDetection` an. Wenn das Label mehrfach im Video erscheint, gibt es mehrere Instanzen des Elements (LabelDetection). Die Standardsortierreihenfolge ist `TIMESTAMP`, die sekundäre Sortierreihenfolge ist `NAME`.
- Label-Informationen – Das `LabelDetection`-Arrayelement enthält ein (Label)-Objekt, das wiederum den Namen des Labels und das Vertrauen, das Amazon Rekognition in die Genauigkeit des erkannten Labels hat, enthält. Ein `Label`-Objekt enthält auch eine hierarchische Taxonomie der Labels und Begrenzungsrahmen-Informationen für gängige Labels. `Timestamp` ist die Zeit, zu der das Label erkannt wurde, definiert als die Anzahl der seit dem Start des Videos verstrichenen Millisekunden.

  Informationen zu allen Kategorien oder Aliasnamen, die einem Label zugeordnet sind, werden ebenfalls zurückgegeben. Bei nach Video `SEGMENTS` aggregierten Ergebnissen werden die `StartTimestampMillis`-, `EndTimestampMillis`- und `DurationMillis`-Strukturen zurückgegeben, die jeweils die Startzeit, die Endzeit und die Dauer eines Segments definieren.
- Aggregation: Gibt an, wie Ergebnisse aggregiert werden, wenn sie zurückgegeben werden. Standardmäßig wird nach `TIMESTAMPS` aggregiert. Sie können sich auch für die Aggregation

nach SEGMENTS entscheiden, wodurch die Ergebnisse über ein Zeitfenster aggregiert werden. Bei der Aggregation nach SEGMENTS werden Informationen über erkannte Instances mit Begrenzungsrahmen nicht zurückgegeben. Es werden nur Label zurückgegeben, die während der Segmente erkannt wurden.

- Seiteninformationen – Das Beispiel zeigt eine Seite mit Informationen der Label-Erkennung. Sie können festlegen, wie viele `LabelDetection`-Objekte zurückgegeben werden sollen, durch den Eingabeparameter `MaxResults` von `GetLabelDetection`. Wenn mehr Ergebnisse als `MaxResults` vorhanden sind, gibt `GetLabelDetection` einen Token zurück (`NextToken`), der dazu verwendet wird, die nächste Seite mit Ergebnissen zu erhalten. Weitere Informationen finden Sie unter Analyseergebnisse von Amazon Rekognition Video abrufen.
- Video-Informationen – Die Antwort enthält Informationen über das Videoformat (`VideoMetadata`) in jeder Seite mit Informationen, die von `GetLabelDetection` zurückgegeben werden.

Im Folgenden finden Sie eine GetLabelDetection Beispielantwort in JSON-Form mit Aggregation durch 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
    }
}
```

Im Folgenden finden Sie eine GetLabelDetection Beispielantwort in JSON-Form mit Aggregation nach SEGMENTEN:

```
{
    "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
    }
}
```

Transformation der Antwort GetLabelDetection

Beim Abrufen von Ergebnissen mit der GetLabelDetection API-Operation muss die Antwortstruktur möglicherweise die ältere API-Antwortstruktur nachahmen, bei der sowohl primäre Labels als auch Aliase in derselben Liste enthalten waren.

Die JSON-Beispielantwort aus dem vorherigen Abschnitt zeigt die aktuelle Form der API-Antwort von. GetLabelDetection

Das folgende Beispiel zeigt die vorherige Antwort der 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
    }
}
```

Bei Bedarf können Sie die aktuelle Antwort so transformieren, dass sie dem Format der älteren Antwort folgt. Sie können den folgenden Beispielcode verwenden, um die neueste API-Antwort in die vorherige API-Antwortstruktur umzuwandeln:

```
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
```

Erkennung von Labels in Streaming-Videoereignissen

Sie können Amazon Rekognition Video verwenden, um Label in Streaming-Videos zu erkennen. Dazu erstellen Sie einen Stream-Prozessor (CreateStreamProcessor), um die Analyse von Streaming-Videos zu starten und zu verwalten.

Amazon Rekognition Video verwendet Amazon Kinesis Video Streams, um einen Videostrom zu empfangen und zu verarbeiten. Wenn Sie den Stromprozessor erstellen, wählen Sie aus, was der Stromprozessor erkennen soll. Sie können zwischen Personen, Paketen und Haustieren oder Personen und Paketen wählen. Das Analyseergebnis wird in Ihrem Amazon-S3-Bucket und in Amazon-SNS-Benachrichtigungen ausgegeben. Beachten Sie, dass Amazon Rekognition Video die Anwesenheit einer Person im Video erkennt, aber nicht erkennt, ob es sich bei der Person um eine bestimmte Person handelt. Informationen zur Suche nach einem Gesicht aus einer Sammlung in einem Streaming-Video finden Sie unter the section called “Suchen nach Gesichtern in einer Sammlung im Streaming-Video”.

Um Amazon Rekognition Video mit Streaming-Video zu verwenden, benötigt Ihre Anwendung Folgendes:

- Ein Kinesis-Videostrom zum Senden von Streaming-Videos an Amazon Rekognition Video. Weitere Informationen Sie im Entwicklerleitfaden für Amazon Kinesis Video Streams.
- Einen Amazon-Rekognition-Video-Stromprozessor zur Verwaltung der Analyse des Streaming-Videos. Weitere Informationen finden Sie unter Überblick über den Betrieb des Amazon-Rekognition-Video-Stromprozessors.
- Ein Amazon-S3-Bucket Amazon Rekognition Video veröffentlicht Sitzungsausgaben im S3-Bucket. Die Ausgabe umfasst den Bildrahmen, in dem eine Person oder ein Objekt von Interesse zum ersten Mal erkannt wurde. Sie müssen Eigentümer des S3-Buckets sein.
- Ein Amazon SNS SNS-Thema, zu dem Amazon Rekognition Video Smart Alerts und eine end-of-session Zusammenfassung veröffentlicht.

Themen

- Einrichtung Ihrer Amazon-Rekognition-Video- und Amazon-Kinesis-Ressourcen

- Rufen Sie Operationen zur Erkennung von Bezeichnungen für Streaming-Videoereignisse auf

Einrichtung Ihrer Amazon-Rekognition-Video- und Amazon-Kinesis-Ressourcen

In den folgenden Verfahren werden die Schritte beschrieben, die Sie ergreifen, um den Kinesis-Videostrom und andere Ressourcen bereitzustellen, die zur Erkennung von Labels in einem Streaming-Video verwendet werden.

Voraussetzungen

Um dieses Verfahren ausführen zu können, AWS SDK für Java muss es installiert sein. Weitere Informationen finden Sie unter Erste Schritte mit Amazon Rekognition. Das AWS -Konto, das Sie verwenden, benötigt Zugriffsberechtigungen für die Amazon-Rekognition-API. Weitere Informationen finden Sie unter Von Amazon Rekognition definierte Aktionen im IAM-Benutzerhandbuch.

So erkennen Sie Labels in einem Videostrom (AWS-SDK)

1. Erstellen Sie einen Amazon-S3-Bucket. Notieren Sie den Bucket-Namen und alle Schlüsselpräfixe, die Sie verwenden möchten. Sie benötigen diese Informationen später.
2. Erstellen Sie ein Amazon-SNS-Thema. Sie können damit Benachrichtigungen erhalten, wenn ein Objekt von Interesse zum ersten Mal im Videostrom entdeckt wird. Notieren Sie sich den Amazon-Ressourcennamen (ARN) für das Thema. Weitere Informationen finden Sie unter Erstellen eines Amazon-SNS-Themas im Amazon-SNS-Entwicklerhandbuch.
3. Richten Sie ein Endpunkt-Abonnement für ein Amazon-SNS-Thema ein. Weitere Informationen finden Sie unter Amazon-SNS-Thema abonnieren im Amazon-SNS-Entwicklerhandbuch.
4. Erstellen Sie einen Kinesis-Videostrom und notieren Sie sich den Amazon-Ressourcennamen (ARN) des Streams.
5. Falls Sie dies noch nicht getan haben, erstellen Sie eine IAM-Servicerolle, um Amazon Rekognition Video Zugriff auf Ihre Kinesis-Videoströme, Ihren S3-Bucket und Ihr Amazon-SNS-Thema zu gewähren. Weitere Informationen finden Sie unter Zugriff für Stromprozessoren zur Labelerkennung gewähren.

Anschließend können Sie den Label-Erkennungs-Stromprozessor erstellen und den Stromprozessor mit dem von Ihnen ausgewählten Stromprozessornamen starten.

> **Note**
>
> Starten Sie den Stromprozessor erst, nachdem Sie sich vergewissert haben, dass Sie Medien in den Kinesis-Videostrom aufnehmen können.

Ausrichtung und Einrichtung der Kamera

Amazon Rekognition Video Streaming Video Events unterstützt alle Kameras, die von Kinesis Video Streams unterstützt werden. Für optimale Ergebnisse empfehlen wir, die Kamera zwischen 0 und 45 Grad über dem Boden aufzustellen. Die Kamera muss sich in ihrer kanonischen, aufrechten Position befinden. Wenn sich beispielsweise eine Person im Bild befindet, sollte die Person vertikal ausgerichtet sein und der Kopf der Person sollte höher im Bild stehen als die Füße.

Zugriff für Stromprozessoren zur Labelerkennung gewähren

Sie verwenden eine AWS Identity and Access Management (IAM) -Servicerolle, um Amazon Rekognition Video Lesezugriff auf Kinesis-Videostreams zu gewähren. Verwenden Sie dazu IAM-Rollen, um Amazon Rekognition Video Zugriff auf Ihren Amazon-S3-Bucket und auf ein Amazon-SNS-Thema zu gewähren.

Sie können eine Berechtigungsrichtlinie erstellen, die Amazon Rekognition Video den Zugriff auf ein vorhandenes Amazon-SNS-Thema, einen Amazon-S3-Bucket und einen Kinesis-Videostrom ermöglicht. Ein step-by-step Verfahren mit dem finden Sie unter. AWS CLIthe section called “AWS CLI Befehle zum Einrichten einer IAM-Rolle für die Labelerkennung”

Um Amazon Rekognition Video Zugriff auf Ressourcen für die Labelerkennung zu gewähren

1. Erstellen Sie mit dem IAM JSON Policy Editor eine neue Berechtigungsrichtlinie und verwenden Sie die folgende Richtlinie. Ersetzen Sie `kvs-stream-name` durch den Namen des Kinesis-Videostroms, `topicarn` durch den Amazon-Ressourcennamen (ARN) des Amazon-SNS-Themas, das Sie verwenden möchten, und `bucket-name` durch den Namen des Amazon-S3-Buckets.

```
{
    "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. Erstellen Sie ein IAM-Servicerolle oder aktualisieren Sie eine bestehende IAM-Servicerolle. Verwenden Sie die folgenden Informationen, um die IAM-Servicerolle zu erstellen:

   1. Wählen Sie Rekognition als Servicenamen.
   2. Wählen Sie Rekognition für den Anwendungsfall der Service-Rolle.
   3. Fügen Sie die Berechtigungsrichtlinie hinzu, die Sie in Schritt 1 erstellt haben.

3. Notieren Sie den ARN der Service-Rolle. Sie benötigen sie, um den Stromprozessor zu erstellen, bevor Sie Videoanalysevorgänge durchführen.

4. (Optional) Wenn Sie Ihren eigenen AWS KMS Schlüssel verwenden, um an Ihren S3-Bucket gesendete Daten zu verschlüsseln, müssen Sie die folgende Anweisung mit der IAM-Rolle hinzufügen. (Dies ist die IAM-Rolle, die Sie für die Schlüsselrichtlinie erstellt haben. Sie entspricht dem kundenverwalteten Schlüssel, den Sie verwenden möchten.)

```
            {
                        "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 Befehle zum Einrichten einer IAM-Rolle für die Labelerkennung

Falls Sie dies noch nicht getan haben, richten Sie das AWS CLI mit Ihren Anmeldeinformationen ein und konfigurieren Sie es.

Geben Sie die folgenden Befehle in die ein AWS CLI , um eine IAM-Rolle mit den erforderlichen Berechtigungen für die Labelerkennung einzurichten.

1. `export IAM_ROLE_NAME=labels-test-role`
2. `export AWS_REGION=us-east-1`
3. Erstellen Sie eine Richtliniendatei für Vertrauensbeziehungen (z. B.. assume-role-rekognition json) mit dem folgenden Inhalt.

```
{
  "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. Wenn der Name Ihres SNS-Themas, mit dem Sie Benachrichtigungen erhalten möchten, nicht mit dem Präfix "AmazonRekognition" beginnt, fügen Sie die folgende Richtlinie hinzu:

   `aws iam attach-role-policy --role-name $IAM_ROLE_NAME --policy-arn "arn:aws:iam::aws:policy/AmazonSNSFullAccess" --region $AWS_REGION`
7. Wenn Sie Ihren eigenen AWS-KMS-Schlüssel verwenden, um Daten zu verschlüsseln, die an Ihren Amazon-S3-Bucket gesendet werden, aktualisieren Sie die Schlüsselrichtlinie des kundenverwalteten Schlüssels, den Sie verwenden möchten.

   a. Erstellen Sie eine Datei kms_key_policy.json, die den folgenden Inhalt enthält:

```
{
"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`. Ersetzen Sie KMS_KEY_ID durch die KMS-Schlüssel-ID, die Sie erstellt haben.

c. `aws kms put-key-policy --policy-name default --key-id $KMS_KEY_ID --policy file://path-to-kms-key-policy.json`

Rufen Sie Operationen zur Erkennung von Bezeichnungen für Streaming-Videoereignisse auf

Amazon Rekognition Video kann Personen oder relevante Objekte in einem Streaming-Video erkennen und Sie benachrichtigen, wenn sie erkannt werden. Wenn Sie einen Stromprozessor zur Labelerkennung erstellen, wählen Sie aus, welche Labels Amazon Rekognition Video erkennen soll. Das können Personen, Pakete und Haustiere sein, oder Personen, Pakete und Haustiere. Wählen Sie nur die spezifischen Label aus, die Sie erkennen möchten. Auf diese Weise erstellen nur die relevanten Label Benachrichtigungen. Sie können Optionen konfigurieren, um festzulegen, wann Videoinformationen gespeichert werden sollen, und dann weitere Verarbeitungsschritte auf der Grundlage der im Frame erkannten Label durchführen.

Nachdem Sie Ihre Ressourcen eingerichtet haben, gehen Sie wie folgt vor, um Label in einem Streaming-Video zu erkennen:

1. Erstellen Sie den Stromprozessor
2. Starten Sie den Stromprozessor
3. Wenn ein Objekt von Interesse erkannt wird, erhalten Sie eine Amazon-SNS-Benachrichtigung für das erste Auftreten jedes Objekts von Interesse.
4. Der Stromprozessor stoppt, wenn die unter `MaxDurationInSeconds` angegebene Zeit abgelaufen ist.
5. Sie erhalten eine abschließende Amazon-SNS-Benachrichtigung mit einer Zusammenfassung des Ereignisses.
6. Amazon Rekognition Video veröffentlicht eine detaillierte Sitzungszusammenfassung in Ihrem S3-Bucket.

Themen

- Erstellen des Amazon-Rekognition-Video-Stromprozessors zur Labelerkennung
- Starten des Amazon-Rekognition-Video-Stromprozessors zur Labelerkennung

- Analysieren der Ergebnisse der Labelerkennung

Erstellen des Amazon-Rekognition-Video-Stromprozessors zur Labelerkennung

Bevor Sie ein Streaming-Video analysieren können, erstellen Sie einen Amazon Rekognition Video Video-Stream-Prozessor () CreateStreamProcessor.

Wenn Sie einen Stromprozessor erstellen möchten, der Labels von Interesse und Personen erkennt, geben Sie als Eingabe einen Kinesis-Videostrom (`Input`), Amazon-S3-Bucket-Informationen (`Output`) und einen Amazon-SNS-Themen-ARN (`StreamProcessorNotificationChannel`) ein. Sie können auch eine KMS-Schlüssel-ID angeben, um die an Ihren S3-Bucket gesendeten Daten zu verschlüsseln. Sie geben an, was Sie in `Settings` erkennen möchten, z. B. Personen, Pakete und Personen oder Haustiere, Personen und Pakete. Sie können auch angeben, wo im Frame Amazon Rekognition mit `RegionsOfInterest` überwachen soll. Im Folgenden sehen Sie ein JSON-Beispiel für die Anforderung von `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:nnnnnnnnnnn:MyTopic"
  },
  "RoleArn": "arn:aws:iam::nnnnnnnnn:role/Admin",
  "Settings": {
```

```
    "ConnectedHome": {
      "Labels": [
        "PET"
      ]
    "MinConfidence": 80
    }
  },
  "RegionsOfInterest": [
    {
      "BoundingBox": {
        "Top": 0.11,
        "Left": 0.22,
        "Width": 0.33,
        "Height": 0.44
      }
    },
    {
      "Polygon": [
        {
          "X": 0.11,
          "Y": 0.11
        },
        {
          "X": 0.22,
          "Y": 0.22
        },
        {
          "X": 0.33,
          "Y": 0.33
        }
      ]
    }
  ]
}
```

Beachten Sie, dass Sie den `MinConfidence`-Wert ändern können, wenn Sie den `ConnectedHomeSettings` für den Stromprozessor angeben. `MinConfidence` ist ein numerischer Wert im Bereich von 0 bis 100, der angibt, wie sicher der Algorithmus in Bezug auf seine Vorhersagen ist. Beispielsweise bedeutet eine Benachrichtigung für `person` mit einem Zuverlässigkeitswert von 90, dass der Algorithmus absolut sicher ist, dass die Person im Video

anwesend ist. Ein Zuverlässigkeitswert von 10 gibt an, dass es sich möglicherweise um eine Person handelt. Sie können `MinConfidence` auf einen gewünschten Wert Ihrer Wahl zwischen 0 und 100 festlegen, je nachdem, wie oft Sie benachrichtigt werden möchten. Wenn Sie beispielsweise nur benachrichtigt werden möchten, wenn Rekognition absolut sicher ist, dass sich ein Paket im Videoframe befindet, können Sie `MinConfidence` auf 90 setzen.

Standardmäßig ist `MinConfidence` auf „50“ gesetzt. Wenn Sie den Algorithmus für eine höhere Genauigkeit optimieren möchten, können Sie `MinConfidence` auf einen Wert über 50 einstellen. Sie erhalten dann weniger Benachrichtigungen, aber jede Benachrichtigung ist zuverlässiger. Wenn Sie den Algorithmus für einen höheren Erinnerungswert optimieren möchten, können Sie `MinConfidence` auf einen Wert unter 50 einstellen, um mehr Benachrichtigungen zu erhalten.

Starten des Amazon-Rekognition-Video-Stromprozessors zur Labelerkennung

Sie beginnen die Analyse des Streaming-Videos, indem Sie StartStreamProcessor mit dem Streaming-Prozessor-Namen aufrufen, den Sie in `CreateStreamProcessor` festgelegt haben. Wenn Sie die `StartStreamProcessor`-Operation auf einem Stromprozessor zur Labelerkennung ausführen, geben Sie Start- und Stoppinformationen ein, um die Verarbeitungszeit zu bestimmen.

Wenn Sie den Stromprozessor starten, ändert sich der Status des Label-Erkennungs-Stromprozessors auf folgende Weise:

1. Wenn Sie `StartStreamProcessor` aufrufen, wechselt der Status des Stromprozessors für die Labelerkennung von `STOPPED` oder `FAILED` zu `STARTING`.
2. Solange der Stromprozessor für die Labelerkennung läuft, bleibt er in `STARTING` aktiviert.
3. Wenn der Labelerkennungs-Stromprozessor nicht mehr läuft, wird der Status entweder `STOPPED` oder `FAILED`.

Der `StartSelector` gibt den Startpunkt im Kinesis-Stream an, an dem die Verarbeitung gestartet werden soll. Sie können den KVS-Producer-Zeitstempel oder die KVS-Fragmentnummer verwenden. Weitere Informationen finden Sie unter Fragment.

> **Note**
>
> Wenn Sie den KVS-Producer-Zeitstempel verwenden, müssen Sie die Zeit in Millisekunden eingeben.

Der `StopSelector` gibt an, wann die Verarbeitung des Streams beendet werden soll. Sie können eine Höchstdauer für die Verarbeitung des Videos angeben. Standardmäßig ist eine Höchstdauer von 10 Sekunden angegeben. Beachten Sie, dass die tatsächliche Verarbeitungszeit je nach Größe der einzelnen KVS-Fragmente etwas länger als die maximale Dauer sein kann. Wenn die maximale Dauer am Ende eines Fragments erreicht oder überschritten wurde, stoppt die Verarbeitungszeit.

Im Folgenden sehen Sie ein JSON-Beispiel für die Anforderung von `StartStreamProcessor`.

```
{
   "Name": "string",
   "StartSelector": {
     "KVSStreamStartSelector": {
         "KVSProducerTimestamp": 1655930623123
      },
        "StopSelector": {
            "MaxDurationInSeconds": 11
      }
   }
}
```

Wenn der Stromprozessor erfolgreich startet, wird eine HTTP-200-Antwort zurückgegeben. Ein leerer JSON-Text ist enthalten.

Analysieren der Ergebnisse der Labelerkennung

Es gibt drei Möglichkeiten, wie Amazon Rekognition Video Benachrichtigungen von einem Stream-Prozessor zur Etikettenerkennung veröffentlicht: Amazon SNS SNS-Benachrichtigungen für Objekterkennungsereignisse, eine Amazon SNS-Benachrichtigung für eine end-of-session Zusammenfassung und einen detaillierten Amazon S3-Bucket-Bericht.

- Amazon-SNS-Benachrichtigungen für Objekterkennungsereignisse.

  Wenn Label im Videostrom erkannt werden, erhalten Sie Amazon-SNS-Benachrichtigungen für Objekterkennungsereignisse. Amazon Rekognition veröffentlicht eine Benachrichtigung, wenn ein Objekt oder eine Person von Interesse zum ersten Mal im Videostrom entdeckt wird. Benachrichtigungen enthalten Informationen wie die Art des erkannten Labels, die Zuverlässigkeit und einen Link zum Hero-Bild. Sie enthalten auch ein zugeschnittenes Bild der erkannten Person oder des Objekts und einen Erkennungszeitstempel. Die Benachrichtigung hat das folgende Format:

```
{"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
    }
}
```

- end-of-sessionZusammenfassung von Amazon SNS.

Sie erhalten auch eine Amazon-SNS-Benachrichtigung, wenn die Stream-Verarbeitungssitzung abgeschlossen ist. In dieser Benachrichtigung sind die Metadaten für die Sitzung aufgeführt. Dazu gehören Details wie die Dauer des Streams, der verarbeitet wurde. Die Benachrichtigung hat das folgende Format:

```
{"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
    }
}
```

- Amazon-S3-Bucket-Bericht.

  Amazon Rekognition Video veröffentlicht die detaillierten Schlussfolgerungsergebnisse eines Videoanalysevorgangs in dem Amazon-S3-Bucket, der in der `CreateStreamProcessor`-Operation angegeben wird. Zu diesen Ergebnissen gehören auch Bildrahmen, in denen ein Objekt von Interesse oder eine Person zum ersten Mal erkannt wurde.

  Die Frames sind in S3 im folgenden Pfad verfügbar: ObjectKeyPrefix// StreamProcessorNameSessionId/*`service_determined_unique_path`*. In diesem Pfad LabelKeyPrefixbefindet sich ein vom Kunden bereitgestelltes optionales Argument,

StreamProcessorNamedas der Name der Stream-Prozessor-Ressource und eine eindeutige ID für die Stream-Verarbeitungssitzung SessionIdist. Ersetzen Sie diese entsprechend Ihrer Situation.

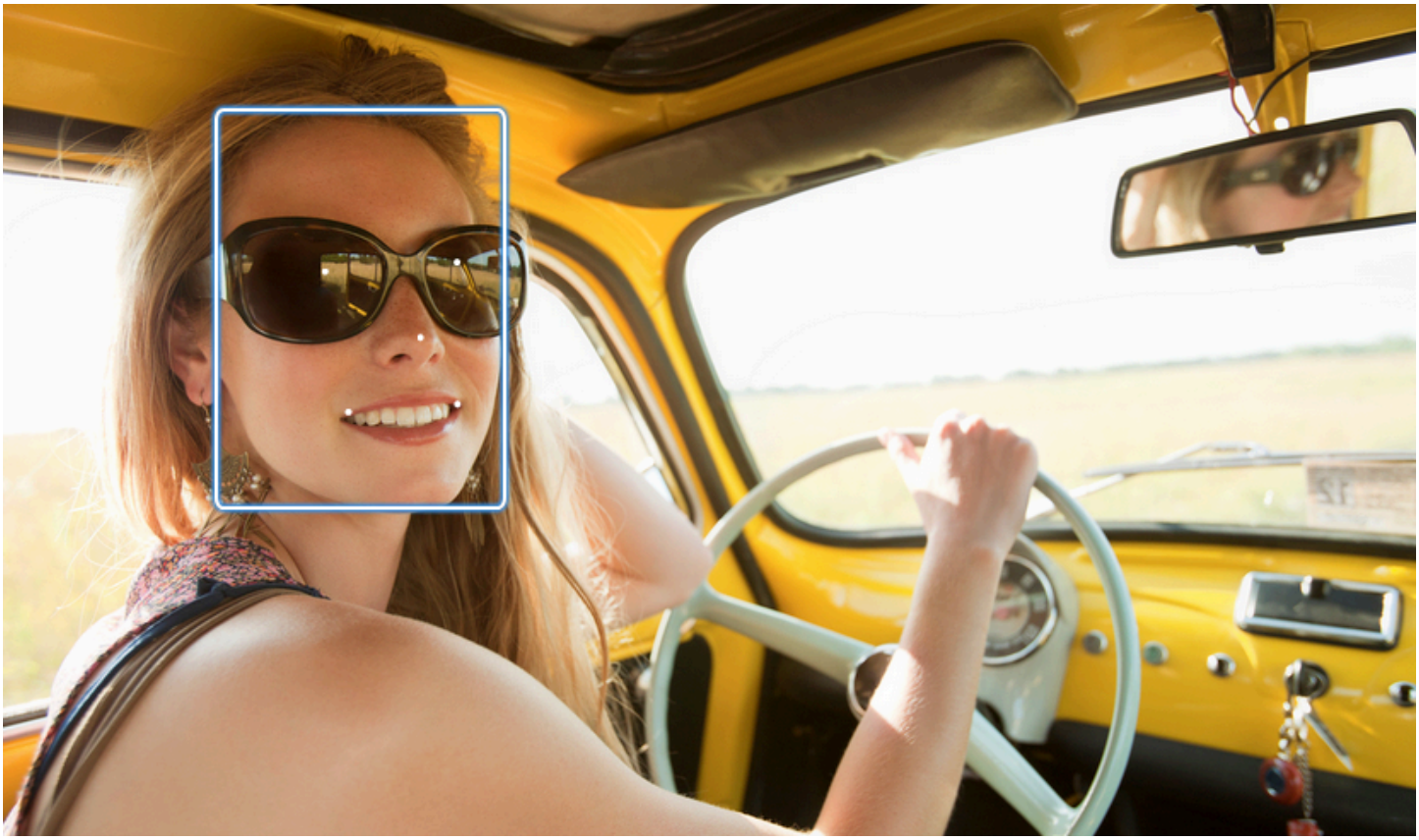
Erkennen benutzerdefinierter Label

Mit Amazon Rekognition Custom Labels können Sie Objekte und Szenen in Bildern identifizieren, die auf Ihre geschäftlichen Anforderungen zugeschnitten sind, z. B. Logos oder Maschinenteile. Weitere Informationen finden Sie unter Was ist Amazon Rekognition Custom Labels? im Entwicklerhandbuch für Amazon Rekognition Custom Labels.

Erkennung und Analyse von Gesichtern

Amazon Rekognition bietet APIs Ihnen die Möglichkeit, Gesichter in Bildern und Videos zu erkennen und zu analysieren. Dieser Abschnitt bietet einen Überblick über die Vorgänge, die bei der Gesichtsanalyse nicht gespeichert werden. Zu diesen Operationen gehören Funktionen wie das Erkennen von Wahrzeichen im Gesicht, das Analysieren von Emotionen und das Vergleichen von Gesichtern.

Amazon Rekognition kann Landmarken im Gesicht (z. B. Augenposition), Emotionen (z. B. Glück oder Traurigkeit) und andere Merkmale (z. B. Vorhandensein einer Brille, Gesichtsverdeckung) erkennen. Wenn ein Gesicht erkannt wird, analysiert das System die Gesichtsmerkmale und gibt für jedes Attribut einen Konfidenzwert zurück.



Dieser Abschnitt enthält Beispiele für Bild- und Videooperationen.

Weitere Informationen zur Verwendung der Bildoperationen von Rekognition finden Sie unter. Arbeiten mit Bildern

Weitere Informationen zur Verwendung der Videooperationen von Rekognition finden Sie unter. Arbeiten mit gespeicherten Videoanalysevorgängen

Beachten Sie, dass es sich bei diesen Vorgängen nicht um Speichervorgänge handelt. Sie können Speichervorgänge und Gesichtssammlungen verwenden, um Gesichtsmetadaten für Gesichter zu speichern, die in einem Bild erkannt wurden. Später können Sie nach gespeicherten Gesichtern in Bildern und Videos suchen. So kann z. B. in einem Video nach einer bestimmten Person gesucht werden. Weitere Informationen finden Sie unter Gesichtssuche in einer Sammlung.

Weitere Informationen finden Sie im Abschnitt Gesichter von Amazon Rekognition FAQs.

> **Note**
>
> Die von Amazon Rekognition Image und Amazon Rekognition Video verwendeten Gesichtserkennungsmodelle unterstützen nicht die Erkennung von Gesichtern in Zeichentrickfilmen/animierten Figuren oder nicht-menschlichen Wesen. Wenn Sie Zeichentrickfiguren in Bildern oder Videos erkennen möchten, empfehlen wir die Verwendung von Amazon Rekognition Custom Labels. Weitere Informationen finden Sie im Entwicklerhandbuch für Amazon Rekognition Custom Labels.

Themen

- Übersicht über Gesichtserkennung und Gesichtsvergleich
- Richtlinien zu Gesichtsattributen
- Erkennen von Gesichtern in einem Bild
- Vergleichen von Gesichtern in Bildern
- Erkennen von Gesichtern in einem gespeicherten Video

Übersicht über Gesichtserkennung und Gesichtsvergleich

Amazon Rekognition bietet Benutzern Zugriff auf zwei Hauptanwendungen für maschinelles Lernen für Bilder mit Gesichtern: Gesichtserkennung und Gesichtsvergleich. Sie ermöglichen wichtige Funktionen wie Gesichtsanalyse und Identitätsprüfung und sind daher für verschiedene Anwendungen unverzichtbar, von der Sicherheit bis hin zur Organisation persönlicher Fotos.

Gesichtserkennung

Ein Gesichtserkennungssystem befasst sich mit der Frage: „Ist auf diesem Bild ein Gesicht zu sehen?“ Zu den wichtigsten Aspekten der Gesichtserkennung gehören:

- Position und Ausrichtung: Bestimmt das Vorhandensein, die Position, den Maßstab und die Ausrichtung von Gesichtern in Bildern oder Videobildern.
- Gesichtsattribute: Erkennt Gesichter unabhängig von Attributen wie Geschlecht, Alter oder Gesichtsbehaarung.
- Zusätzliche Informationen: Enthält Informationen zur Gesichtsverdeckung und zur Blickrichtung.

Vergleich von Gesichtern

Ein Gesichtsvergleichssystem konzentriert sich auf die Frage: „Stimmt das Gesicht auf einem Bild mit einem Gesicht auf einem anderen Bild überein?“ Zu den Funktionen des Gesichtsvergleichssystems gehören:

- Vorhersagen zum Gesichtsabgleich: Vergleicht ein Gesicht in einem Bild mit einem Gesicht in einer bereitgestellten Datenbank, um Treffer vorherzusagen.
- Behandlung von Gesichtsattributen: Verarbeitet Attribute, um Gesichter unabhängig von Gesichtsausdruck, Gesichtsbehaarung und Alter zu vergleichen.

Konfidenzwerte und verpasste Erkennungen

Sowohl Gesichtserkennungs- als auch Gesichtsvergleichssysteme verwenden Konfidenzwerte. Ein Konfidenzwert gibt die Wahrscheinlichkeit von Vorhersagen an, z. B. das Vorhandensein eines Gesichts oder eine Übereinstimmung zwischen Gesichtern. Höhere Werte deuten auf eine höhere Wahrscheinlichkeit hin. Beispielsweise deutet ein Konfidenzwert von 90% auf eine höhere Wahrscheinlichkeit einer korrekten Erkennung oder Übereinstimmung hin als 60%

Wenn ein Gesichtserkennungssystem ein Gesicht nicht richtig erkennt oder eine Vorhersage mit geringer Zuverlässigkeit für ein echtes Gesicht liefert, ist dies ein verpasstes detection/false negative. If the system incorrectly predicts the presence of a face at a high confidence level, this is a false alarm/false Ergebnis.

Ebenso kann es sein, dass ein Gesichtsvergleichssystem zwei Gesichter, die derselben Person gehören, nicht zuordnen kann (verpasstes detection/false negative) or may incorrectly predict that two faces from different people are the same person (false alarm/false positives Ergebnis).

Anwendungsdesign und Schwellenwerteinstellung

- Sie können einen Schwellenwert festlegen, der das Mindestkonfidenzniveau angibt, das für die Rückgabe eines Ergebnisses erforderlich ist. Die Auswahl geeigneter Konfidenzschwellen ist für

das Anwendungsdesign und die Entscheidungsfindung auf der Grundlage der Systemergebnisse von entscheidender Bedeutung.

- Das von Ihnen gewählte Konfidenzniveau sollte Ihren Anwendungsfall widerspiegeln. Einige Beispiele für Anwendungsfälle und Konfidenzschwellen:
- • Fotoanwendungen: Ein niedrigerer Schwellenwert (z. B. 80%) könnte ausreichen, um Familienmitglieder auf Fotos zu identifizieren.
  - Szenarien mit hohem Risiko: In Anwendungsfällen, in denen das Risiko einer verpassten Erkennung oder eines Fehlalarms höher ist, wie z. B. bei Sicherheitsanwendungen, sollte das System ein höheres Konfidenzniveau verwenden. In solchen Fällen wird ein höherer Schwellenwert (z. B. 99%) für genaue Gesichtsabgleiche empfohlen.

Weitere Informationen zur Festlegung und Erläuterung von Konfidenzschwellen finden Sie unterGesichtssuche in einer Sammlung.

Richtlinien zu Gesichtsattributen

Im Folgenden finden Sie Einzelheiten dazu, wie Amazon Rekognition Gesichtsattribute verarbeitet und zurückgibt.

- FaceDetail Objekt: Für jedes erkannte Gesicht wird ein FaceDetail Objekt zurückgegeben. Dies FaceDetail enthält Daten zu Wahrzeichen, Qualität, Pose und mehr von Gesichtern.
- Attributvorhersagen: Attribute wie Emotionen, Geschlecht, Alter und andere werden vorhergesagt. Jeder Vorhersage wird ein Konfidenzniveau zugewiesen, und die Vorhersagen werden mit dem jeweiligen Konfidenzwert zurückgegeben. Für sensible Anwendungsfälle wird ein Konfidenzschwellenwert von 99% empfohlen. Für die Altersschätzung bietet der Mittelpunkt der prognostizierten Altersspanne die beste Näherung.

Beachten Sie, dass Geschlechts- und Emotionsvorhersagen auf der körperlichen Erscheinung basieren und nicht zur Bestimmung der tatsächlichen Geschlechtsidentität oder des emotionalen Zustands verwendet werden sollten. Eine Geschlecht-Binärprognose (männlich/weiblich) basiert auf dem physischen Erscheinungsbild eines Gesichts in einem bestimmten Bild. Es gibt keinen Hinweis auf die Geschlechtsidentität einer Person, und Sie sollten Rekognition nicht verwenden, um eine solche Entscheidung zu treffen. Wir raten davon ab, anhand von binären Geschlechtervoraussagen Entscheidungen zu treffen, durch sich auf die Rechte, Datenschutz oder Zugriff auf Services von Einzelpersonen auswirken. In ähnlicher Weise gibt eine Vorhersage eines Gefühls keinen Hinweis auf den tatsächlichen inneren emotionalen Zustand einer Person, und Sie sollten Rekognition nicht

verwenden, um eine solche Entscheidung zu treffen. Eine Person, die auf einem Bild vorgibt, ein glückliches Gesicht zu haben, sieht vielleicht glücklich aus, ist aber möglicherweise nicht glücklich.

Anwendung und Anwendungsfälle

Hier sind einige praktische Anwendungen und Anwendungsfälle für diese Attribute:

- Anwendungen: Attribute wie Lächeln, Pose und Schärfe können zur Auswahl von Profilbildern oder zur anonymen Schätzung demografischer Daten verwendet werden.
- Häufige Anwendungsfälle: Social-Media-Anwendungen und demografische Schätzungen bei Veranstaltungen oder Einzelhandelsgeschäften sind typische Beispiele.

Ausführlichere Informationen zu den einzelnen Attributen finden Sie unter FaceDetail.

Erkennen von Gesichtern in einem Bild

Amazon Rekognition Image ermöglicht die DetectFacesSuche nach wichtigen Gesichtsmerkmalen wie Augen, Nase und Mund, um Gesichter in einem Eingabebild zu erkennen. Amazon Rekognition Image erkennt die 100 größten Gesichter in einem Bild.

Sie können das Eingabebild als Array von Bild-Bytes (base64-kodierte Bild-Bytes) bereitstellen oder ein Amazon-S3-Objekt festlegen. Bei diesem Verfahren laden Sie ein Bild (JPEG oder PNG) auf Ihren S3-Bucket hoch und geben den Namen des Objektschlüssels an.

So erkennen Sie Gesichter in einem Bild

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`- und `AmazonS3ReadOnlyAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das. AWS SDKs Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Laden Sie ein Bild in Ihren S3-Bucket hoch, auf dem ein oder mehrere Gesichter abgebildet sind.

   Weitere Anleitungen finden Sie unter Upload eines Objekts in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service.

3. Halten Sie sich an die folgenden Beispiele zum Aufruf von `DetectFaces`.

Java

In diesem Beispiel werden der geschätzte Altersbereich für alle erkannten Gesichter und die JSON für alle erkannten Gesichtsmerkmale angezeigt. Ändern Sie den Wert von `photo` in den Bilddateinamen. Ändern Sie den Wert von `amzn-s3-demo-bucket` in den Amazon-S3-Bucket, in dem das Bild gespeichert ist.

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

package aws.example.rekognition.image;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.AgeRange;
import com.amazonaws.services.rekognition.model.Attribute;
import com.amazonaws.services.rekognition.model.DetectFacesRequest;
import com.amazonaws.services.rekognition.model.DetectFacesResult;
import com.amazonaws.services.rekognition.model.FaceDetail;
import com.fasterxml.jackson.databind.ObjectMapper;
import java.util.List;


public class DetectFaces {


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

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

      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();


      DetectFacesRequest request = new DetectFacesRequest()
         .withImage(new Image()
            .withS3Object(new S3Object()
```

```
                .withName(photo)
                .withBucket(bucket)))
           .withAttributes(Attribute.ALL);
        // Replace Attribute.ALL with Attribute.DEFAULT to get default values.

        try {
           DetectFacesResult result = rekognitionClient.detectFaces(request);
           List < FaceDetail > faceDetails = result.getFaceDetails();

           for (FaceDetail face: faceDetails) {
              if (request.getAttributes().contains("ALL")) {
                 AgeRange ageRange = face.getAgeRange();
                 System.out.println("The detected face is estimated to be between
 "
                    + ageRange.getLow().toString() + " and " +
 ageRange.getHigh().toString()
                    + " years old.");
                 System.out.println("Here's the complete set of attributes:");
              } else { // non-default attributes have null values.
                 System.out.println("Here's the default set of attributes:");
              }

              ObjectMapper objectMapper = new ObjectMapper();

 System.out.println(objectMapper.writerWithDefaultPrettyPrinter().writeValueAsString(fac
           }

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

     }

}
```

Java V2

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
import java.util.List;
```

```
//snippet-start:[rekognition.java2.detect_labels.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.DetectFacesRequest;
import software.amazon.awssdk.services.rekognition.model.DetectFacesResponse;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceDetail;
import software.amazon.awssdk.services.rekognition.model.AgeRange;

//snippet-end:[rekognition.java2.detect_labels.import]

public class DetectFaces {

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage: " +
            "   <bucket> <image>\n\n" +
            "Where:\n" +
            "   bucket - The name of the Amazon S3 bucket that contains the
 image (for example, ,amzn-s3-demo-bucket)." +
            "   image - The name of the image located in the Amazon S3 bucket
 (for example, Lake.png). \n\n";

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

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

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

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

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

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

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

               DetectFacesResponse facesResponse =
 rekClient.detectFaces(facesRequest);
               List<FaceDetail> faceDetails = facesResponse.faceDetails();
               for (FaceDetail face : faceDetails) {
                   AgeRange ageRange = face.ageRange();
                   System.out.println("The detected face is estimated to be
 between "
                                   + ageRange.low().toString() + " and " +
 ageRange.high().toString()
                                   + " years old.");

                   System.out.println("There is a smile :
 "+face.smile().value().toString());
               }

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

AWS CLI

In diesem Beispiel wird die JSON-Ausgabe des `detect-faces` AWS CLI Vorgangs angezeigt. Ersetzen Sie `file` durch den Namen einer Bilddatei. Ersetzen Sie `amzn-s3-demo-bucket` durch den Namen des Amazon-S3-Buckets, der die Bilddatei enthält.

```
aws rekognition detect-faces --image "{"S3Object":{"Bucket":"amzn-s3-demo-
bucket,"Name":"image-name"}}"\
                            --attributes "ALL" --profile profile-name --region
 region-name
```

Wenn Sie auf einem Windows-Gerät auf die CLI zugreifen, verwenden Sie doppelte Anführungszeichen anstelle von einfachen Anführungszeichen und maskieren Sie die inneren doppelten Anführungszeichen durch einen Backslash (d. h. \), um eventuell auftretende Parserfehler zu beheben. Sehen Sie sich zum Beispiel Folgendes an:

```
aws rekognition detect-faces --image "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --attributes "ALL"
--profile profile-name --region region-name
```

Python

In diesem Beispiel werden der geschätzte Altersbereich und andere Attribute für alle erkannten Gesichter und die JSON für alle erkannten Gesichtsmerkmale angezeigt. Ändern Sie den Wert von `photo` in den Bilddateinamen. Ändern Sie den Wert von `amzn-s3-demo-bucket` in den Amazon-S3-Bucket, in dem das Bild gespeichert ist. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
import boto3
import json

def detect_faces(photo, bucket, region):

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

```
    client = session.client('rekognition', region_name=region)

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

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

        print('Here are the other attributes:')
        print(json.dumps(faceDetail, indent=4, sort_keys=True))

        # Access predictions for individual face details and print them
        print("Gender: " + str(faceDetail['Gender']))
        print("Smile: " + str(faceDetail['Smile']))
        print("Eyeglasses: " + str(faceDetail['Eyeglasses']))
        print("Face Occluded: " + str(faceDetail['FaceOccluded']))
        print("Emotions: " + str(faceDetail['Emotions'][0]))

    return len(response['FaceDetails'])

def main():
    photo='photo'
    bucket='amzn-s3-demo-bucket'
    region='region'
    face_count=detect_faces(photo, bucket, region)
    print("Faces detected: " + str(face_count))

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

.NET

In diesem Beispiel werden der geschätzte Altersbereich für alle erkannten Gesichter und die JSON für alle erkannten Gesichtsmerkmale angezeigt. Ändern Sie den Wert von `photo` in den Bilddateinamen. Ändern Sie den Wert von `amzn-s3-demo-bucket` in den Amazon-S3-Bucket, in dem das Bild gespeichert ist.

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

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

using System;
using System.Collections.Generic;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

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

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        DetectFacesRequest detectFacesRequest = new DetectFacesRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket
                },
            },
            // Attributes can be "ALL" or "DEFAULT".
            // "DEFAULT": BoundingBox, Confidence, Landmarks, Pose, and Quality.
            // "ALL": See https://docs.aws.amazon.com/sdkfornet/v3/apidocs/
items/Rekognition/TFaceDetail.html
            Attributes = new List<String>() { "ALL" }
        };

        try
        {
            DetectFacesResponse detectFacesResponse =
 rekognitionClient.DetectFaces(detectFacesRequest);
            bool hasAll = detectFacesRequest.Attributes.Contains("ALL");
            foreach(FaceDetail face in detectFacesResponse.FaceDetails)
            {
                Console.WriteLine("BoundingBox: top={0} left={1} width={2}
 height={3}", face.BoundingBox.Left,
```

```
                    face.BoundingBox.Top, face.BoundingBox.Width,
 face.BoundingBox.Height);
                Console.WriteLine("Confidence: {0}\nLandmarks: {1}\nPose:
 pitch={2} roll={3} yaw={4}\nQuality: {5}",
                    face.Confidence, face.Landmarks.Count, face.Pose.Pitch,
                    face.Pose.Roll, face.Pose.Yaw, face.Quality);
                if (hasAll)
                    Console.WriteLine("The detected face is estimated to be
 between " +
                        face.AgeRange.Low + " and " + face.AgeRange.High + "
 years old.");
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```

Ruby

In diesem Beispiel wird der geschätzte Altersbereich für erkannte Gesichter angezeigt, außerdem sind verschiedene Gesichtsattribute aufgeführt. Ändern Sie den Wert von `photo` in den Bilddateinamen. Ändern Sie den Wert von `amzn-s3-demo-bucket` in den Amazon-S3-Bucket, in dem das Bild gespeichert ist.

```
  # 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

In diesem Beispiel wird der geschätzte Altersbereich für erkannte Gesichter angezeigt, außerdem sind verschiedene Gesichtsattribute aufgeführt. Ändern Sie den Wert von `photo` in den Bilddateinamen. Ändern Sie den Wert von `amzn-s3-demo-bucket` in den Amazon-S3-Bucket, in dem das Bild gespeichert ist.

Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

Wenn Sie TypeScript Definitionen verwenden, müssen Sie möglicherweise `import AWS from 'aws-sdk'` anstelle von`const AWS = require('aws-sdk')`, verwenden, um das Programm mit Node.js auszuführen. Weitere Informationen finden Sie im AWS - SDK für Javascript. Je nachdem, wie Sie Ihre Konfigurationen eingerichtet haben, müssen Sie möglicherweise auch Ihre Region mit `AWS.config.update({region:`*`region`*`});` angeben.

```
// 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 Operationsanforderung

Die Eingabe in `DetectFaces` ist ein Bild. In diesem Beispiel wird das Bild aus einem Amazon-S3-Bucket geladen. Der `Attributes`-Parameter gibt an, dass alle Gesichtsmerkmale zurückgegeben werden sollen. Weitere Informationen finden Sie unter Arbeiten mit Bildern.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "input.jpg"
        }
    },
    "Attributes": [
        "ALL"
    ]
}
```

DetectFaces Antwort auf die Operation

`DetectFaces` gibt die folgenden Informationen für jedes erkannte Gesicht zurück:

- Begrenzungsrahmen – Koordinaten des Begrenzungsrahmens, der das Gesicht umgibt.
- Zuverlässigkeit – Maß an Sicherheit, dass der Begrenzungsrahmen ein Gesicht enthält.
- Wichtige Gesichtsmerkmale – Eine Vielzahl von wichtigen Gesichtsmerkmalen. Für jedes wichtige Merkmal (wie beispielsweise das linke Auge, das rechte Auge oder den Mund) liefert die Antwort die X- und Y-Koordinaten.
- Gesichtsattribute: Eine Reihe von Gesichtsattributen, z. B. ob das Gesicht verdeckt ist, wird als `FaceDetail`-Objekt zurückgegeben. Das Set beinhaltet: Bart AgeRange, Emotions,

Brillen EyeDirection,, Gender, EyesOpen FaceOccluded, Schnurrbart MouthOpen, Lächeln und Sonnenbrille. Für jedes dieser Attribute liefert die Antwort einen Wert. Der Wert kann von unterschiedlichem Typ sein, z. B. ein boolescher Typ (ob eine Person eine Sonnenbrille trägt), eine Zeichenkette (ob die Person männlich oder weiblich ist) oder ein Winkelgradwert (für die Blickrichtung der Augen). Außerdem enthält die Antwort für die meisten Attribute einen Zuverlässigkeitswert für den erkannten Wert. Beachten Sie, dass die EyeDirection Attribute FaceOccluded und zwar bei der Verwendung unterstützt werden`DetectFaces`, bei der Analyse von Videos mit und jedoch nicht. `StartFaceDetection` `GetFaceDetection`

- Qualität – Beschreibt die Helligkeit und die Schärfe des Gesichts. Informationen zur Sicherstellung der bestmöglichen Gesichtserkennung finden Sie unter Empfehlungen zu Eingabebildern für den Gesichtsvergleich.
- Pose – Beschreibt die Rotation des Gesichts auf dem Bild.

Die Anforderung kann eine Reihe von Gesichtsattributen enthalten, die Sie zurückgeben möchten. Eine `DEFAULT`-Teilmenge der Gesichtsattribute – `BoundingBox`, `Confidence`, `Pose`, `Quality` und `Landmarks` – wird immer zurückgegeben. Sie können die Rückgabe bestimmter Gesichtsattribute (zusätzlich zur Standardliste) anfordern, indem Sie `["DEFAULT", "FACE_OCCLUDED", "EYE_DIRECTION"]` oder nur ein Attribut verwenden, wie z. B. `["FACE_OCCLUDED"]`, verwenden. Sie können alle Gesichtsattribute anfordern, indem Sie `["ALL"]` verwenden. Das Anfordern weiterer Attribute kann die Antwortzeit verlängern.

Nachfolgend finden Sie eine Beispielantwort eines Amazon-Rekognition-Video-API`DetectFaces`-Aufrufs.


```
{
  "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
  }
}
```


Beachten Sie Folgendes:

- Die `Pose` Daten beschreiben die Rotation des erkannten Gesichts. Auf Basis der Daten zu `BoundingBox` und `Pose` können Sie den Begrenzungsrahmen um die Gesichter ziehen, die Ihre Anwendung anzeigt.
- Das Attribut `Quality` beschreibt die Helligkeit und die Schärfe des Gesichts. Es kann dabei helfen, Gesichter auf mehreren Bildern zu vergleichen und das Beste auszuwählen.
- Die vorherige Antwort zeigt alle `landmarks` des Gesichts, die der Dienst erkennen kann, sowie alle Gesichtsattribute und Emotionen. Um all diese Ergebnisse in der Antwort zu erhalten, müssen Sie den Parameter `attributes` mit Wert `ALL` angeben. Standardmäßig liefert die `DetectFaces`-API nur die folgenden fünf Gesichtsattribute: `BoundingBox`, `Confidence`, `Pose`, `Quality`

und `landmarks`. Die gelieferten standardmäßigen Merkmale sind: `eyeLeft`, `eyeRight`, `nose`, `mouthLeft` und `mouthRight`.

Vergleichen von Gesichtern in Bildern

Mit Rekognition können Sie mithilfe der Operation Gesichter zwischen zwei Bildern vergleichen. CompareFaces Diese Funktion ist nützlich für Anwendungen wie die Identitätsprüfung oder den Abgleich von Fotos.

CompareFaces vergleicht ein Gesicht im Quellbild mit jedem Gesicht im Zielbild. Bilder werden entweder CompareFaces wie folgt übergeben:

- Eine Base64-kodierte Darstellung eines Bilds.
- Amazon S3 S3-Objekte.

Vergleich von Gesichtserkennung und Gesichtserkennung

Der Gesichtsvergleich unterscheidet sich von der Gesichtserkennung. Die Gesichtserkennung (die verwendet DetectFaces) identifiziert nur das Vorhandensein und die Position von Gesichtern in einem Bild oder Video. Im Gegensatz dazu wird beim Gesichtsvergleich ein erkanntes Gesicht in einem Quellbild mit Gesichtern in einem Zielbild verglichen, um Übereinstimmungen zu finden.

Schwellenwerte für Ähnlichkeiten

Verwenden Sie den `similarityThreshold` Parameter, um das Mindestkonfidenzniveau für Übereinstimmungen zu definieren, die in die Antwort aufgenommen werden sollen. Standardmäßig werden in der Antwort nur Gesichter mit einem Ähnlichkeitswert von mehr als oder gleich 80% zurückgegeben.

> **Note**
>
> `CompareFaces`verwendet Algorithmen für maschinelles Lernen, die probabilistisch sind. Ein falsches Negativ ist eine falsche Vorhersage, dass ein Gesicht im Zielbild im Vergleich zum Gesicht im Quellbild einen niedrigen Ähnlichkeitswert aufweist. Um die Wahrscheinlichkeit falsch negativer Bilder zu verringern, empfehlen wir, das Zielbild mit mehreren Quellbildern zu vergleichen. Wenn Sie `CompareFaces` nutzen möchten, um eine Entscheidung zu treffen, die sich auf die Rechte, den Datenschutz oder den Zugang zu Diensten einer Person

auswirkt, empfehlen wir Ihnen, das Ergebnis zur weiteren Überprüfung und Bestätigung an einen Mitarbeiter weiterzuleiten, bevor Sie Maßnahmen ergreifen.

Die folgenden Codebeispiele zeigen, wie die CompareFaces Operationen für verschiedene verwendet werden. AWS SDKs In dem AWS CLI Beispiel laden Sie zwei JPEG-Bilder in Ihren Amazon S3 S3-Bucket hoch und geben den Objektschlüsselnamen an. In den anderen Beispielen laden Sie zwei Dateien aus dem lokalen Dateisystem hoch und geben sie als Bild-Byte-Arrays ein.

So vergleichen Sie Gesichter

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess` und `AmazonS3ReadOnlyAccess` (nur AWS CLI Beispiel) Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie den folgenden Beispiel-Code zum Aufrufen der `CompareFaces`-Operation.

   Java

   Dieses Beispiel zeigt Informationen über die übereinstimmenden Gesichter in Quell- und Zielbildern an, die aus dem lokalen Dateisystem geladen werden.

   Ersetzen Sie die Werte von `sourceImage` und `targetImage` durch den Pfad und Dateinamen der Quell- und Zielbilder.

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

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.CompareFacesMatch;
```

```
import com.amazonaws.services.rekognition.model.CompareFacesRequest;
import com.amazonaws.services.rekognition.model.CompareFacesResult;
import com.amazonaws.services.rekognition.model.ComparedFace;
import java.util.List;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import com.amazonaws.util.IOUtils;

public class CompareFaces {

   public static void main(String[] args) throws Exception{
       Float similarityThreshold = 70F;
       String sourceImage = "source.jpg";
       String targetImage = "target.jpg";
       ByteBuffer sourceImageBytes=null;
       ByteBuffer targetImageBytes=null;

       AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

       //Load source and target images and create input parameters
       try (InputStream inputStream = new FileInputStream(new
 File(sourceImage))) {
          sourceImageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
       }
       catch(Exception e)
       {
           System.out.println("Failed to load source image " + sourceImage);
           System.exit(1);
       }
       try (InputStream inputStream = new FileInputStream(new
 File(targetImage))) {
           targetImageBytes =
 ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
       }
       catch(Exception e)
       {
           System.out.println("Failed to load target images: " + targetImage);
           System.exit(1);
       }

       Image source=new Image()
```

```
            .withBytes(sourceImageBytes);
       Image target=new Image()
            .withBytes(targetImageBytes);

       CompareFacesRequest request = new CompareFacesRequest()
               .withSourceImage(source)
               .withTargetImage(target)
               .withSimilarityThreshold(similarityThreshold);

       // Call operation
       CompareFacesResult
 compareFacesResult=rekognitionClient.compareFaces(request);


       // Display results
       List <CompareFacesMatch> faceDetails =
 compareFacesResult.getFaceMatches();
       for (CompareFacesMatch match: faceDetails){
         ComparedFace face= match.getFace();
         BoundingBox position = face.getBoundingBox();
         System.out.println("Face at " + position.getLeft().toString()
               + " " + position.getTop()
               + " matches with " + match.getSimilarity().toString()
               + "% confidence.");

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

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

Java V2

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
import java.util.List;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
```

```
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import software.amazon.awssdk.services.rekognition.model.CompareFacesMatch;
import software.amazon.awssdk.services.rekognition.model.CompareFacesRequest;
import software.amazon.awssdk.services.rekognition.model.CompareFacesResponse;
import software.amazon.awssdk.services.rekognition.model.ComparedFace;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;

// snippet-end:[rekognition.java2.detect_faces.import]

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

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "   <pathSource> <pathTarget>\n\n" +
            "Where:\n" +
            "   pathSource - The path to the source image (for example, C:\\AWS\
\pic1.png). \n " +
            "   pathTarget - The path to the target image (for example, C:\\AWS\
\pic2.png). \n\n";

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

        Float similarityThreshold = 70F;
        String sourceImage = args[0];
        String targetImage = args[1];
```

```
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        compareTwoFaces(rekClient, similarityThreshold, sourceImage,
 targetImage);
        rekClient.close();
   }

    // snippet-start:[rekognition.java2.compare_faces.main]
    public static void compareTwoFaces(RekognitionClient rekClient, Float
 similarityThreshold, String sourceImage, String targetImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            InputStream tarStream = new FileInputStream(targetImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            SdkBytes targetBytes = SdkBytes.fromInputStream(tarStream);

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

            Image tarImage = Image.builder()
                .bytes(targetBytes)
                .build();

            CompareFacesRequest facesRequest = CompareFacesRequest.builder()
                .sourceImage(souImage)
                .targetImage(tarImage)
                .similarityThreshold(similarityThreshold)
                .build();

            // Compare the two images.
            CompareFacesResponse compareFacesResult =
 rekClient.compareFaces(facesRequest);
            List<CompareFacesMatch> faceDetails =
 compareFacesResult.faceMatches();
            for (CompareFacesMatch match: faceDetails){
                ComparedFace face= match.face();
                BoundingBox position = face.boundingBox();
```

```
                System.out.println("Face at " + position.left().toString()
                        + " " + position.top()
                        + " matches with " + face.confidence().toString()
                        + "% confidence.");

            }
            List<ComparedFace> uncompared = compareFacesResult.unmatchedFaces();
            System.out.println("There was " + uncompared.size() + " face(s) that
 did not match");
            System.out.println("Source image rotation: " +
 compareFacesResult.sourceImageOrientationCorrection());
            System.out.println("target image rotation: " +
 compareFacesResult.targetImageOrientationCorrection());

        } catch(RekognitionException | FileNotFoundException e) {
            System.out.println("Failed to load source image " + sourceImage);
            System.exit(1);
        }
    }
    // snippet-end:[rekognition.java2.compare_faces.main]
}
```

AWS CLI

In diesem Beispiel wird die JSON-Ausgabe des `compare-faces` AWS CLI Vorgangs angezeigt.

Ersetzen Sie `amzn-s3-demo-bucket` durch den Namen des Amazon-S3-Buckets, der die Quell- und Zielbilder enthält. Ersetzen Sie `source.jpg` und `target.jpg` durch die Dateinamen für die Quell- und Zielbilder.

```
aws rekognition compare-faces --target-image \
"{"S3Object":{"Bucket":"amzn-s3-demo-bucket","Name":"image-name"}}" \
--source-image "{"S3Object":{"Bucket":"amzn-s3-demo-bucket","Name":"image-
name"}}"
--profile profile-name
```

Wenn Sie auf einem Windows-Gerät auf die CLI zugreifen, verwenden Sie doppelte Anführungszeichen anstelle von einfachen Anführungszeichen und maskieren Sie die inneren

doppelten Anführungszeichen durch einen Backslash (d. h. \), um eventuell auftretende Parserfehler zu beheben. Sehen Sie sich zum Beispiel Folgendes an:

```
aws rekognition compare-faces --target-image "{\"S3Object\":{\"Bucket\":\"amzn-
s3-demo-bucket\",\"Name\":\"image-name\"}}" \
--source-image "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-bucket\",\"Name\":
\"image-name\"}}" --profile profile-name
```

Python

Dieses Beispiel zeigt Informationen über die übereinstimmenden Gesichter in Quell- und Zielbildern an, die aus dem lokalen Dateisystem geladen werden.

Ersetzen Sie die Werte von `source_file` und `target_file` durch den Pfad und Dateinamen der Quell- und Zielbilder. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
# Copyright 2018 Amazon.com, Inc. or its 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

Dieses Beispiel zeigt Informationen über die übereinstimmenden Gesichter in Quell- und Zielbildern an, die aus dem lokalen Dateisystem geladen werden.

Ersetzen Sie die Werte von `sourceImage` und `targetImage` durch den Pfad und Dateinamen der Quell- und Zielbilder.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Dieses Beispiel zeigt Informationen über die übereinstimmenden Gesichter in Quell- und Zielbildern an, die aus dem lokalen Dateisystem geladen werden.

Ersetzen Sie die Werte von `photo_source` und `photo_target` durch den Pfad und Dateinamen der Quell- und Zielbilder.

```
 # 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

Dieses Beispiel zeigt Informationen über die übereinstimmenden Gesichter in Quell- und Zielbildern an, die aus dem lokalen Dateisystem geladen werden.

Ersetzen Sie die Werte von `photo_source` und `photo_target` durch den Pfad und Dateinamen der Quell- und Zielbilder. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
// 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 Operationsanforderung

Die Eingabe in `CompareFaces` ist ein Bild. In diesem Beispiel werden die Quell- und Zielbilder aus dem lokalen Dateisystem geladen. Der `SimilarityThreshold`-Eingabeparameter gibt den minimalen Zuverlässigkeitswert an, dem verglichene Gesichter entsprechen müssen, um in die Antwort aufgenommen zu werden. Weitere Informationen finden Sie unter Arbeiten mit Bildern.

```
{
    "SourceImage": {
        "Bytes": "/9j/4AAQSk2Q==..."
    },
    "TargetImage": {
        "Bytes": "/9j/4O1Q==..."
    },
    "SimilarityThreshold": 70
}
```

CompareFaces Antwort auf die Operation

Die Antwort enthält:

- Eine Reihe von Übereinstimmungen mit Gesichtern: Eine Liste übereinstimmender Gesichter mit Ähnlichkeitswerten und Metadaten für jedes passende Gesicht. Wenn mehrere Gesichter übereinstimmen, `faceMatches`

  Das Array umfasst alle übereinstimmenden Gesichter.
- Details zur Übereinstimmung von Gesichtern: Für jedes übereinstimmende Gesicht stehen außerdem ein Begrenzungsrahmen, ein Konfidenzwert, die Position von Wahrzeichen und eine Ähnlichkeitsbewertung zur Verfügung.
- Eine Liste nicht übereinstimmender Gesichter: Die Antwort umfasst auch Gesichter aus dem Zielbild, die nicht mit dem Gesicht auf dem Quellbild übereinstimmen. Beinhaltet einen Begrenzungsrahmen für jedes Gesicht, das nicht zugeordnet wurde.
- Informationen zum Quellgesicht: Enthält Informationen über das Gesicht aus dem Quellbild, das für den Vergleich verwendet wurde, einschließlich des Begrenzungsrahmens und des Konfidenzwerts.

Das Beispiel zeigt, dass im Zielbild eine Gesichtsübereinstimmung gefunden wurde. Für diese Gesichtsübereinstimmung liefert sie einen Begrenzungsrahmen und einen Zuverlässigkeitswert (das Maß an Vertrauen, das Amazon Rekognition hat, dass der Begrenzungsrahmen ein Gesicht enthält). Der Ähnlichkeitswert von 99,99 gibt an, wie ähnlich sich die Gesichter sind. Das Beispiel zeigt auch ein Gesicht, das Amazon Rekognition im Zielbild gefunden hat und das nicht mit dem Gesicht übereinstimmt, das im Quellbild analysiert wurde.


```
{
    "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
```

```
    }
}
```

Erkennen von Gesichtern in einem gespeicherten Video

Amazon Rekognition Video kann Gesichter in Videos, die in einem Amazon-S3-Bucket gespeichert sind, erkennen und Informationen bereitstellen wie:

- Die Zeit(en), wann die Gesichter in einem Video erkannt werden.
- Die Position von Gesichtern im Videobild zum Zeitpunkt der Erkennung.
- Gesichtsmarken wie z. B. die Position des linken Auges.
- Zusätzliche Attribute, wie auf der Seite the section called “Richtlinien zu Gesichtsattributen” erklärt.

Die Gesichtserkennung von Amazon Rekognition Video in gespeicherten Videos ist eine asynchrone Operation. Rufen Sie an, um mit der Erkennung von Gesichtern in Videos zu beginnen. StartFaceDetection Amazon Rekognition Video veröffentlicht den Abschlussstatus der Videoanalyse in einem Amazon-Simple-Notification-Service-Thema (Amazon SNS). Wenn die Videoanalyse erfolgreich ist, können Sie GetFaceDetection aufrufen, um die Ergebnisse der Videoanalyse abzurufen. Weitere Informationen zum Starten der Videoanalyse und zum Abrufen der Ergebnisse finden Sie unter Amazon-Rekognition-Video-Operationen aufrufen.

Dieses Verfahren erweitert den Code in Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK), der eine Amazon-Simple-Queue-Service-Warteschlange (Amazon SQS) verwendet, um den Abschlussstatus einer Videoanalyseanforderung zu erhalten.

So erkennen Sie Gesichter in einem Video, das in einem Amazon-S3-Bucket gespeichert ist (SDK)

1. Führen Sie Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) aus.
2. Fügen Sie den folgenden Code in der Klasse `VideoDetect` ein, die Sie in Schritt 1 erstellt haben.

   AWS CLI

   - Ändern Sie im folgenden Codebeispiel `amzn-s3-demo-bucket` und `video-name` in den Amazon-S3-Bucket-Namen und den Dateinamen, die Sie in Schritt 2 angegeben haben.

- Ändern Sie `region-name` in die von Ihnen verwendete AWS-Region. Ersetzen Sie den Wert von `profile_name` mit dem Namen Ihres Entwicklerprofils.
- Ändern Sie `TopicARN` in den ARN des Amazon-SNS-Themas, das Sie in Schritt 3 von Amazon Rekognition Video konfigurieren erstellt haben.
- Ändern Sie `RoleARN` in den ARN der IAM-Servicerolle, die Sie in Schritt 7 von Amazon Rekognition Video konfigurieren erstellt haben.

```
aws rekognition start-face-detection --video "{"S3Object":{"Bucket":"amzn-s3-
demo-bucket","Name":"Video-Name"}}" --notification-channel
"{"SNSTopicArn":"Topic-ARN","RoleArn":"Role-ARN"}" --region region-name --
profile profile-name
```

Wenn Sie auf einem Windows-Gerät auf die CLI zugreifen, verwenden Sie doppelte Anführungszeichen anstelle von einfachen Anführungszeichen und maskieren Sie die inneren doppelten Anführungszeichen durch einen Backslash (d. h. \), um eventuell auftretende Parserfehler zu beheben. Sehen Sie sich zum Beispiel Folgendes an:

```
aws rekognition start-face-detection --video "{\"S3Object\":{\"Bucket\":\"amzn-
s3-demo-bucket\",\"Name\":\"Video-Name\"}}" --notification-channel
"{\"SNSTopicArn\":\"Topic-ARN\",\"RoleArn\":\"Role-ARN\"}" --region region-name
 --profile profile-name
```

Nachdem Sie die `StartFaceDetection`-Operation ausgeführt und die Auftrags-ID-Nummer abgerufen haben, führen Sie die folgende `GetFaceDetection`-Operation aus und geben Sie die Auftrags-ID-Nummer ein:

```
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);


}
```

Ersetzen Sie in der Funktion `main` die folgenden Zeilen:

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

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

mit:

```
        StartFaceDetection(amzn-s3-demo-bucket, video);

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

Java V2

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
//snippet-start:[rekognition.java2.recognize_video_faces.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import java.util.List;
//snippet-end:[rekognition.java2.recognize_video_faces.import]


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

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

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

     if (args.length != 4) {
         System.out.println(usage);
         System.exit(1);
     }

     String bucket = args[0];
     String video = args[1];
     String topicArn = args[2];
```

```
    String roleArn = args[3];

    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
        .build();

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

    StartFaceDetection(rekClient, channel, bucket, video);
    GetFaceResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

// snippet-start:[rekognition.java2.recognize_video_faces.main]
public static void StartFaceDetection(RekognitionClient rekClient,
                                       NotificationChannel channel,
                                       String bucket,
                                       String video) {

    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

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

        StartFaceDetectionRequest  faceDetectionRequest =
StartFaceDetectionRequest.builder()
            .jobTag("Faces")
            .faceAttributes(FaceAttributes.ALL)
            .notificationChannel(channel)
            .video(vidOb)
            .build();
```

```
        StartFaceDetectionResponse startLabelDetectionResult =
rekClient.startFaceDetection(faceDetectionRequest);
        startJobId=startLabelDetectionResult.jobId();

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

public static void GetFaceResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetFaceDetectionResponse faceDetectionResponse=null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (faceDetectionResponse !=null)
                paginationToken = faceDetectionResponse.nextToken();

            GetFaceDetectionRequest recognitionRequest =
GetFaceDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {

                faceDetectionResponse =
rekClient.getFaceDetection(recognitionRequest);
                status = faceDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
```

```
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null
            VideoMetadata videoMetaData=faceDetectionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());
            System.out.println("Job");

            // Show face information
            List<FaceDetection> faces= faceDetectionResponse.faces();

            for (FaceDetection face: faces) {
                String age = face.face().ageRange().toString();
                String smile = face.face().smile().toString();
                System.out.println("The detected face is estimated to be"
                            + age + " years old.");
                System.out.println("There is a smile : "+smile);
            }

        } while (faceDetectionResponse !=null &&
 faceDetectionResponse.nextToken() != null);

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

Python

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

    # ============== Faces===============
    def StartFaceDetection(self):
```

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

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

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

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

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

            for faceDetection in response['Faces']:
                print('Face: ' + str(faceDetection['Face']))
                print('Confidence: ' + str(faceDetection['Face']['Confidence']))
                print('Timestamp: ' + str(faceDetection['Timestamp']))
                print()

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

Ersetzen Sie in der Funktion `main` die folgenden Zeilen:

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

mit:

```
    analyzer.StartFaceDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetFaceDetectionResults()
```

> **Note**
>
> Wenn Sie bereits ein anderes Video-Beispiel als Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) ausgeführt haben, ist der zu ersetzende Name der Funktion anders.

3. Führen Sie den Code aus. Die Informationen zu den erkannten Gesichtern im Video werden angezeigt.

GetFaceDetection Antwort auf den Vorgang

`GetFaceDetection` gibt ein Array (`Faces`) zurück, das Informationen über die erkannten Gesichter im Video enthält. Das Array -Element FaceDetection wird jedes Mal erzeugt, wenn ein Gesicht im Video erkannt wird. Die Array-Elemente werden nach Zeit sortiert zurückgegeben, in Millisekunden ab dem Start des Videos.

Das folgende Beispiel ist ein Teil einer JSON-Antwort von `GetFaceDetection`. In der Antwort ist Folgendes zu beachten:

- Begrenzungsrahmen – Koordinaten des Begrenzungsrahmens, der das Gesicht umgibt.
- Zuverlässigkeit – Maß an Sicherheit, dass der Begrenzungsrahmen ein Gesicht enthält.
- Wichtige Gesichtsmerkmale – Eine Vielzahl von wichtigen Gesichtsmerkmalen. Für jedes wichtige Merkmal (wie beispielsweise das linke Auge, das rechte Auge oder den Mund) liefert die Antwort die x- und y-Koordinaten.
- Gesichtsattribute — Eine Reihe von Gesichtsattributen, zu denen Bart AgeRange, Emotionen, Brille, Geschlecht, EyesOpen Schnurrbart MouthOpen, Lächeln und Sonnenbrille gehören. Für diesen Wert gibt es unterschiedliche Typen, beispielsweise einen booleschen Typ (ob eine Person eine Sonnenbrille trägt oder nicht) oder eine String (ob die Person männlich oder weiblich ist). Außerdem enthält die Antwort für die meisten Attribute einen Zuverlässigkeitswert für den erkannten Wert. Beachten Sie, dass EyeDirection Attribute FaceOccluded und zwar bei der

Verwendung unterstützt werden`DetectFaces`, bei der Analyse von Videos mit und jedoch nicht. `StartFaceDetection` `GetFaceDetection`

- Zeitstempel: Der Zeitpunkt, zu dem das Gesicht im Video erkannt wurde.
- Seiteninformationen – Das Beispiel zeigt eine Seite mit Informationen der Gesichtererkennung. Sie können festlegen, wie viele Personenelemente zurückgegeben werden sollen, durch den Eingabeparameter `MaxResults` von `GetFaceDetection`. Wenn mehr Ergebnisse als `MaxResults` vorhanden sind, gibt `GetFaceDetection` ein Token zurück (`NextToken`), mit dem die nächste Seite von Ergebnissen angefordert wird. Weitere Informationen finden Sie unter Analyseergebnisse von Amazon Rekognition Video abrufen.
- Video-Informationen – Die Antwort enthält Informationen über das Videoformat (`VideoMetadata`) auf jeder Seite mit Informationen, die von `GetFaceDetection` zurückgegeben werden.
- Qualität – Beschreibt die Helligkeit und die Schärfe des Gesichts.
- Pose – Beschreibt die Drehung des Gesichts.


```
{
    "Faces": [
        {
            "Face": {
                "BoundingBox": {
                    "Height": 0.23000000417232513,
                    "Left": 0.42500001192092896,
                    "Top": 0.16333332657814026,
                    "Width": 0.12937499582767487
                },
                "Confidence": 99.97504425048828,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.46415066719055176,
                        "Y": 0.2572723925113678
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.5068183541297913,
                        "Y": 0.23705792427062988
                    },
                    {
                        "Type": "nose",
                        "X": 0.49765899777412415,
```

```
                    "Y": 0.28383663296699524
                },
                {
                    "Type": "mouthLeft",
                    "X": 0.487221896648407,
                    "Y": 0.3452930748462677
                },
                {
                    "Type": "mouthRight",
                    "X": 0.5142884850502014,
                    "Y": 0.33167609572410583
                }
            ],
            "Pose": {
                "Pitch": 15.966927528381348,
                "Roll": -15.547388076782227,
                "Yaw": 11.34195613861084
            },
            "Quality": {
                "Brightness": 44.80223083496094,
                "Sharpness": 99.95819854736328
            }
        },
        "Timestamp": 0
    },
    {
        "Face": {
            "BoundingBox": {
                "Height": 0.20000000298023224,
                "Left": 0.029999999329447746,
                "Top": 0.2199999988079071,
                "Width": 0.11249999701976776
            },
            "Confidence": 99.85971069335938,
            "Landmarks": [
                {
                    "Type": "eyeLeft",
                    "X": 0.06842322647571564,
                    "Y": 0.3010137975215912
                },
                {
                    "Type": "eyeRight",
                    "X": 0.10543643683195114,
                    "Y": 0.29697132110595703
```

```
                },
                {
                    "Type": "nose",
                    "X": 0.09569807350635529,
                    "Y": 0.33701086044311523
                },
                {
                    "Type": "mouthLeft",
                    "X": 0.0732642263174057,
                    "Y": 0.3757539987564087
                },
                {
                    "Type": "mouthRight",
                    "X": 0.10589495301246643,
                    "Y": 0.3722417950630188
                }
            ],
            "Pose": {
                "Pitch": -0.5589138865470886,
                "Roll": -5.1093974113464355,
                "Yaw": 18.69594955444336
            },
            "Quality": {
                "Brightness": 43.052337646484375,
                "Sharpness": 99.68138885498047
            }
        },
        "Timestamp": 0
    },
    {
        "Face": {
            "BoundingBox": {
                "Height": 0.2177777737379074,
                "Left": 0.7593749761581421,
                "Top": 0.13333334028720856,
                "Width": 0.12250000238418579
            },
            "Confidence": 99.63436889648438,
            "Landmarks": [
                {
                    "Type": "eyeLeft",
                    "X": 0.8005779385566711,
                    "Y": 0.20915353298187256
                },
```

```
                    {
                        "Type": "eyeRight",
                        "X": 0.8391435146331787,
                        "Y": 0.21049551665782928
                    },
                    {
                        "Type": "nose",
                        "X": 0.8191410899162292,
                        "Y": 0.2523227035999298
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.8093273043632507,
                        "Y": 0.29053622484207153
                    },
                    {
                        "Type": "mouthRight",
                        "X": 0.8366993069648743,
                        "Y": 0.29101791977882385
                    }
                ],
                "Pose": {
                    "Pitch": 3.165884017944336,
                    "Roll": 1.4182015657424927,
                    "Yaw": -11.151537895202637
                },
                "Quality": {
                    "Brightness": 28.910892486572266,
                    "Sharpness": 97.61507415771484
                }
            },
            "Timestamp": 0
        }.......

    ],
    "JobStatus": "SUCCEEDED",
    "NextToken": "i7fj5XPV/
fwviXqz0eag9Ow332Jd5G8ZGWf7hooirD/6V1qFmjKFOQZ6QPWUiqv29HbyuhMNqQ==",
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 67301,
        "FileExtension": "mp4",
        "Format": "QuickTime / MOV",
        "FrameHeight": 1080,
```

```
        "FrameRate": 29.970029830932617,
        "FrameWidth": 1920
    }
}
```

Gesichtssuche in einer Sammlung

Mit Amazon Rekognition können Sie ein Eingabegesicht verwenden, um in einer Sammlung von gespeicherten Gesichtern nach Übereinstimmungen zu suchen. Sie beginnen damit, Informationen über erkannte Gesichter in serverseitigen Containern zu speichern, die als „Sammlungen“ bezeichnet werden. Sammlungen speichern sowohl einzelne Gesichter als auch Benutzer (mehrere Gesichter derselben Person). Einzelne Gesichter werden als Gesichtsvektoren gespeichert, eine mathematische Darstellung des Gesichts (kein echtes Bild des Gesichts). Verschiedene Bilder derselben Person können verwendet werden, um mehrere Gesichtsvektoren in derselben Sammlung zu erstellen und zu speichern. Sie können dann mehrere Gesichtsvektoren derselben Person aggregieren, um einen Benutzervektor zu erstellen. Benutzervektoren können eine höhere Genauigkeit bei der Suche nach Gesichtern mit robusteren Darstellungen bieten, die unterschiedliche Lichtverhältnisse, Schärfe, Pose, Erscheinungsbild usw. enthalten.

Sobald Sie eine Sammlung erstellt haben, können Sie ein Eingabegesicht verwenden, um nach passenden Benutzervektoren oder Gesichtsvektoren in einer Sammlung zu suchen. Die Suche anhand von Benutzervektoren kann die Genauigkeit im Vergleich zur Suche anhand einzelner Gesichtsvektoren erheblich verbessern. Sie können Gesichter verwenden, die in Bildern, gespeicherten Videos und Streaming-Videos erkannt wurden, um anhand von gespeicherten Gesichtsvektoren zu suchen. Sie können Gesichter, die in Bildern erkannt wurden, verwenden, um anhand von gespeicherten Benutzervektoren zu suchen.

Um Gesichtsinformationen zu speichern, müssen Sie Folgendes tun:

1. Eine Sammlung erstellen — Um Gesichtsinformationen zu speichern, müssen Sie zunächst eine Gesichtssammlung in einer der AWS Regionen in Ihrem Konto erstellen (CreateCollection). Diese Gesichtersammlung wird näher spezifiziert, sobald Sie die `IndexFaces`-Operation aufrufen.
2. Gesichter indexieren — Der IndexFacesVorgang erkennt Gesichter in einem Bild, extrahiert die Gesichtsvektoren und speichert sie in der Sammlung. Sie können diese Operation verwenden, um Gesichter in einem Bild zu erkennen und um die Informationen über die erkannten Gesichtsmerkmale in einer Sammlung bewahren. Dies ist ein Beispiel für eine speicherbasierte API-Operation, da der Dienst die Gesichtsvektorinformationen auf dem Server speichert.

Um einen Benutzer zu erstellen und mehrere Gesichtsvektoren einem Benutzer zuzuordnen, müssen Sie wie folgt vorgehen:

1. Einen Benutzer erstellen — Sie müssen zuerst einen Benutzer mit erstellen CreateUser. Sie können die Genauigkeit beim Abgleich von Gesichtern verbessern, indem Sie mehrere Gesichtsvektoren derselben Person zu einem Benutzervektor zusammenfassen. Sie können einem Benutzervektor bis zu 100 Gesichtsvektoren zuordnen.
2. Gesichter zuordnen — Nachdem Sie den Benutzer erstellt haben, können Sie diesem Benutzer im Rahmen des AssociateFacesVorgangs vorhandene Gesichtsvektoren hinzufügen. Gesichtsvektoren müssen sich in derselben Sammlung wie ein Benutzervektor befinden, damit sie diesem Benutzervektor zugeordnet werden können.

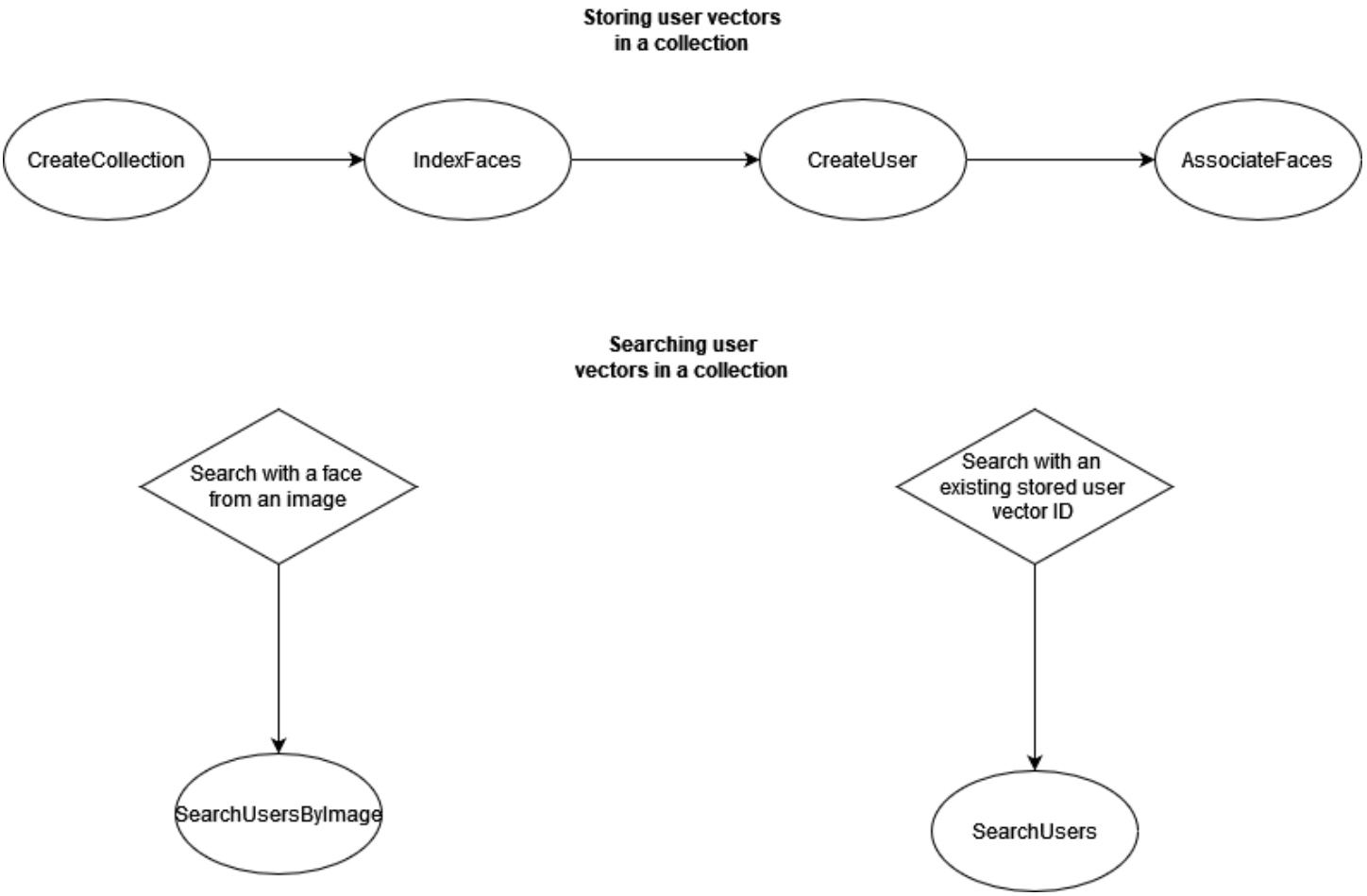
Nachdem Sie eine Sammlung erstellt und Gesichts- und Benutzervektoren gespeichert haben, können Sie mit den folgenden Operationen nach übereinstimmenden Gesichtern suchen:

- SearchFacesByImage- Um anhand von gespeicherten einzelnen Gesichtern anhand eines Gesichts aus einem Bild zu suchen.
- SearchFaces- Um nach gespeicherten einzelnen Gesichtern mit einer angegebenen Gesichts-ID zu suchen.
- SearchUsers- Um nach gespeicherten Benutzern mit einer angegebenen Gesichts-ID oder Benutzer-ID zu suchen.
- SearchUsersByImage- Um nach gespeicherten Benutzern mit einem Gesicht aus einem Bild zu suchen.
- StartFaceSearch- Um in einem gespeicherten Video nach Gesichtern zu suchen.
- CreateStreamProcessor- Um in einem Streaming-Video nach Gesichtern zu suchen.

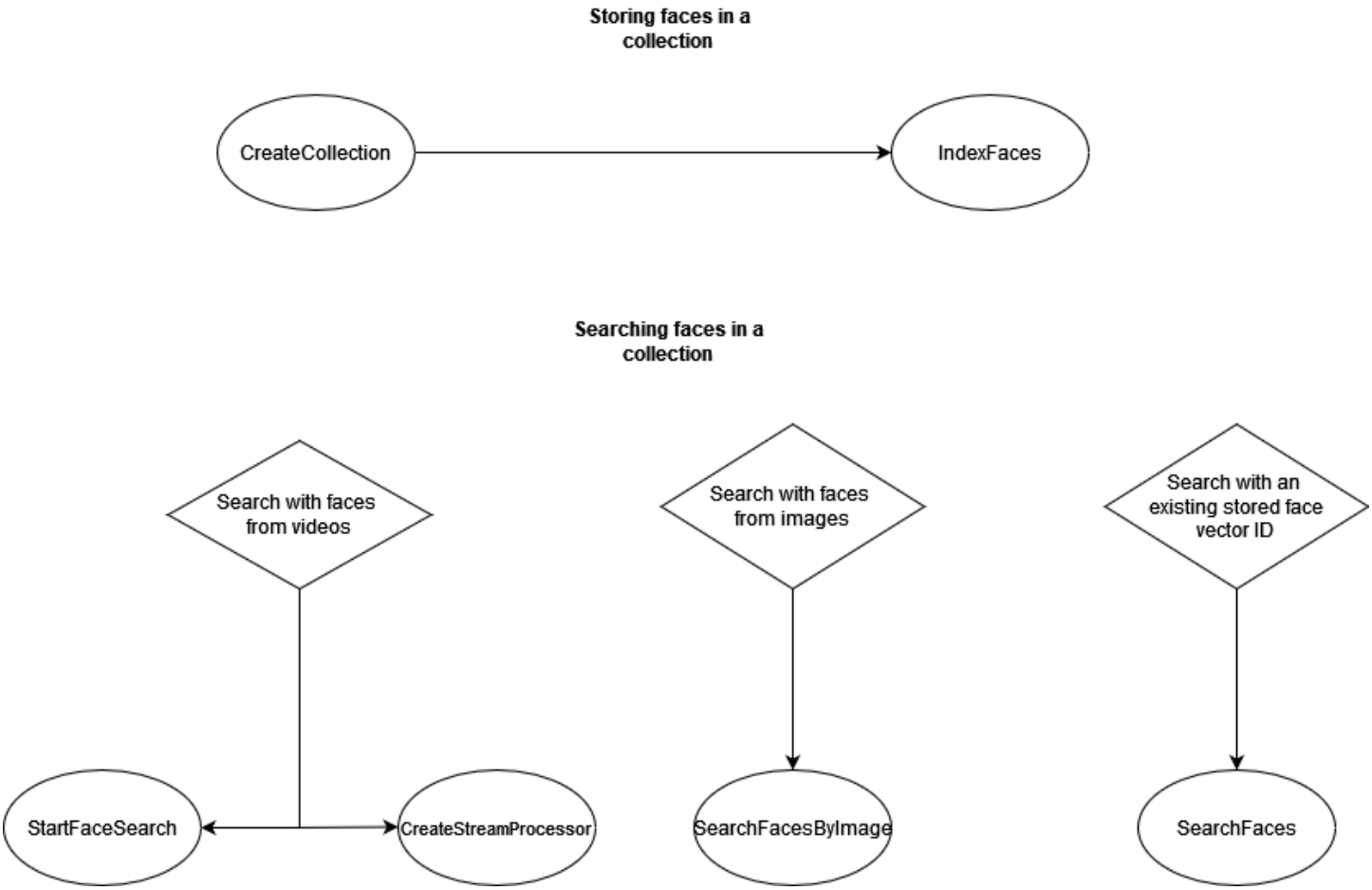
> **Note**
>
> In Sammlungen werden Gesichtsvektoren gespeichert, bei denen es sich um mathematische Darstellungen von Gesichtern handelt. Sammlungen speichern keine Bilder von Gesichtern.

Die folgenden Diagramme zeigen die Reihenfolge der Aufrufvorgänge auf der Grundlage Ihrer Ziele für die Verwendung von Sammlungen:

Für einen möglichst genauen Abgleich mit Benutzervektoren:



Für einen hochgenauen Abgleich mit einzelnen Gesichtsvektoren:



Sie können Sammlungen in verschiedenen Szenarien verwenden. Sie könnten beispielsweise eine Gesichtssammlung erstellen, in der Gesichter gespeichert werden, die anhand von gescannten Bildern von Mitarbeiterausweisen erkannt wurden und die IDs von Behörden mithilfe der `AssociateFaces` Operationen `IndexFaces` und ausgestellt wurden. Sobald ein Mitarbeiter das Gebäude betritt, wird ein Foto seines Gesichts gemacht und an die `SearchUsersByImage`-Operation übermittelt. Wenn bei der Gesichtsübereinstimmung ein ausreichend hoher Ähnlichkeitswert erzielt wird (beispielsweise 99 %), können Sie den Mitarbeiter autorisierten.

Verwaltung von Gesichtssammlungen, Gesichtern und Benutzern

Die Gesichtssammlung ist die Ressource, die von Amazon Rekognition verwendet wird, um Informationen zu Gesichtern und Benutzern zu enthalten. Sie verwenden verschiedene Rekognition-API-Operationen, um die Sammlung selbst sowie alle in der Sammlung gespeicherten Gesichter oder Gesichtsvektoren zu verwalten.

Eine Sammlung verwalten

Die Gesichtersammlung ist die primäre Amazon-Rekognition-Ressource. Jede Gesichtersammlung, die Sie erstellen, hat einen einzigartigen Amazon-Ressourcennamen (ARN). Sie erstellen jede Gesichtssammlung in einer bestimmten AWS Region in Ihrem Konto. Wenn eine Sammlung erstellt wird, wird diese mit der neuesten Version des Gesichtserkennungsmodells verknüpft. Weitere Informationen finden Sie unter Grundlegendes zur Modellversionierung.

Sie können die folgenden Verwaltungsoperationen in einer Sammlung ausführen:

- Erstellen Sie eine Sammlung mit CreateCollection. Weitere Informationen finden Sie unter Erstellen einer Sammlung.
- Listen Sie die verfügbaren Sammlungen mit ListCollections auf. Weitere Informationen finden Sie unter Auflisten von Sammlungen.
- Beschreiben Sie eine Sammlung mit DescribeCollection. Weitere Informationen finden Sie unter Beschreiben einer Sammlung.
- Löschen Sie eine Sammlung mit DeleteCollection. Weitere Informationen finden Sie unter Löschen einer Sammlung.

Verwalten von Gesichtern in einer Sammlung

Nach der Erstellung einer Gesichtersammlung können Sie darin Gesichter speichern. Amazon Rekognition bietet die folgenden Operationen zur Verwaltung von Gesichtern in einer Sammlung:

- Die IndexFaces-Operation erkennt Gesichter auf dem Eingangsbild (JPEG oder PNG) und fügt sie zur festgelegten Gesichtersammlung hinzu. Eine eindeutige Gesichts-ID wird für jedes Gesicht zurückgegeben, das im Bild erkannt wird. Nach der Erhaltung von Gesichtern können Sie die Gesichtersammlung nach Übereinstimmungen durchsuchen. Weitere Informationen finden Sie unter Hinzufügen von Gesichtern zu einer Sammlung.
- Die ListFaces-Operation listet die Gesichter in einer Sammlung auf. Weitere Informationen finden Sie unter Hinzufügen von Gesichtern zu einer Sammlung.
- Die DeleteFaces-Operation löscht die Gesichter aus einer Sammlung. Weitere Informationen finden Sie unter Löschen von Gesichtern aus einer Sammlung.

Benutzer in einer Sammlung verwalten

Nachdem Sie mehrere Gesichtsvektoren von derselben Person gespeichert haben, können Sie die Genauigkeit verbessern, indem Sie all diese Gesichtsvektoren zu einem Benutzervektor verknüpfen. Sie können die folgenden Operationen verwenden, um Ihre Benutzer zu verwalten:

- CreateUser- Der Vorgang erstellt einen neuen Benutzer in einer Sammlung mit einer angegebenen eindeutigen Benutzer-ID.
- AssociateUsers- Fügt einer Benutzer-ID ein IDs bis 100 eindeutiges Gesicht hinzu. Nachdem Sie einem Benutzer mindestens eine Gesichts-ID zugeordnet haben, können Sie in Ihrer Sammlung nach Übereinstimmungen mit diesem Benutzer suchen.
- ListUsers- Listet die Benutzer in einer Sammlung auf.
- DeleteUsers- Löscht einen Benutzer aus einer Sammlung mit der angegebenen Benutzer-ID.
- DisassociateFaces- Entfernt ein oder mehrere Gesichter IDs von einem Benutzer.

Verwendung von Ähnlichkeitsschwellenwerten für die Zuordnung und Zuordnung von Gesichtern

Ähnlichkeitsschwellenwerte werden sowohl für das Zuordnen als auch für das Zuordnen von Gesichtern verwendet. Im Folgenden finden Sie Anleitungen zur Verwendung von Ähnlichkeitsschwellenwerten für beide Anwendungsfälle.

Verwendung von Ähnlichkeitsschwellenwerten für die Zuordnung von Gesichtern

Beim Zuordnen von Gesichtern mithilfe dieser AssociateFacesOperation ist es wichtig sicherzustellen, dass Gesichter, die einem Benutzer zugeordnet werden, alle von derselben Person stammen. Als Hilfestellung gibt der `UserMatchThreshold`-Parameter die Mindestzuverlässigkeit der Benutzerübereinstimmung an, die erforderlich ist, damit das neue Gesicht einem `UserID` zugeordnet werden kann, das bereits mindestens ein `FaceID` enthält. Auf diese Weise wird sichergestellt, dass die `FaceIds` dem richtigen `UserID` zugeordnet werden. Der Wert liegt zwischen 0 und 100 und der Standardwert ist 75.

Verwenden von Übereinstimmungsschwellenwerten für den Gesichterabgleich

Wir ermöglichen es Ihnen, die Ergebnisse aller Suchoperationen (CompareFaces,,,, SearchUsersByImage) zu kontrollieren SearchFacesSearchFacesByImageSearchUsers, indem Sie einen Ähnlichkeitsschwellenwert als Eingabeparameter angeben.

`FaceMatchThreshold` ist das Eingabeattribut für den Ähnlichkeitsschwellenwert für `SearchFaces` sowie `SearchFacesByImage` und steuert, wie viele Ergebnisse auf der Grundlage der Ähnlichkeit mit dem übereinstimmenden Gesicht zurückgegeben werden. Das Attribut für den Ähnlichkeitsschwellenwert für `SearchUsers` und `SearchUsersByImage` ist `UserMatchThreshold`. Es steuert, wie viele Ergebnisse auf der Grundlage der Ähnlichkeit mit dem zu übereinstimmenden Benutzervektor zurückgegeben werden. Das Schwellenwertattribut ist `SimilarityThreshold` für `CompareFaces`.

Antworten mit einem `Similarity`-Antwortattributwert, der niedriger als der Schwellenwert ist, werden nicht zurückgegeben. Es ist wichtig, diesen Schwellenwert für Ihren Anwendungsfall zu kalibrieren, da mit ihm bestimmt werden kann, wie viele Falschmeldungen im Ihren Übereinstimmungsergebnissen enthalten sind. Dies steuert die Wiedererkennung Ihrer Suchergebnisse. Je niedriger der Schwellenwert, desto höher die Wiedererkennung.

Alle Machine Learning-Systeme sind probabilistisch. Nehmen Sie die Einstellung des richtigen Übereinstimmungsschwellenwerts je nach Anwendungsfall nach eigenem Ermessen vor. Wenn Sie z. B. eine Foto-App zur Ermittlung ähnlich aussehender Familienmitglieder entwickeln wollen, könnten Sie einen niedrigeren Grenzwert wählen (z. B. 80 %). Für viele Anwendungsfälle in der Strafverfolgung empfehlen wir hingegen, einen hohen Schwellenwert von 99 % oder höher anzusetzen, um Fehlidentifikationen zu vermeiden.

Zusätzlich zu `FaceMatchThreshold` und `UserMatchThreshold` können Sie das `Similarity`-Antwortattribut als Mittel einsetzen, um Fehlidentifikationen zu verringern. Beispielsweise können Sie einen niedrigen Grenzwert (z. B. 80 %) auswählen, um mehr Ergebnisse zurückgeben. Anschließend verwenden Sie das Antwortattribut „Ähnlichkeit“ (Übereinstimmung in %) zum Einschränken der Auswahl und zum Filtern der richtigen Antworten in Ihrer Anwendung. Auch hier verringert ein höherer Übereinstimmungswert (z. B. 99 % und höher) die Gefahr der Fehlidentifikation.

Anleitung zur Indexierung von Gesichtern in gängigen Szenarien

Im Folgenden finden Sie einen Leitfaden für die Verwendung von `IndexFaces` in geläufigen Szenarien.

Kritische oder die öffentliche Sicherheit betreffende Anwendungen

- Rufen Sie IndexFaces bei Bildern auf, auf denen sich nur jeweils ein Gesicht befindet, und verknüpfen Sie die zurückgegebene Gesichts-ID mit der Kennung der aufgenommenen Person des Bildes.
- Sie können mithilfe von DetectFaces vor der Indizierung prüfen, dass sich auf dem Bild nur ein Gesicht befindet. Wenn mehr als ein Gesicht erkannt wird, übermitteln Sie das Bild nach einer Überprüfung mit nur einem vorhandenen Gesicht erneut. Dadurch wird verhindert, dass versehentlich mehrere Gesichter indiziert und derselben Person zugeordnet werden.

Anwendungen für Bildaustausch und soziale Medien

- Sie sollten `IndexFaces` ohne Einschränkungen für Bilder aufrufen, die bei Anwendungsfällen wie z B. Familienalben mehrere Gesichter enthalten. In solchen Fällen müssen Sie jede Person in jedem Foto identifizieren und diese Informationen dann zum Gruppieren von Fotos nach den auf ihnen enthaltenen Personen verwenden.

Allgemeine Nutzung

- Indizieren Sie mehrere verschiedene Bilder derselben Person, insbesondere mit unterschiedlichen Gesichtsattributen (Gesichtsposen, Gesichtsbehaarung usw.), erstellen Sie einen Benutzer und ordnen Sie die verschiedenen Gesichter diesem Benutzer zu, um die Abgleichqualität zu verbessern.
- Schließen Sie einen Überprüfungsvorgang ein, sodass fehlerhafte Übereinstimmungen mit der korrekten Gesichtskennung indiziert werden können, um die Fähigkeit zum Abgleichen von Gesichtern nachfolgend zu verbessern.
- Weitere Informationen zur Bildqualität finden Sie unter Empfehlungen zu Eingabebildern für den Gesichtsvergleich.

Suche nach Gesichtern und Benutzern in einer Sammlung

Nachdem Sie eine Gesichtssammlung erstellt und Gesichtsvektoren und/oder Benutzervektoren gespeichert haben, können Sie eine Gesichtssammlung nach Gesichtsübereinstimmungen durchsuchen. Mit Amazon Rekognition können Sie übereinstimmende Gesichter in einer Sammlung suchen:

- Eine bereitgestellte Gesichts-ID (SearchFaces). Weitere Informationen finden Sie unter Suche nach einem Gesicht mit einer Gesichts-ID.
- Das größte Gesicht in einem bereitgestellten Bild (SearchFacesByImage). Weitere Informationen finden Sie unter Suche nach einem Gesicht mit einem Bild.
- Gesichter in einem gespeicherten Video. Weitere Informationen finden Sie unter Suche nach Gesichtern in gespeicherten Videos.
- Gesichter in einem Streaming-Video. Weitere Informationen finden Sie unter Arbeiten mit Streaming-Videoereignissen.

Sie können die `CompareFaces`/Operation verwenden, um ein Gesicht in einem Quellbild mit Gesichtern im Zielbild zu vergleichen. Dieser Vergleich ist auf die auf dem Zielbild erkannten Gesichter begrenzt. Weitere Informationen finden Sie unter Vergleich von Gesichtern in Bildern.

Die verschiedenen Suchoperationen in der folgenden Liste vergleichen ein Gesicht (das entweder durch eine `FaceId` oder ein Eingabebild identifiziert wird) mit allen Gesichtern, die in einer bestimmten Gesichtssammlung gespeichert sind:

- SearchFaces
- SearchFacesByImage
- SearchUsers
- SearchUsersByImage

Anwendungsfälle, die die öffentliche Sicherheit betreffen

Zusätzlich zu den Empfehlungen in Bewährte Methoden für Sensoren, Eingabebilder und Videos und Anleitung zur Indexierung von Gesichtern in gängigen Szenarien sollten Sie die folgenden bewährten Methoden bei der Bereitstellung von Gesichtserkennungs- und -vergleichssystemen in Anwendungsfällen beachten, die die öffentliche Sicherheit betreffen. Erstens sollten Sie Vertrauensschwellen von 99 % oder höher verwenden, um Fehler und Falschmeldungen zu

reduzieren. Zweitens sollten Sie veranlassen, dass die mit einem Gesichtserkennungs- oder -vergleichssystem erzielten Ergebnisse von einem menschlichen Prüfer überprüft werden. Sie sollten ohne zusätzliche menschliche Überprüfung keine Entscheidungen basierend auf der Systemausgabe treffen. Gesichtserkennungs- und -vergleichssysteme dienen als Hilfsmittel zur Eingrenzung möglicher Übereinstimmungen, die dann von Menschen rasch überprüft und in Betracht gezogen werden können. Drittens wird in Bezug auf die Verwendung von Gesichtserkennungs- und -vergleichssystemen in diesen Anwendungsfällen Transparenz empfohlen. Dazu gehört auch, Endbenutzer und aufgenommene Personen über die Verwendung dieser Systeme zu informieren, Einverständnis für eine solche Nutzung einzuholen und eine Methode bereitzustellen, mit der Endbenutzer und aufgenommene Personen Feedback zur Verbesserung des Systems geben können.

Wenn Sie eine Strafverfolgungsbehörde sind, die die Amazon-Rekognition-Gesichtsvergleichsfunktion im Zusammenhang mit strafrechtlichen Ermittlungen verwendet, müssen Sie die in den AWS -Servicebedingungen aufgeführten Anforderungen erfüllen. Diese beinhalten Folgendes:

- Lassen Sie alle Entscheidungen von ausreichend ausgebildeten Personen überprüfen, wenn es um Maßnahmen geht, die die bürgerlichen Freiheiten oder gleichwertige Menschenrechte einer Person beeinträchtigen könnten.
- Schulen Sie Ihr Personal für den verantwortungsbewussten Einsatz von Gesichtserkennungssystemen.
- Legen Sie Ihre Nutzung von Gesichtserkennungssystemen in der Öffentlichkeit offen.
- Verwenden Sie Amazon Rekognition nicht für eine dauerhafte Überwachung einer Person ohne unabhängige Überprüfung oder zwingende Umstände.

In allen Fällen müssen die Übereinstimmungen des Gesichtsvergleichs zusätzlich anderweitig belegt werden und sollten nicht als einzige Bestimmungsfaktoren verwendet werden. Wenn der Gesichtsvergleich jedoch für non-law-enforcement Szenarien verwendet wird (z. B. um ein Telefon zu entsperren oder die Identität eines Mitarbeiters für den Zugang zu einem sicheren, privaten Bürogebäude zu authentifizieren), müssten diese Entscheidungen nicht manuell geprüft werden, da sie sich nicht auf die bürgerlichen Freiheiten einer Person oder vergleichbare Menschenrechte auswirken würden.

Wenn Sie vorhaben, ein Gesichtserkennungs- oder -vergleichssystem für Anwendungsfälle zu verwenden, die die öffentliche Sicherheit betreffen, sollten Sie die bereits erwähnten bewährten Methoden einsetzen. Darüber hinaus sollten Sie veröffentlichte Ressourcen bezüglich der

Verwendung des Gesichtsvergleichs einsehen. Dazu gehört die Veröffentlichung Face Recognition Policy Development Template For Use In Criminal Intelligence and Investigative Activities des Bureau of Justice Assistance des Department of Justice. Die Vorlage stellt mehrere auf Gesichtsvergleich und Biometrie bezogene Ressourcen zur Verfügung und soll Vollstreckungsbehörden und öffentlichen Sicherheitsbehörden als Leitfaden bei der Entwicklung von Richtlinien für den Gesichtsvergleich dienen, die den geltenden Gesetzen entsprechen, Datenschutzrisiken reduzieren und Rechenschaftspflicht und Beaufsichtigung von Entitäten einrichten. Zu weiteren Ressourcen gehören die Veröffentlichungen Best Privacy Practices for Commercial Use of Facial Recognition von der National Telecommunications and Information Administration und Best Practices for Common Uses of Facial Recognition von den Mitarbeitern der Federal Trade Commission. Zukünftig werden möglicherweise weitere Ressourcen entwickelt und veröffentlicht. Sie sollten sich fortwährend zu diesem wichtigen Thema weiterbilden.

Zur Erinnerung: Sie müssen sämtliche geltenden Gesetze bei der Anwendung von AWS-Services beachten und Sie dürfen keinen AWS-Service auf eine Art und Weise verwenden, durch die die Rechte anderer verletzt werden oder die für andere schädlich sein könnte. Dies bedeutet, dass Sie AWS-Services für die öffentliche Sicherheit betreffende Anwendungsfälle nicht auf eine Art und Weise verwenden dürfen, durch die eine Person rechtswidrig benachteiligt wird oder faire Prozesse, Privatsphäre oder bürgerliche Freiheiten einer Person verletzt werden. Sie sollten entsprechende rechtliche Auskünfte einholen, wenn dies zur Überprüfung der gesetzlichen Anforderungen oder bei Fragen zu Ihrem Anwendungsfall erforderlich ist.

Verwendung von Amazon Rekognition zur Unterstützung der öffentlichen Sicherheit

Amazon Rekognition kann für Aufgaben in Szenarios der öffentlichen Sicherheit und Strafverfolgung herangezogen werden, z. B. bei der Suche vermisster Kinder, dem Kampf gegen den Menschenhandel oder zur Verhinderung von Straftaten. Beachten Sie bei Szenarios der öffentlichen Sicherheit und Strafverfolgung folgende Hinweise:

- Verwenden Sie Amazon Rekognition als ersten Schritt bei der Suche nach möglichen Übereinstimmungen. Durch die Antworten von Gesichtsoperationen von Amazon Rekognition erhalten Sie schnell eine Reihe von potenziellen Übereinstimmungen zur weiteren Überprüfung.
- Treffen Sie in Szenarios, die eine menschliche Analyse erfordern, keine autonomen Entscheidungen auf Grundlage von Amazon-Rekognition-Antworten. Wenn Sie eine Strafverfolgungsbehörde sind, die mithilfe von Amazon Rekognition bei der Identifizierung einer Person im Zusammenhang mit einer strafrechtlichen Untersuchung hilft, und Maßnahmen auf

der Grundlage der Identifizierung ergriffen werden, die sich auf die bürgerlichen Freiheiten oder die gleichwertigen Menschenrechte dieser Person auswirken könnten, muss die Entscheidung für solche Maßnahmen, von einer entsprechend geschulten Person auf der Grundlage ihrer unabhängigen Prüfung des Identifikationsnachweises getroffen werden.

- Verwenden Sie für Szenarios, in denen eine äußerst genaue Übereinstimmung notwendig ist, einen Ähnlichkeitsschwellenwert von 99 %. Ein Beispiel hierfür ist die Authentifizierung, um Zutritt zu einem Gebäude zu erhalten.
- Wenn Grundrechte betroffen sind, wie bei Anwendungsfällen, bei denen Strafverfolgungsbehörden involviert sind, sollten Sie Vertrauensschwellen von 99 % oder höher verwenden und Gesichtsvergleichsprognosen von Menschen prüfen lassen, um sicherzustellen, dass keine Grundrechte verletzt werden.
- Verwenden Sie einen Ähnlichkeitsschwellenwert von unter 99 % für Szenarios, bei denen eine höhere Anzahl von potenziellen Übereinstimmungen hilfreich ist. Das ist zum Beispiel bei der Suche nach vermissten Personen der Fall. Falls erforderlich, verwenden Sie das Similarity-Antwortattribut, um zu bestimmen, wie viel Ähnlichkeit potenzielle Übereinstimmungen mit der Person haben, die Sie erkennen möchten.
- Planen Sie Ihr Vorgehen im Falle von falschpositiven Gesichtsübereinstimmungen, die von Amazon Rekognition zurückgegeben werden. Beispiel: Verbessern Sie den Abgleich, indem Sie mehrere Bilder von derselben Person bei der Indexerstellung mithilfe der IndexFaces-Operation verwenden. Weitere Informationen finden Sie unter Anleitung zur Indexierung von Gesichtern in gängigen Szenarien.

In anderen Anwendungsfälle (z. B. soziale Medien) empfehlen wir, nach eigenem Ermessen zu beurteilen, ob für die Amazon-Rekognition-Ergebnisse eine menschliche Überprüfung nötig ist. Außerdem kann je nach den Anforderungen Ihrer Anwendung der Ähnlichkeitsschwellenwert niedriger sein.

Erstellen einer Sammlung

Sie können die Operation CreateCollection zum Erstellen einer Sammlung verwenden.

Weitere Informationen finden Sie unter Eine Sammlung verwalten.

So erstellen Sie eine Sammlung (SDK)

1. Wenn Sie dies noch nicht getan haben:

a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

b. Installieren und konfigurieren Sie das und das AWS CLI . AWS SDKs Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `CreateCollection`-Operation.

   Java

   Im folgenden Beispiel wird eine Sammlung erstellt und der Amazon-Ressourcenname (ARN) angezeigt.

   Ändern Sie den Wert von `collectionId` in den Namen der Sammlung, die Sie erstellen möchten.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
//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

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `create-collection` CLI-Vorgang an.

Ersetzen Sie den Wert von `collection-id` durch den Namen der Sammlung, die Sie erstellen möchten.

Ersetzen Sie den Wert von `profile_name` mit dem Namen Ihres Entwicklerprofils.

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

Python

Im folgenden Beispiel wird eine Sammlung erstellt und der Amazon-Ressourcenname (ARN) angezeigt.

Ändern Sie den Wert von `collection_id` in den Namen der Sammlung, die Sie erstellen möchten. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
# Copyright 2018 Amazon.com, Inc. or its 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

Im folgenden Beispiel wird eine Sammlung erstellt und der Amazon-Ressourcenname (ARN) angezeigt.

Ändern Sie den Wert von `collectionId` in den Namen der Sammlung, die Sie erstellen möchten.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Ersetzen Sie im folgenden Beispiel den Wert von `region` durch den Namen der Region, die mit Ihrem Konto verknüpft ist, und ersetzen Sie den Wert von `collectionName` durch den gewünschten Namen Ihrer Sammlung.

Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
//Copyright 2018 Amazon.com, Inc. or its 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 Operationsanforderung

Bei der Eingabe in `CreationCollection` handelt es sich um den Namen der Sammlung, die Sie erstellen möchten.

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

CreateCollection Antwort auf die Operation

Amazon Rekognition erstellt die Sammlung und gibt den Amazon-Ressourcennamen (ARN) der neu erstellten Sammlung zurück.

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

Markieren von Sammlungen

Sie können Amazon-Rekognition-Sammlungen mithilfe von Tags identifizieren, organisieren, suchen und filtern. Jedes Tag ist ein Label, das aus einem benutzerdefinierten Schlüssel und Wert besteht.

Sie können Tags auch verwenden, um den Zugriff auf eine Sammlung mithilfe von Identity and Access Management (IAM) zu steuern. Weitere Informationen finden Sie unter Steuern des Zugriffs auf AWS Ressourcen mithilfe von Ressourcen-Tags.

Themen

- Hinzufügen von Tags zu einer neuen Sammlung
- Hinzufügen von Tags zu einer vorhandenen Sammlung
- Stichwörter in einer Sammlung auflisten
- Löschen von Tags aus einer Sammlung

Hinzufügen von Tags zu einer neuen Sammlung

Sie können Tags zu einer Sammlung hinzufügen, während Sie sie mit der `CreateCollection` Operation erstellen. Geben Sie ein oder mehrere Tags im `Tags`-Array-Eingabeparameter an.

AWS CLI

Ersetzen Sie den Wert von `profile_name` mit dem Namen Ihres Entwicklerprofils.

```
aws rekognition create-collection --collection-id "collection-name" --tags
 "{"key1":"value1","key2":"value2"}" --profile profile-name
```

Für Windows-Geräte:

```
aws rekognition create-collection --collection-id "collection-name" --tags
 "{\"key1\":\"value1\",\"key2\":\"value2\"}" --profile profile-name
```

Python

Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
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()
```

Hinzufügen von Tags zu einer vorhandenen Sammlung

Um eine oder mehrere Tags zu einer bestehenden Sammlung hinzuzufügen, verwenden Sie die `TagResource`-Operation. Geben Sie den Amazon-Ressourcennamen (ARN) (`ResourceArn`) der Sammlung und die Tags (`Tags`) an, die Sie hinzufügen möchten. Das folgende Beispiel zeigt, wie zwei Tags hinzugefügt werden.

AWS CLI

Ersetzen Sie den Wert von `profile_name` mit dem Namen Ihres Entwicklerprofils.

```
aws rekognition tag-resource --resource-arn collection-arn --tags
 "{"key1":"value1","key2":"value2"}" --profile profile-name
```

Für Windows-Geräte:

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

Python

Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
# Copyright 2018 Amazon.com, Inc. or its 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**
>
> Wenn Sie den Amazon-Ressourcennamen der Sammlung nicht kennen, können Sie die `DescribeCollection`-Operation verwenden.

Stichwörter in einer Sammlung auflisten

Um die an eine Sammlung angehängten Tags aufzulisten, verwenden Sie die `ListTagsForResource`-Operation und geben Sie den ARN der Sammlung (`ResourceArn`) an. Die Antwort ist eine Zuordnung von Tag-Schlüsseln und -Werten, die mit der angegebenen Sammlung verknüpft sind.

AWS CLI

Ersetzen Sie den Wert von `profile_name` mit dem Namen Ihres Entwicklerprofils.

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

Python

Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
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()
```

Die Ausgabe zeigt eine Liste der an die Sammlung angehängten Tags:

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

Löschen von Tags aus einer Sammlung

Verwenden Sie die `UntagResource`-Operation, um mindestens ein Tag aus einer Sammlung zu entfernen. Geben Sie den ARN des Modells (`ResourceArn`) und die Tag-Schlüssel (`Tag-Keys`) an, die Sie entfernen möchten.

AWS CLI

Ersetzen Sie den Wert von `profile_name` mit dem Namen Ihres Entwicklerprofils.

```
aws rekognition untag-resource --resource-arn resource-arn --profile profile-name --
tag-keys "key1" "key2"
```

Alternativ können Sie Tag-Schlüssel in diesem Format angeben:

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

Python

Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
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()
```

Auflisten von Sammlungen

Sie können die ListCollections-Operation zum Auflisten der Sammlungen in der Region verwenden, die Sie verwenden.

Weitere Informationen finden Sie unter Eine Sammlung verwalten.

Sie können Sammlungen auflisten (SDK)

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

b. Installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `ListCollections`-Operation.

Java

Im folgenden Beispiel sind die Sammlungen in der aktuellen Region aufgelistet.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
//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

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `list-collections` CLI-Vorgang an. Ersetzen Sie den Wert von `profile_name` mit dem Namen Ihres Entwicklerprofils.

```
aws rekognition list-collections --profile profile-name
```

Python

Im folgenden Beispiel sind die Sammlungen in der aktuellen Region aufgelistet.

Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
#Copyright 2018 Amazon.com, Inc. or its 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

Im folgenden Beispiel sind die Sammlungen in der aktuellen Region aufgelistet.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
//Copyright 2018 Amazon.com, Inc. or its 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 Operationsanforderung

Bei der Eingabe in `ListCollections` handelt es sich um die maximale Anzahl der Sammlungen, die zurückgegeben werden sollen.

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

Wenn die Antwort mehr Sammlungen als von `MaxResults` angefordert aufweist, wird ein Token zurückgegeben, das Sie verwenden können, um den nächsten Ergebnissatz in einem nachfolgenden Aufruf an `ListCollections` abzurufen. Zum Beispiel:

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

ListCollections Antwort auf die Operation

Amazon Rekognition gibt eine Reihe von Sammlungen zurück (`CollectionIds`). Ein separates Array (`FaceModelVersions`) enthält die Version des Gesichtsmodells, die zum Analysieren der Gesichter in der jeweiligen Sammlung verwendet wird. Beispiel: In der folgenden JSON-Antwort analysiert die Sammlung `MyCollection` Gesichter mit Version 2.0 des Gesichtsmodells. Die Sammlung `AnotherCollection` nutzt Version 3.0 des Gesichtsmodells. Weitere Informationen finden Sie unter Grundlegendes zur Modellversionierung.

`NextToken` ist das Token, das für den Abruf des nächsten Ergebnissatzes in einem nachfolgenden Aufruf an `ListCollections` verwendet wird.

```
{
    "CollectionIds": [
        "MyCollection",
        "AnotherCollection"
    ],
```

```
    "FaceModelVersions": [
        "2.0",
        "3.0"
    ],
    "NextToken": "MGYZLAHX1T5a...."
}
```

Beschreiben einer Sammlung

Sie können die DescribeCollection-Operation verwenden, um die folgenden Informationen über eine Sammlung abzurufen:

- Die Anzahl der Gesichter, die in den Index der Sammlung aufgenommen werden.
- Die Version des Modells, die mit der Sammlung verwendet wird. Weitere Informationen finden Sie unter the section called “Grundlegendes zur Modellversionierung”.
- Der Amazon-Ressourcenname (ARN) der Sammlung
- Das Erstellungsdatum und der Erstellungszeitpunkt der Sammlung

So beschreiben Sie eine Sammlung (SDK)

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `DescribeCollection`-Operation.

   Java

   In diesem Beispiel wird eine Sammlung beschrieben.

   Ändern Sie den Wert von `collectionId` in die ID der gewünschten Sammlung.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
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

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `describe-collection` CLI-Vorgang an. Ändern Sie den Wert von `collection-id` in die ID der gewünschten Sammlung. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
aws rekognition describe-collection --collection-id collection-name --profile
 profile-name
```

Python

In diesem Beispiel wird eine Sammlung beschrieben.

Ändern Sie den Wert von `collection_id` in die ID der gewünschten Sammlung. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
# Copyright 2018 Amazon.com, Inc. or its 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

In diesem Beispiel wird eine Sammlung beschrieben.

Ändern Sie den Wert von `collectionId` in die ID der gewünschten Sammlung.

```
//Copyright 2018 Amazon.com, Inc. or its 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 Operationsanforderung

Bei der Eingabe in `DescribeCollection` handelt es sich um die ID der gewünschten Sammlung, wie im folgenden JSON-Beispiel gezeigt.

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

DescribeCollectionAntwort auf die Operation

Die Antwort enthält:

- Die Anzahl der Gesichter, die in den Index der Sammlung aufgenommen werden, `FaceCount`.
- Die Version des Gesichtsmodells, die mit der Sammlung verwendet wird, `FaceModelVersion`. Weitere Informationen finden Sie unter the section called “Grundlegendes zur Modellversionierung”.
- Der Amazon-Ressourcenname (ARN) der Sammlung, `CollectionARN`.
- Die Erstellungszeit und das Erstellungsdatum der Sammlung, `CreationTimestamp`. Der Wert von `CreationTimestamp` ist die Anzahl der Millisekunden seit der Unix-Epoche bis zur Erstellung der Sammlung. Die Unix-Epochenzeit ist Donnerstag, 1. Januar 1970, 00:00:00 Uhr in UTC (Coordinated Universal Time). Weitere Informationen finden Sie unter Unix Time.

```
{
    "CollectionARN": "arn:aws:rekognition:us-east-1:nnnnnnnnnnnn:collection/
MyCollection",
    "CreationTimestamp": 1.533422155042E9,
    "FaceCount": 200,
    "UserCount" : 20,
    "FaceModelVersion": "1.0"
}
```

Löschen einer Sammlung

Sie können die Operation DeleteCollection zum Löschen einer Sammlung verwenden.

Weitere Informationen finden Sie unter Eine Sammlung verwalten.

So löschen Sie eine Sammlung (SDK)

1. Wenn Sie dies noch nicht getan haben:
   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.
   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.
2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `DeleteCollection`-Operation.

   Java

   In diesem Beispiel wird eine Sammlung gelöscht.

   Ändern Sie den Wert `collectionId` in die Sammlung, die Sie löschen möchten.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
// 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

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `delete-collection` CLI-Vorgang an. Ersetzen Sie den Wert von `collection-id` durch den Namen der Sammlung, die Sie löschen möchten. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
aws rekognition delete-collection --collection-id collection-name --profile
 profile-name
```

Python

In diesem Beispiel wird eine Sammlung gelöscht.

Ändern Sie den Wert `collection_id` in die Sammlung, die Sie löschen möchten. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
# Copyright 2018 Amazon.com, Inc. or its 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

In diesem Beispiel wird eine Sammlung gelöscht.

Ändern Sie den Wert `collectionId` in die Sammlung, die Sie löschen möchten.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
//Copyright 2018 Amazon.com, Inc. or its 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 Operationsanforderung

Bei der Eingabe in `DeleteCollection` handelt es sich um die ID der Sammlung, die gelöscht werden soll, wie im folgenden JSON-Beispiel gezeigt.

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

DeleteCollection Antwort auf die Operation

Die `DeleteCollection`-Antwort enthält einen HTTP-Statuscode, der angibt, ob die Operation erfolgreich ausgeführt wurde. `200` wird zurückgegeben, wenn die Sammlung erfolgreich gelöscht wurde.

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

Hinzufügen von Gesichtern zu einer Sammlung

Sie können die Operation IndexFaces verwenden, um Gesichter in einem Bild zu erkennen und um sie einer Sammlung hinzuzufügen. Für jedes erkannte Gesicht extrahiert Amazon Rekognition

die Gesichtszüge und speichert die Gesichtsmerkmalinformationen in einer Datenbank. Zusätzlich speichert der Befehl Metadaten für jedes erkannte Gesicht in einer festgelegten Gesichtersammlung. Amazon Rekognition speichert nicht die tatsächlichen Bildbytes.

Weitere Informationen zur Bereitstellung geeigneter Gesichter für die Indizierung finden Sie unter Empfehlungen zu Eingabebildern für den Gesichtsvergleich.

Für jedes Gesicht erhält die `IndexFaces`-Operation die folgenden Informationen:

- Multidimensionale Gesichtszüge – `IndexFaces`Durch eine Gesichtsanalyse werden multidimensionale Informationen über die Gesichtsmerkmale extrahiert und in der Gesichtersammlung gespeichert. Sie haben keinen direkten Zugriff auf diese Informationen. Allerdings nutzt Amazon Rekognition diese Informationen bei der Prüfung einer Gesichtersammlung auf Übereinstimmungen.

- Metadaten — Zu den Metadaten für jedes Gesicht gehören ein Begrenzungsrahmen, ein Konfidenzniveau (dass das Begrenzungsfeld ein Gesicht enthält), das von Amazon Rekognition IDs zugewiesen wurde (Gesichts-ID und Bild-ID), und eine externe Bild-ID (falls Sie sie angegeben haben) in der Anfrage. Diese Informationen werden als Antwort auf den `IndexFaces`-API-Aufruf zurückgegeben. Ein Beispiel sehen Sie im `face`-Element in der nächsten Beispielantwort.

  Der Dienst gibt diese Metadaten als Antwort auf die folgenden API-Aufrufe zurück:

  - `ListFaces`
  - Gesichtssuche-Operationen – Die Antworten für die Funktionen SearchFaces und SearchFacesByImage liefern den Zuverlässigkeitswert für jede Gesichtsübereinstimmung sowie die Metadaten des übereinstimmenden Gesichts.

Die Anzahl der mit `IndexFaces` indizierten Gesichter hängt von der Version des Geischtserkennungsmodells ab, die der Eingabesammlung zugeordnet ist. Weitere Informationen finden Sie unter Grundlegendes zur Modellversionierung.

Informationen über indizierte Gesichter werden in einem Array von FaceRecord Objekten zurückgegeben.

Sie können indizierte Gesichter mit dem Bild verknüpfen, in dem sie erkannt wurden. Beispielsweise können Sie einen kundenseitigen Index von Bildern und Gesichtern in den Bildern pflegen.

Um Gesichter mit einem Bild zu verknüpfen, geben Sie eine Bild-ID als Abfrageparameter `ExternalImageId` an. Die Bild-ID kann der Dateiname oder eine andere ID sein, die Sie erstellen.

Zusätzlich zu den vorangegangenen Informationen, die die API in der Gesichtersammlung erhält, gibt die API auch Einzelheiten zum Gesicht zurück, die nicht in der Sammlung behalten werden. (Sehen Sie sich das `faceDetail`-Element in der folgenden Beispielantwort an).

> **Note**
>
> `DetectFaces` gibt die gleiche Information zurück. Sie müssen daher nicht jeweils einen Aufruf mit `DetectFaces` und `IndexFaces` für das gleiche Bild durchführen.

Filtern von Gesichtern

Mit diesem IndexFaces Vorgang können Sie die Gesichter filtern, die anhand eines Bildes indexiert wurden. Mit `IndexFaces` können Sie eine maximale Anzahl an Gesichtern zum Indizieren angeben, oder Sie können wählen, ob nur Gesichter indiziert werden sollen, bei denen eine hohe Qualität erkannt wurde.

Sie können die maximale Anzahl an Gesichtern angeben, die von `IndexFaces` indiziert werden, indem Sie den Eingabeparameter `MaxFaces` verwenden. Dies ist nützlich, wenn Sie möchten, dass nur die größten Gesichter in einem Bild indiziert werden sollen, aber keine kleineren Gesichter, z. B. solche von Personen, die im Hintergrund stehen.

Standardmäßig wählt `IndexFaces` eine Qualitätsleiste zum Filtern von Gesichtern aus. Mit dem `QualityFilter`-Eingabeparameter können Sie die Qualitätsleiste explizit festzulegen. Die Werte lauten:

- `AUTO` – Amazon Rekognition wählt die Qualitätsleiste aus, die zum Filtern von Gesichtern verwendet wird (Standardwert).
- `LOW` – Alle Gesichter mit Ausnahme derjenigen mit niedrigster Qualität werden indiziert.
- `MEDIUM`
- `HIGH` – Es werden nur Gesichter von höchster Qualität indiziert.
- `NONE` – Es werden keine Gesichter aufgrund der Qualität gefiltert.

`IndexFaces` filtert Gesichter aus folgenden Gründen:

- Das Gesicht ist im Verhältnis zu den Bildmaßen zu klein.
- Das Gesicht ist zu unscharf.
- Das Bild ist zu dunkel.
- Das Gesicht hat eine extreme Pose.
- Das Gesicht aufgrund zu weniger Details nicht für die Gesichtssuche geeignet.

> **Note**
>
> Zur Verwendung der Filterung nach Qualität ist eine mit mindestens Version 3 des Gesichtsmodells verknüpfte Sammlung erforderlich. Um die Version des Gesichtsmodells anzuzeigen, die mit einer Sammlung verknüpft ist, rufen Sie DescribeCollection auf.

Informationen über Gesichter, die nicht von `IndexFaces` indiziert sind, werden in einem Array von UnindexedFace Objekten zurückgegeben. Das Array `Reasons` enthält eine Liste von Gründen, warum ein Gesicht nicht indiziert ist. Beispielsweise bezeichnet der Wert `EXCEEDS_MAX_FACES` ein Gesicht, das nicht indiziert ist, weil die Anzahl der durch `MaxFaces` angegebenen Gesichter bereits erkannt worden ist.

Weitere Informationen finden Sie unter Verwalten von Gesichtern in einer Sammlung.

Hinzufügen von Gesichtern zu einer Sammlung (SDK)

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`- und `AmazonS3ReadOnlyAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Laden Sie ein Bild (mit einem oder mehreren Gesichtern) auf Ihren Amazon-S3-Bucket hoch.

   Weitere Anleitungen finden Sie unter Upload eines Objekts in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service.

3. Verwenden Sie die folgenden Beispiele zum Aufrufen der `IndexFaces`-Operation.

Java

Dieses Beispiel zeigt die Identifikatoren für Gesichter an, die zur Sammlung hinzugefügt wurden.

Ändern Sie den Wert von `collectionId` in den Namen der Sammlung, die Sie einem Gesicht hinzufügen möchten. Ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, das Sie in Schritt 2 verwendet haben. Der Parameter `.withMaxFaces(1)` beschränkt die Anzahl der indizierten Gesichter auf 1. Entfernen oder ändern Sie diesen Werten entsprechend Ihren Anforderungen.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
//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

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `index-faces` CLI-Vorgang an.

Ersetzen Sie den Wert von `collection-id` durch den Namen der Sammlung, in der das Gesicht gespeichert werden soll. Ersetzen Sie die Werte von `Bucket` und `Name` durch den Amazon-S3-Bucket und dem Bilddateinamen, die Sie in Schritt 2 verwendet haben. Der Parameter `max-faces` beschränkt die Anzahl der indizierten Gesichter auf 1. Entfernen oder ändern Sie diesen Werten entsprechend Ihren Anforderungen. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
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
```

Wenn Sie auf einem Windows-Gerät auf die CLI zugreifen, verwenden Sie doppelte Anführungszeichen anstelle von einfachen Anführungszeichen und maskieren Sie die inneren doppelten Anführungszeichen durch einen Backslash (d. h. \), um eventuell auftretende Parserfehler zu beheben. Sehen Sie sich zum Beispiel Folgendes an:

```
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

Dieses Beispiel zeigt die Identifikatoren für Gesichter an, die zur Sammlung hinzugefügt wurden.

Ändern Sie den Wert von `collectionId` in den Namen der Sammlung, die Sie einem Gesicht hinzufügen möchten. Ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, das Sie in Schritt 2 verwendet haben. Der Eingabeparameter `MaxFaces` beschränkt die Anzahl der indizierten Gesichter auf 1. Entfernen oder ändern Sie diesen Werten entsprechend Ihren Anforderungen. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

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

import boto3

def add_faces_to_collection(bucket, photo, collection_id):

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

    response = client.index_faces(CollectionId=collection_id,
                                  Image={'S3Object': {'Bucket': bucket, 'Name':
 photo}},
                                  ExternalImageId=photo,
                                  MaxFaces=1,
                                  QualityFilter="AUTO",
                                  DetectionAttributes=['ALL'])

    print('Results for ' + photo)
    print('Faces indexed:')
```

```
    for faceRecord in response['FaceRecords']:
        print('  Face ID: ' + faceRecord['Face']['FaceId'])
        print('  Location: {}'.format(faceRecord['Face']['BoundingBox']))

    print('Faces not indexed:')
    for unindexedFace in response['UnindexedFaces']:
        print(' Location: {}'.format(unindexedFace['FaceDetail']
['BoundingBox']))
        print(' Reasons:')
        for reason in unindexedFace['Reasons']:
            print('   ' + reason)
    return len(response['FaceRecords'])

def main():
    bucket = 'amzn-s3-demo-bucket'
    collection_id = 'collection-id'
    photo = 'photo-name'

    indexed_faces_count = add_faces_to_collection(bucket, photo, collection_id)
    print("Faces indexed count: " + str(indexed_faces_count))

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

.NET

Dieses Beispiel zeigt die Identifikatoren für Gesichter an, die zur Sammlung hinzugefügt wurden.

Ändern Sie den Wert von `collectionId` in den Namen der Sammlung, die Sie einem Gesicht hinzufügen möchten. Ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, das Sie in Schritt 2 verwendet haben.

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

using System;
using System.Collections.Generic;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class AddFaces
```

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

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        Image image = new Image()
        {
            S3Object = new S3Object()
            {
                Bucket = bucket,
                Name = photo
            }
        };

        IndexFacesRequest indexFacesRequest = new IndexFacesRequest()
        {
            Image = image,
            CollectionId = collectionId,
            ExternalImageId = photo,
            DetectionAttributes = new List<String>(){ "ALL" }
        };

        IndexFacesResponse indexFacesResponse =
 rekognitionClient.IndexFaces(indexFacesRequest);

        Console.WriteLine(photo + " added");
        foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
            Console.WriteLine("Face detected: Faceid is " +
               faceRecord.Face.FaceId);
    }
}
```

IndexFaces Operationsanforderung

Bei der Eingabe in `IndexFaces` handelt es sich um das Image, das indiziert werden soll, und die Sammlung, der das Gesicht oder die Gesichter hinzugefügt werden sollen.

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "ExternalImageId": "input.jpg",
    "DetectionAttributes": [
        "DEFAULT"
    ],
    "MaxFaces": 1,
    "QualityFilter": "AUTO"
}
```

IndexFaces Antwort auf die Operation

`IndexFaces` gibt Informationen über die Gesichter zurück, die im Bild erkannt wurden. Beispiel: Die folgende JSON-Antwort enthält die Standard-Erkennungsattribute für Gesichter, die im eingegebenen Bild erkannt wurden. Das Beispiel zeigt auch nicht indizierte Gesichter, weil der Wert des `MaxFaces`-Eingabeparameters überschritten wurde das `Reasons`-Array enthält EXCEEDS_MAX_FACES. Wenn ein Gesicht aus Qualitätsgründen nicht indiziert ist, enthält `Reasons` Werte wie z. B. LOW_SHARPNESS oder LOW_BRIGHTNESS. Weitere Informationen finden Sie unter 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"
            ]
        }
    ]
}
```

Um alle Gesichtsinformationen zu erhalten, geben Sie 'ALL' als Anforderungsparameter `DetectionAttributes` an. Beachten Sie etwa in der folgenden Beispielantwort die zusätzlichen Informationen im Element `faceDetail`, die nicht auf dem Server erhalten bleiben:

- 25 wichtige Gesichtsmerkmale (verglichen mit nur 5 aus dem vorangegangenen Beispiel)
- Zehn Gesichtsmerkmale (Brille, Bart, Okklusion, Blickrichtung usw.)
- Emotionen (siehe Element `emotion`)

Das `face`-Element stellt Metadaten zur Verfügung, die auf dem Server beibehalten werden.

`FaceModelVersion` ist die Version des Gesichtsmodells, die der Sammlung zugeordnet ist. Weitere Informationen finden Sie unter Grundlegendes zur Modellversionierung.

`OrientationCorrection` ist die geschätzte Ausrichtung des Bilds. Es werden keine Informationen zur Korrektur der Ausrichtung zurückgegeben, wenn Sie eine Version des Gesichtserkennungsmodells nach Version 3 verwenden. Weitere Informationen finden Sie unter Erhalten der Bildausrichtung und der Koordinaten von Begrenzungsrahmen.

Die folgende Beispielantwort zeigt das zurückgegebene JSON, wenn [„ALL“] angegeben wurde:

```
{
    "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": []
}
```

Gesichter und zugehörige Benutzer in einer Sammlung auflisten

Sie können den ListFacesVorgang verwenden, um Gesichter und die ihnen zugehörigen Benutzer in einer Sammlung aufzulisten.

Weitere Informationen finden Sie unter Verwalten von Gesichtern in einer Sammlung.

Auflisten von Gesichtern in einer Sammlung (SDK)

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `ListFaces`-Operation.

   Java

   Dieses Beispiel zeigt eine Liste mit Gesichtern in einer Sammlung an.

   Ändern Sie den Wert von `collectionId` in die gewünschte Sammlung.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
// 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

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `list-faces` CLI-Vorgang an. Ersetzen Sie den Wert von `collection-id` durch den Namen der Sammlung, die Sie auflisten möchten. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
aws rekognition list-faces --collection-id "collection-id"  --profile profile-
name
```

Python

Dieses Beispiel zeigt eine Liste mit Gesichtern in einer Sammlung an.

Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
# Copyright 2018 Amazon.com, Inc. or its 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

Dieses Beispiel zeigt eine Liste mit Gesichtern in einer Sammlung an.

Ändern Sie den Wert von `collectionId` in die gewünschte Sammlung.

```
//Copyright 2018 Amazon.com, Inc. or its 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 Operationsanforderung

Die Eingabe für `ListFaces` ist die ID der Sammlung, für die Sie Gesichter auflisten möchten. `MaxResultsist` die maximale Anzahl von Gesichtern, die zurückgegeben werden sollen. ListFaces nimmt auch eine Liste von Gesichtern auf, IDs nach denen die Ergebnisse gefiltert werden sollen, und eine angegebene Benutzer-ID, um nur Gesichter aufzulisten, die dem angegebenen Benutzer zugeordnet sind.

```
{
    "CollectionId": "MyCollection",
    "MaxResults": 1
}
```

Wenn die Antwort mehr Gesichter als von `MaxResults` angefordert aufweist, wird ein Token zurückgegeben, das Sie verwenden können, um den nächsten Ergebnissatz in einem nachfolgenden Aufruf an `ListFaces` abzurufen. Zum Beispiel:

```
{
    "CollectionId": "MyCollection",
    "NextToken": "sm+5ythT3aeEVIR4WA....",
    "MaxResults": 1
}
```

ListFaces Reaktion auf den Vorgang

Bei der Antwort von `ListFaces` handelt es sich um Informationen über die Gesichtsmetadaten, die in der angegebenen Sammlung gespeichert sind.

- FaceModelVersion— Die Version des Gesichtsmodells, das der Kollektion zugeordnet ist. Weitere Informationen finden Sie unter Grundlegendes zur Modellversionierung.
- Gesichter – Informationen über die Gesichter in der Sammlung. Dies umfasst Informationen über BoundingBox, Vertrauen, Bild-IDs und die Gesichts-ID. Weitere Informationen finden Sie unter Gesicht.

- NextToken— Das Token, das verwendet wird, um die nächsten Ergebnisse zu erhalten.


```
{
    "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"
        }
    ]
```

```
}
```

Löschen von Gesichtern aus einer Sammlung

Sie können die Operation DeleteFaces zum Löschen von Gesichtern aus einer Sammlung verwenden. Weitere Informationen finden Sie unter Verwalten von Gesichtern in einer Sammlung.

Löschen von Gesichtern aus einer Sammlung

1. Wenn Sie dies noch nicht getan haben:
   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.
   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.
2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `DeleteFaces`-Operation.

   Java

   Dieses Beispiel löscht ein einzelnes Gesicht aus einer Sammlung.

   Ändern Sie den Wert von `collectionId` in die Sammlung, die das Gesicht enthält, das Sie löschen möchten. Ändern Sie den Wert von `faces` in die ID des Gesichts, das Sie löschen möchten. Um mehrere Gesichter zu löschen, fügen Sie die Fläche der `faces` Anordnung IDs hinzu.

```
//Copyright 2018 Amazon.com, Inc. or its 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-
xxxxxxxxxxx"};

   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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
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

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `delete-faces` CLI-Vorgang an. Ersetzen Sie den Wert von `collection-id` durch den Namen der Sammlung, die das Gesicht enthält, das gelöscht werden soll. Ersetzen Sie den Wert von `face-ids` durch ein Array von IDs Gesichtern, die Sie löschen möchten. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
aws rekognition delete-faces --collection-id "collection-id" --face-ids "faceid"
 --profile profile-name
```

Python

Dieses Beispiel löscht ein einzelnes Gesicht aus einer Sammlung.

Ändern Sie den Wert von `collectionId` in die Sammlung, die das Gesicht enthält, das Sie löschen möchten. Ändern Sie den Wert von `faces` in die ID des Gesichts, das Sie löschen möchten. Um mehrere Flächen zu löschen, fügen Sie die Fläche der `faces` Reihe IDs hinzu. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
# Copyright 2018 Amazon.com, Inc. or its 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

Dieses Beispiel löscht ein einzelnes Gesicht aus einer Sammlung.

Ändern Sie den Wert von `collectionId` in die Sammlung, die das Gesicht enthält, das Sie löschen möchten. Ändern Sie den Wert von `faces` in die ID des Gesichts, das Sie löschen möchten. Um mehrere Gesichter zu löschen, fügen Sie das Gesicht der `faces` Liste IDs hinzu.

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

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

using System;
using System.Collections.Generic;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class DeleteFaces
{
    public static void Example()
    {
        String collectionId = "MyCollection";
        List<String> faces = new List<String>() { "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx" };

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        DeleteFacesRequest deleteFacesRequest = new DeleteFacesRequest()
        {
            CollectionId = collectionId,
            FaceIds = faces
        };

        DeleteFacesResponse deleteFacesResponse =
 rekognitionClient.DeleteFaces(deleteFacesRequest);
        foreach (String face in deleteFacesResponse.DeletedFaces)
            Console.WriteLine("FaceID: " + face);
    }
}
```

DeleteFaces Betriebsanfrage

Die Eingabe für `DeleteFaces` ist die ID der Sammlung, die die Gesichter enthält, und eine Reihe von Gesichtern IDs für die zu löschenden Gesichter.

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

```
}
```

DeleteFaces Reaktion auf den Vorgang

Die `DeleteFaces` Antwort enthält eine Reihe von Gesichtern IDs für die Gesichter, die gelöscht wurden.

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

Wenn die in der Eingabe IDs angegebenen Gesichter derzeit einem Benutzer zugeordnet sind, werden sie unter Angabe UnsuccessfulFaceDeletions triftiger Gründe zurückgegeben.

```
{
   "DeletedFaces": [
        "daf29cac-f910-41e9-851f-6eeb0e08f973"
    ],
     "UnsuccessfulFaceDeletions" : [
        {
           "FaceId" : "0b683aed-a0f1-48b2-9b5e-139e9cc2a757",
           "UserId" : "demoUser1",
           "Reason" : ["ASSOCIATED_TO_AN_EXISTING_USER"]
        }
     ]
}
```

Erstellen eines Benutzers

Sie können den CreateUserVorgang verwenden, um mithilfe einer von Ihnen angegebenen eindeutigen Benutzer-ID einen neuen Benutzer in einer Sammlung zu erstellen. Anschließend können Sie dem neu erstellten Benutzer mehrere Gesichter zuordnen.

So erstellen Sie einen Benutzer (SDK)

1. Wenn Sie dies noch nicht getan haben:

a. Erstellen oder aktualisieren Sie ein IAM-Benutzerkonto mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `CreateUser`-Operation.

Java

In diesem Java-Codebeispiel wird ein Benutzer erstellt.

```
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

Dieser AWS CLI Befehl erstellt mithilfe der `create-user` CLI-Operation einen Benutzer.

```
aws rekognition create-user --user-id user-id --collection-id collection-name --
region region-name
--client-request-token request-token
```

Python

In diesem Python-Codebeispiel wird ein Benutzer erstellt.

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

import boto3
from botocore.exceptions import ClientError
import logging

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def create_user(collection_id, user_id):
    """
    Creates a new User within a collection specified by CollectionId.
    Takes UserId as a parameter, which is a user provided ID which
    should be unique within the collection.

    :param collection_id: The ID of the collection where the indexed faces will
 be stored at.
    :param user_id: ID for the UserID to be created. This ID needs to be unique
 within the collection.

    :return: The indexFaces response
    """
    try:
        logger.info(f'Creating user: {collection_id}, {user_id}')
        client.create_user(
            CollectionId=collection_id,
            UserId=user_id
```

```
        )
    except ClientError:
        logger.exception(f'Failed to create user with given user id:
 {user_id}')
        raise

def main():
    collection_id = "collection-id"
    user_id = "user-id"
    create_user(collection_id, user_id)

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

Go

Dieses Go-Codebeispiel verwendet das AWS Go SDK V2 und erstellt einen Benutzer.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rekognition"
)

func main() {
 // Load the AWS SDK configuration
 cfg, err := config.LoadDefaultConfig(context.TODO(), config.WithRegion("us-
west-2"))
 if err != nil {
  log.Fatalf("Failed to load configuration: %v", err)
 }

 // Create a Rekognition client
 client := rekognition.NewFromConfig(cfg)

 // Set up the input parameters
```

```
 input := &rekognition.CreateUserInput{
  CollectionId: aws.String("my-new-collection"), // Replace with your collection
 ID
  UserId:       aws.String("user12345678910"),   // Replace with desired user ID
 }

 // Call the CreateUser operation
 result, err := client.CreateUser(context.TODO(), input)
 if err != nil {
  log.Fatalf("Failed to create user: %v", err)
 }

 // Print out the results
 fmt.Printf("User created successfully:\n")

}
```

Löschen eines Benutzers

Sie können den DeleteUserVorgang verwenden, um einen Benutzer basierend auf der angegebenen UserID aus einer Sammlung zu löschen. Beachten Sie, dass alle Gesichter, die mit der UserID verknüpft sind, von der UserID getrennt werden, bevor die angegebene UserID gelöscht wird.

So löschen Sie einen Benutzer (SDK)

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie ein IAM-Benutzerkonto mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie die AWS CLI und die. AWS SDKs Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `DeleteUser`-Operation.

   Java

   In diesem Java-Codebeispiel wird ein Benutzer gelöscht.

```
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

Dieser AWS CLI Befehl löscht einen Benutzer mithilfe der `create-user` CLI-Operation.

```
aws rekognition delete-user --collection-id MyCollection
--user-id user-id --collection-id collection-name --region region-name
```

Python

In diesem Python-Codebeispiel wird ein Benutzer gelöscht.

```
# Copyright 2018 Amazon.com, Inc. or its 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()
```

Gesichter einem Benutzer zuordnen

Sie können den AssociateFacesVorgang verwenden, um mehrere einzelne Gesichter einem einzelnen Benutzer zuzuordnen. Um ein Gesicht einem Benutzer zuzuordnen, müssen Sie zuerst eine Sammlung und einen Benutzer erstellen. Beachten Sie, dass sich die Gesichtsvektoren in derselben Sammlung befinden müssen, in der sich der Benutzervektor befindet.

Um Gesichter zuzuordnen (SDK)

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `AssociateFaces`-Operation.

   Java

   In diesem Java-Codebeispiel wird ein Gesicht einem Benutzer zugeordnet.

```
import java.util.Arrays;
import java.util.List;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AssociateFacesRequest;
import com.amazonaws.services.rekognition.model.AssociateFacesResult;


public class AssociateFaces {

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


        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

        /* Replace the below configurations to allow you successfully run the
 example
```

```
            @collectionId: The collection where user and faces are stored
            @userId: The user which faces will get associated to
            @faceIds: The list of face IDs that will get associated to the given
 user
            @userMatchThreshold: Minimum User match confidence required for the
 face to
                                 be associated with a User that has at least one
 faceID already associated
            */

           String collectionId = "MyCollection";
           String userId = "demoUser";
           String faceId1 = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";
           String faceId2 = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";
           List<String> faceIds = Arrays.asList(faceid1,faceid2);

           float userMatchThreshold = 0f;
           System.out.println("Associating faces to the existing user: " +
                   userId);

           AssociateFacesRequest request = new AssociateFacesRequest()
                   .withCollectionId(collectionId)
                   .withUserId(userId)
                   .withFaceIds(faceIds)
                   .withUserMatchThreshold(userMatchThreshold);

           AssociateFacesResult result = rekognitionClient.associateFaces(request);

           System.out.println("Successful face associations: " +
 result.getAssociatedFaces().size());
           System.out.println("Unsuccessful face associations: " +
 result.getUnsuccessfulFaceAssociations().size());
       }

}
```

AWS CLI

Dieser AWS CLI Befehl ordnet einem Benutzer mithilfe der `associate-faces` CLI-Operation ein Gesicht zu.

```
aws rekognition associate-faces --user-id user-id --face-ids face-id-1 face-id-2
```

```
--collection-id collection-name
--region region-name
```

Python

In diesem Python-Codebeispiel wird ein Gesicht einem Benutzer zugeordnet.

```
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 Antwort auf den Vorgang

Die Antwort für `AssociateFaces` enthält den `UserStatus`, d. h. den Status der Anforderung zur Trennung der Verbindung sowie eine Liste der zu verknüpfenden `FaceIds`. Eine Liste von `UnsuccessfulFaceAssociations` wird ebenfalls zurückgegeben. Nach dem Absenden einer Anforderung an `AssociateFaces` kann es etwa eine Minute dauern, bis die Operation abgeschlossen ist.

Aus diesem Grund UserStatus wird der zurückgegeben, der die folgenden Werte haben kann:

- ERSTELLT: Gibt an, dass der „Benutzer“ erfolgreich erstellt wurde und ihm derzeit keine Gesichter zugeordnet sind. 'Benutzer' befindet sich in diesem Status, bevor ein erfolgreicher 'AssociateFaces' -Aufruf getätigt wird.
- AKTUALISIERUNG: Bedeutet, dass der „Benutzer“ aktualisiert wird, um die neu zugeordneten/getrennten Gesichter widerzuspiegeln, und dass er in wenigen Sekunden AKTIV wird. In diesem Status können Suchergebnisse den Begriff „Benutzer“ enthalten, und Kunden können sich dafür entscheiden, diesen Begriff in den zurückgegebenen Ergebnissen zu ignorieren.
- AKTIV: Bedeutet, dass der „Benutzer“ aktualisiert wurde, sodass er alle zugeordneten/getrennten Gesichter wiedergibt, und dass er sich in einem durchsuchbaren Status befindet.

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

Trennen der Zuordnung von Gesichtern zu einem Benutzer

Sie können den DisassociateFacesVorgang verwenden, um die Zuordnung zwischen einer Benutzer-ID und einer Gesichts-ID aufzuheben.

Um Gesichter zu trennen (SDK)

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `DisassociateFaces`-Operation.

   Java

   Dieses Java-Beispiel entfernt die Zuordnung zwischen einer Gesichts-ID und einer UserID mit der `DisassociateFaces` Operation.

```
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

Dieser AWS CLI Befehl entfernt die Zuordnung zwischen einer FaceID und einer UserID bei der Operation. `DisassociateFaces`

```
aws rekognition disassociate-faces --face-ids list-of-face-ids
--user-id user-id --collection-id collection-name --region region-name
```

Python

Das folgende Beispiel entfernt die Zuordnung zwischen einer Gesichts-ID und einer UserID mit der `DisassociateFaces`-Operation.

```
from botocore.exceptions import ClientError
import boto3
import logging

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def disassociate_faces(collection_id, user_id, face_ids):
    """
    Disassociate stored faces within collection to the given user

    :param collection_id: The ID of the collection where user and faces are
 stored.
    :param user_id: The ID of the user that we want to disassociate faces from
    :param face_ids: The list of face IDs to be disassociated from the given
 user

    :return: response of AssociateFaces API
    """
    logger.info(f'Disssociating faces from user: {user_id}, {face_ids}')
    try:
        response = client.disassociate_faces(
            CollectionId=collection_id,
            UserId=user_id,
            FaceIds=face_ids
        )
```

```
        print(f'- disassociated {len(response["DisassociatedFaces"])} faces')
    except ClientError:
        logger.exception("Failed to disassociate faces from the given user")
        raise
    else:
        print(response)
        return response

def main():
    face_ids = ["faceId1", "faceId2"]
    collection_id = "collection-id"
    user_id = "user-id"
    disassociate_faces(collection_id, user_id, face_ids)

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

DisassociateFaces Antwort auf den Vorgang

Die Antwort für `DisassociateFaces` enthält den `UserStatus`, d. h. den Status der Anforderung zur Trennung der Verbindung sowie eine Liste der zu trennenden `FaceIds`. Eine Liste von `UnsuccessfulFaceDisassociations` wird ebenfalls zurückgegeben. Nach dem Absenden einer Anfrage an kann es etwa eine Minute dauern DisassociateFaces, bis der Vorgang abgeschlossen ist. Aus diesem Grund UserStatus wird der zurückgegeben, der die folgenden Werte haben kann:

- ERSTELLT: Gibt an, dass der „Benutzer“ erfolgreich erstellt wurde und ihm derzeit keine Gesichter zugeordnet sind. 'Benutzer' befindet sich in diesem Status, bevor ein erfolgreicher 'AssociateFaces' -Aufruf getätigt wird.
- AKTUALISIERUNG: Bedeutet, dass der „Benutzer“ aktualisiert wird, um die neu zugeordneten/getrennten Gesichter widerzuspiegeln, und dass er in wenigen Sekunden AKTIV wird. In diesem Status können Suchergebnisse den Begriff „Benutzer“ enthalten, und Kunden können sich dafür entscheiden, diesen Begriff in den zurückgegebenen Ergebnissen zu ignorieren.
- AKTIV: Bedeutet, dass der „Benutzer“ aktualisiert wurde, sodass er alle zugeordneten/getrennten Gesichter wiedergibt, und dass er sich in einem durchsuchbaren Status befindet.

```
{
    "UserStatus": "UPDATING",
    "DisassociatedFaces": [
```

```
        {
            "FaceId": "c92265d4-5f9c-43af-a58e-12be0ce02bc3"
        }
    ],
    "UnsuccessfulFaceDisassociations": [
        {
            "Reasons": [
                "ASSOCIATED_TO_A_DIFFERENT_IDENTITY"
            ],
            "FaceId": "f5817d37-94f6-4335-bfee-6cf79a3d806e",
            "UserId": "demoUser1"
        }
    ]
}
```

Auflisten von Benutzern in einer Sammlung

Sie können die ListUsersOperation zum Auflisten UserIds und zum Auflisten verwenden. UserStatus Verwenden Sie die Operation, um IDs das Gesicht zu sehen, das mit einer UserID verknüpft ist ListFaces.

Um Benutzer aufzulisten (SDK)

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das. AWS SDKs Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `ListUsers`-Operation.

   Java

   Dieses Java-Beispiel listet die Benutzer in einer Sammlung auf, die die `ListUsers`-Operation verwenden.

```
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

Dieser AWS CLI Befehl listet die Benutzer in einer Sammlung mit dem `ListUsers` Vorgang auf.

```
aws rekognition list-users --collection-id collection-id --max-results number-
of-max-results
```

Python

Das folgende Beispiel listet die Benutzer in einer Sammlung mit der `ListUsers`-Operation auf.

```
# Copyright 2018 Amazon.com, Inc. or its 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 Antwort auf den Vorgang

Die Antwort auf eine Anfrage an ListUsers enthält eine Liste der `Users` in der Sammlung enthaltenen Personen `UsedId` sowie die `UserStatus` Daten des Benutzers.


```
{
    "NextToken": "B1asJT3bAb/ttuGgPFV8BZoBZyGQzlUHXbuTNLh48a6enU7kXKw43hpOwizW7LOk/
Gk7Em09lznoq6+FcDCcSq2olrn7A98BLkt5keu+ZRVRUTyrXtT6J7Hmp
+ieQ2an6Zu0qzPfcdPeaJ9eAxG2d0WNrzJgi5hvmjoiSTTfKX3MQz1sduWQkvAAs4hZfhZoKFahFlqWofshCXa/
FHAAY3PL1PjxXbkNeSSMq8V7i1MlKCdrPVykCv9MokpPt7jtNvKPEZGUhxgBTFMxNWLEcFnzAiCWDg91dFy/
LalshPjXA9UVc5Gx9vIJNQ/
eO3cQRghAkCT3FOAiXsLAnA015ODTomZpWWVpqB21wKpI3LYmfAVFrDPGzpbTVlRmLsJm41bkmnBBBw9+DHz1Jn7zW
+qc5Fs3yaHu0f51Xg==",
    "Users": [
        {
            "UserId": "demoUser4",
            "UserStatus": "CREATED"
        },
        {
            "UserId": "demoUser2",
            "UserStatus": "CREATED"
        }
    ]
}
```


Suche nach einem Gesicht mit einer Gesichts-ID

Sie können diesen SearchFacesVorgang verwenden, um in einer Sammlung nach Benutzern zu suchen, die dem größten Gesicht in einem bereitgestellten Bild entsprechen.

Die Gesichts-ID wird in der Antwort der Operation IndexFaces zurückgegeben, wenn ein Gesicht erkannt wird, und der Sammlung hinzugefügt. Weitere Informationen finden Sie unter Verwalten von Gesichtern in einer Sammlung.

Suchen nach einem Gesicht in einer Sammlung mithilfe der Gesichts-ID (SDK)

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `SearchFaces`-Operation.

   Java

   Dieses Beispiel zeigt Informationen über Gesichter an, die mit einem Gesicht übereinstimmen, das durch seine ID gekennzeichnet ist.

   Ändern Sie den Wert von `collectionID` in die Sammlung, die das erforderliche Gesicht enthält. Ändern Sie den Wert von `faceId` in die Kennung des Gesichts, das Sie finden möchten.

```
//Copyright 2018 Amazon.com, Inc. or its 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();
      }
    }

}
```

Führen Sie den Beispielcode aus. Informationen zu übereinstimmenden Gesichtern werden angezeigt.

Java V2

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
// 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

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `search-faces` CLI-Vorgang an. Ersetzen Sie den Wert von `face-id` mit der Gesichtskennung, nach der Sie suchen

möchten, und ersetzen Sie den Wert von `collection-id` mit der Sammlung, die Sie durchsuchen möchten. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
aws rekognition search-faces --face-id face-id --collection-id "collection-id"
 --profile profile-name
```

Python

Dieses Beispiel zeigt Informationen über Gesichter an, die mit einem Gesicht übereinstimmen, das durch seine ID gekennzeichnet ist.

Ändern Sie den Wert von `collectionID` in die Sammlung, die das erforderliche Gesicht enthält. Ändern Sie den Wert von `faceId` in die Kennung des Gesichts, das Sie finden möchten. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

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

import boto3

def search_face_in_collection(face_id, collection_id):
    threshold = 90
    max_faces = 2

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

    response = client.search_faces(CollectionId=collection_id,
                                   FaceId=face_id,
                                   FaceMatchThreshold=threshold,
                                   MaxFaces=max_faces)

    face_matches = response['FaceMatches']
    print('Matching faces')
    for match in face_matches:
        print('FaceId:' + match['Face']['FaceId'])
        print('Similarity: ' + "{:.2f}".format(match['Similarity']) + "%")

    return len(face_matches)
```

```
def main():
    face_id = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
    collection_id = 'collection-id'

    faces = []
    faces.append(face_id)

    faces_count = search_face_in_collection(face_id, collection_id)
    print("faces found: " + str(faces_count))

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

.NET

Dieses Beispiel zeigt Informationen über Gesichter an, die mit einem Gesicht übereinstimmen, das durch seine ID gekennzeichnet ist.

Ändern Sie den Wert von `collectionID` in die Sammlung, die das erforderliche Gesicht enthält. Ändern Sie den Wert von `faceId` in die Kennung des Gesichts, das Sie finden möchten.

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

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

public class SearchFacesMatchingId
{
    public static void Example()
    {
        String collectionId = "MyCollection";
        String faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        // Search collection for faces matching the face id.
```

```
        SearchFacesRequest searchFacesRequest = new SearchFacesRequest()
        {
            CollectionId = collectionId,
            FaceId = faceId,
            FaceMatchThreshold = 70F,
            MaxFaces = 2
        };

        SearchFacesResponse searchFacesResponse =
 rekognitionClient.SearchFaces(searchFacesRequest);

        Console.WriteLine("Face matching faceId " + faceId);

        Console.WriteLine("Matche(s): ");
        foreach (FaceMatch face in searchFacesResponse.FaceMatches)
            Console.WriteLine("FaceId: " + face.Face.FaceId + ", Similarity: " +
 face.Similarity);
    }
}
```

Führen Sie den Beispielcode aus. Informationen zu übereinstimmenden Gesichtern werden angezeigt.

SearchFaces Operationsanforderung

Nach der Bereitstellung einer Gesichts-ID (jedem Gesicht in der Sammlung ist eine Gesichts-ID zugeordnet) durchsucht `SearchFaces` die festgelegte Gesichtersammlung nach ähnlichen Gesichtern. Die Antwort beinhaltet nicht das Gesicht, nach dem Sie gesucht haben. Sie enthält nur ähnliche Gesichter. Standardmäßig gibt `SearchFaces` Gesichter zurück, bei denen der Algorithmus mindestens 80% Ähnlichkeit erkennt. Die Ähnlichkeit gibt an, inwieweit das Gesicht mit dem Bild auf dem Eingabebild übereinstimmt. Optional können Sie die Funktion `FaceMatchThreshold` nutzen, um einen anderen Wert zu bestimmen.

```
{
    "CollectionId": "MyCollection",
    "FaceId": "0b683aed-a0f1-48b2-9b5e-139e9cc2a757",
    "MaxFaces": 2,
    "FaceMatchThreshold": 99
}
```

SearchFaces Antwort auf die Operation

Die Operation gibt eine Reihe von gefundenen übereinstimmenden Gesichtern sowie die von Ihnen eingegebene Gesichts-ID zurück.

```
{
    "SearchedFaceId": "7ecf8c19-5274-5917-9c91-1db9ae0449e2",
    "FaceMatches": [ list of face matches found ]
}
```

Für jede gefundene Gesichtsübereinstimmung enthält die Antwort Metadaten zur Ähnlichkeit und zum Gesicht wie in der folgenden Beispielantwort angezeigt:

```
{
   ...
    "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"
```

```
            }
        }
    ]
}
```

Suche nach einem Gesicht mit einem Bild

Sie können die Operation SearchFacesByImage für die Suche nach Gesichtern in einer Sammlung verwenden, die dem größten Gesicht in einem bereitgestellten Bild entspricht.

Weitere Informationen finden Sie unter Suche nach Gesichtern und Benutzern in einer Sammlung.

Suchen nach einem Gesicht in einer Sammlung mithilfe eines Bildes (SDK)

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`- und `AmazonS3ReadOnlyAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Laden Sie ein Bild in Ihren S3-Bucket hoch, auf dem ein oder mehrere Gesichter abgebildet sind.

   Weitere Anleitungen finden Sie unter Upload eines Objekts in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service.

3. Verwenden Sie die folgenden Beispiele zum Aufrufen der `SearchFacesByImage`-Operation.

   Java

   Dieses Beispiel zeigt Informationen über Gesichter an, die mit dem größten Gesicht in einem Bild übereinstimmen. Der Beispielcode legt die beiden Parameter `FaceMatchThreshold` und `MaxFaces` fest, um die in der Antwort zurückgegebenen Ergebnisse einzugrenzen.

   Im folgenden Beispiel ändern Sie Folgendes: Ändern Sie den Wert von `collectionId` in die Sammlung, die Sie durchsuchen möchten, ändern Sie den Wert von `bucket` in den Bucket mit dem Eingabebild und ändern Sie den Wert von `photo` in das Eingabebild.

   ```
   //Copyright 2018 Amazon.com, Inc. or its 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
// 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

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `search-faces-by-image` CLI-Vorgang an. Ersetzen Sie den Wert von `Bucket` durch den S3-Bucket, den Sie in Schritt 2 verwendet haben. Ersetzen Sie den Wert von `Name` durch den Namen der Bilddatei, die Sie in Schritt 2 verwendet haben. Ersetzen Sie den Wert von `collection-id` durch den Namen der Sammlung, in der Sie suchen möchten. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

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

Wenn Sie auf einem Windows-Gerät auf die CLI zugreifen, verwenden Sie doppelte Anführungszeichen anstelle von einfachen Anführungszeichen und maskieren Sie die inneren doppelten Anführungszeichen durch einen Backslash (d. h. \), um eventuell auftretende Parserfehler zu beheben. Sehen Sie sich zum Beispiel Folgendes an:

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

Python

Dieses Beispiel zeigt Informationen über Gesichter an, die mit dem größten Gesicht in einem Bild übereinstimmen. Der Beispielcode legt die beiden Parameter `FaceMatchThreshold` und `MaxFaces` fest, um die in der Antwort zurückgegebenen Ergebnisse einzugrenzen.

Nehmen Sie im folgenden Beispiel diese Änderungen vor: Ändern Sie den Wert von `collectionId` in die Sammlung, die Sie durchsuchen möchten, und ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, die Sie in Schritt 2 verwendet haben. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

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

import boto3

if __name__ == "__main__":

    bucket='amzn-s3-demo-bucket'
    collectionId='MyCollection'
    fileName='input.jpg'
    threshold = 70
    maxFaces=2

    client=boto3.client('rekognition')
```

```
    response=client.search_faces_by_image(CollectionId=collectionId,
                                Image={'S3Object':
{'Bucket':bucket,'Name':fileName}},
                                FaceMatchThreshold=threshold,
                                MaxFaces=maxFaces)


    faceMatches=response['FaceMatches']
    print ('Matching faces')
    for match in faceMatches:
            print ('FaceId:' + match['Face']['FaceId'])
            print ('Similarity: ' + "{:.2f}".format(match['Similarity']) + "%")
            print
```

.NET

Dieses Beispiel zeigt Informationen über Gesichter an, die mit dem größten Gesicht in einem Bild übereinstimmen. Der Beispielcode legt die beiden Parameter `FaceMatchThreshold` und `MaxFaces` fest, um die in der Antwort zurückgegebenen Ergebnisse einzugrenzen.

Nehmen Sie im folgenden Beispiel diese Änderungen vor: Ändern Sie den Wert von `collectionId` in die Sammlung, die Sie durchsuchen möchten, und ersetzen Sie die Werte von `bucket` und `photo` durch die Namen des Amazon-S3-Buckets und des Bilds, die Sie in Schritt 2 verwendet haben.

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

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

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

```
        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        // Get an image object from S3 bucket.
        Image image = new Image()
        {
            S3Object = new S3Object()
            {
                Bucket = bucket,
                Name = photo
            }
        };

        SearchFacesByImageRequest searchFacesByImageRequest = new
 SearchFacesByImageRequest()
        {
            CollectionId = collectionId,
            Image = image,
            FaceMatchThreshold = 70F,
            MaxFaces = 2
        };

        SearchFacesByImageResponse searchFacesByImageResponse =
 rekognitionClient.SearchFacesByImage(searchFacesByImageRequest);

        Console.WriteLine("Faces matching largest face in image from " + photo);
        foreach (FaceMatch face in searchFacesByImageResponse.FaceMatches)
            Console.WriteLine("FaceId: " + face.Face.FaceId + ", Similarity: " +
 face.Similarity);
    }
}
```

SearchFacesByImage Operationsanforderung

Die in `SearchFacesImageByImage` eingegebenen Parameter entsprechen der Sammlung, in der gesucht werden soll, und dem Speicherort des Quellbilds. In diesem Beispiel wird das in einem Amazon-S3-Bucket gespeicherte Quellenbild (`S3Object`) geladen. Außerdem sind die maximale Anzahl von Gesichtern (`Maxfaces`), die zurückgegeben wird, und die minimale Übereinstimmung angegeben, die eingehalten werden muss, damit ein Gesicht (`FaceMatchThreshold`) zurückgegeben wird.

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "MaxFaces": 2,
    "FaceMatchThreshold": 99
}
```

SearchFacesByImage Antwort auf die Operation

Für ein bereitgestelltes Eingabebild (im JPEG- oder PNG-Format) erkennt die Operation zuerst das Gesicht auf dem Eingabebild. Anschließend durchsucht sie die festgelegte Gesichtersammlung nach ähnlichen Gesichtern.

> **Note**
>
> Wenn der Dienst mehrere Gesichter auf dem Eingabebild erkennt, wird nach dem größten erkannten Gesicht gesucht.

Die Operation gibt eine Reihe von gefundenen übereinstimmenden Gesichtern sowie Informationen zu den eingegebenen Gesichtern zurück. Dies umfasst Informationen wie den Begrenzungsrahmen sowie den Wahrscheinlichkeitswert, der die Wahrscheinlichkeit angibt, dass der Begrenzungsrahmen ein Gesicht enthält.

Standardmäßig gibt `SearchFacesByImage` Gesichter zurück, bei denen der Algorithmus mindestens 80% Ähnlichkeit erkennt. Die Ähnlichkeit gibt an, inwieweit das Gesicht mit dem Bild auf dem Eingabebild übereinstimmt. Optional können Sie die Funktion `FaceMatchThreshold` nutzen, um einen anderen Wert zu bestimmen. Für jede gefundene Gesichtsübereinstimmung enthält die Antwort Metadaten zur Ähnlichkeit und zum Gesicht wie in der folgenden Beispielantwort angezeigt:

```
{
    "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
}
```


Suche nach Benutzern (Gesichts-ID/Benutzer-ID)

Sie können den SearchUsersVorgang verwenden, um in einer bestimmten Sammlung nach Benutzern zu suchen, die mit einer angegebenen Gesichts-ID oder Benutzer-ID übereinstimmen. Bei der Operation werden die zurückgegebenen `UserIds` Objekte nach dem höchsten Ähnlichkeitswert über dem angeforderten Wert sortiert UserMatchThreshold. Die Benutzer-ID wird während des CreateUsers Vorgangs erstellt. Weitere Informationen finden Sie unter Benutzer in einer Sammlung verwalten.

Um Benutzer zu suchen (SDK)

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `SearchUsers`-Operation.

   Java

   In diesem Java-Beispiel werden die Benutzer in einer Sammlung mithilfe der `SearchUsers`-Operation durchsucht.

```
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

Dieser AWS CLI Befehl durchsucht die Benutzer in einer Sammlung mit der `SearchUsers` Operation.

```
aws rekognition search-users --face-id face-id --collection-id collection-id --
region region-name
```

Python

Das folgende Beispiel sucht die Benutzer in einer Sammlung mit der `SearchUsers`-Operation.

```
# Copyright 2018 Amazon.com, Inc. or its 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 Operationsanforderung

Bei Angabe einer FaceID oder UserID wird die angegebene CollectionID nach Benutzerübereinstimmungen SearchUsers durchsucht. Gibt standardmäßig Benutzer SearchUsers zurück, IDs für die der Ähnlichkeitswert größer als 80% ist. Die Ähnlichkeit gibt an, wie genau die UserID mit der angegebenen Gesichts-ID oder UserID übereinstimmt. Wenn mehrere Benutzer zurückgegeben IDs werden, werden sie in der Reihenfolge vom höchsten bis zum niedrigsten Ähnlichkeitswert aufgeführt. Optional können Sie den verwenden UserMatchThreshold , um einen anderen Wert anzugeben. Weitere Informationen finden Sie unter Benutzer in einer Sammlung verwalten.

Im Folgenden finden Sie ein Beispiel für eine SearchUsers Anfrage mit`UserId`:

```
{
   "CollectionId": "MyCollection",
   "UserId": "demoUser1",
   "MaxUsers": 2,
   "UserMatchThreshold": 99
}
```

Im Folgenden finden Sie ein Beispiel für eine SearchUsers Anfrage mit`FaceId`:

```
{
    "CollectionId": "MyCollection",
    "FaceId": "bff43c40-cfa7-4b94-bed8-8a08b2205107",
    "MaxUsers": 2,
    "UserMatchThreshold": 99
}
```

SearchUsers Antwort auf den Vorgang

Bei der Suche mit einem `FaceId` SearchUsers beinhaltet die Antwort `FaceId` für das für den `SearchedFace` sowie eine Liste von `UserMatches` und das `UserId` und `UserStatus` für jeden Benutzer.

```
{
    "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"
}
```

Bei der Suche mit a `UserId` SearchUsers beinhaltet die Antwort für zusätzlich zu den `SearchedUser` anderen Antwortelementen auch das `UserId` für.

```
{
    "SearchedUser": {
        "UserId": "demoUser1"
    },
    "UserMatches": [
        {
            "User": {
                "UserId": "demoUser2",
                "UserStatus": "ACTIVE"
            },
            "Similarity": 99.97946166992188
        }
    ],
    "FaceModelVersion": "6"
}
```

Suchen nach Benutzern (Bild)

`SearchUsersByImage` durchsucht die angegebene CollectionID nach Benutzern in einer Sammlung, die dem größten in einem bereitgestellten Bild erkannten Gesicht entsprechen. Standardmäßig wird ein Benutzer SearchUsersByImage zurückgegeben, IDs für den der Ähnlichkeitswert größer als 80% ist. Die Ähnlichkeit gibt an, wie genau die UserID mit dem größten Gesicht übereinstimmt, das im bereitgestellten Bild erkannt wurde. Wenn mehrere Benutzer zurückgegeben IDs werden, werden sie in der Reihenfolge vom höchsten bis zum niedrigsten Ähnlichkeitswert aufgeführt. Optional können Sie den verwenden UserMatchThreshold , um einen anderen Wert anzugeben. Weitere Informationen finden Sie unter the section called “Benutzer in einer Sammlung verwalten”.

Um Benutzer nach Bildern zu suchen (SDK)

1. Wenn Sie dies noch nicht getan haben:

    a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

b. Installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `SearchUsersByImage`-Operation.

Java

In diesem Java-Beispiel werden die Benutzer in einer Sammlung basierend auf einem Eingabebild mithilfe der `SearchUsersByImage`-Operation durchsucht.

```
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 Mit diesem Befehl werden die Benutzer in einer Sammlung anhand eines Eingabebilds mit der `SearchUsersByImage` Operation durchsucht.

```
aws rekognition search-users-by-image --image '{"S3Object":
{"Bucket":"s3BucketName","Name":"file-name"}}' --collection-id MyCollectionId --
region region-name
```

Python

Das folgende Beispiel sucht die Benutzer in einer Sammlung basierend auf einem Eingabebild mit der `SearchUsersByImage`-Operation.

```
# Copyright 2018 Amazon.com, Inc. or its 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 Operationsanforderung

Die Anforderung für `SearchUsersByImage` enthält die Sammlung, in der gesucht werden soll, und den Speicherort des Quellbildes. In diesem Beispiel wird das in einem Amazon-S3-Bucket gespeicherte Quellenbild (`S3Object`) geladen. Außerdem sind die maximale Anzahl von Benutzern (`MaxUsers`), die zurückgegeben wird, und die minimale Übereinstimmung angegeben, die eingehalten werden muss, damit ein Benutzer (`UserMatchThreshold`) zurückgegeben wird.

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "MaxUsers": 2,
    "UserMatchThreshold": 99
}
```

SearchUsersByImage Antwort auf die Operation

Die Antwort für `SearchUsersByImage` beinhaltet ein `FaceDetail` Objekt für die`SearchedFace`, sowie eine Liste von Objekten UserMatches mit dem `UserIdSimilarity`, und `UserStatus` für jedes Objekt. Wenn das Eingabebild mehr als ein Gesicht enthält, UnsearchedFaces wird auch eine Liste von Gesichtern zurückgegeben.

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

Suche nach Gesichtern in gespeicherten Videos

Sie können eine Sammlung nach Gesichtern durchsuchen, die auf Gesichter von Personen in einem gespeicherten Video oder einem Streaming-Video passen. Dieser Abschnitt behandelt die Suche nach Gesichtern in einem gespeicherten Video. Weitere Informationen zum Suchen von Gesichtern in einem Streaming-Video finden Sie unter Arbeiten mit Streaming-Videoereignissen.

Die Gesichter, nach denen Sie suchen, müssen zunächst in einer Sammlung mithilfe von IndexFaces indiziert werden. Weitere Informationen finden Sie unter Hinzufügen von Gesichtern zu einer Sammlung.

Die Amazon-Rekognition-Video-Gesichtssuche folgt demselben asynchronen Arbeitsablauf wie andere Amazon-Rekognition-Video-Operationen, die in einem Amazon-S3-Bucket gespeicherte Videos analysieren. Um mit der Suche nach Gesichtern in einem gespeicherten Video zu beginnen, rufen Sie an StartFaceSearchund geben Sie die ID der Sammlung an, die Sie durchsuchen möchten. Amazon Rekognition Video veröffentlicht den Abschlussstatus der Videoanalyse in einem Amazon-Simple-Notification-Service-Thema (Amazon SNS). Wenn die Videoanalyse erfolgreich ist, rufen Sie GetFaceSearch auf, um die Suchergebnisse abzurufen. Weitere Informationen zum Starten der Videoanalyse und zum Abrufen der Ergebnisse finden Sie unter Amazon-Rekognition-Video-Operationen aufrufen.

Die folgende Prozedur zeigt, wie eine Sammlung nach Gesichtern durchsucht wird, die den Gesichtern von in einem Video erkannten Personen entsprechen. Die Prozedur zeigt auch, wie Sie die Tracking-Daten für Personen erhalten, die im Video gefunden wurden. Das Verfahren erweitert den Code in Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK), der eine Amazon-Simple-Queue-Service-Warteschlange (Amazon SQS) verwendet, um den Abschlussstatus einer Videoanalyseanforderung zu erhalten.

So durchsuchen Sie ein Video nach übereinstimmenden Gesichtern (SDK)

1. Erstellen Sie eine Sammlung.
2. Indizieren Sie ein Gesicht in der Sammlung.

3. Führen Sie Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) aus.
4. Fügen Sie den folgenden Code in der Klasse `VideoDetect` ein, die Sie in Schritt 3 erstellt haben.

   Java

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

      //Face collection search in video
 ======================================================================
      private static void StartFaceSearchCollection(String bucket, String
 video, String collection) throws Exception{

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

       StartFaceSearchRequest req = new StartFaceSearchRequest()
               .withCollectionId(collection)
               .withVideo(new Video()
                       .withS3Object(new S3Object()
                               .withBucket(bucket)
                               .withName(video)))
               .withNotificationChannel(channel);



       StartFaceSearchResult startPersonCollectionSearchResult =
 rek.startFaceSearch(req);
       startJobId=startPersonCollectionSearchResult.getJobId();

   }

   //Face collection search in video
 ======================================================================
   private static void GetFaceSearchCollectionResults() throws Exception{

      GetFaceSearchResult faceSearchResult=null;
      int maxResults=10;
      String paginationToken=null;
```

```
      do {

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


          faceSearchResult  = rek.getFaceSearch(
                  new GetFaceSearchRequest()
                  .withJobId(startJobId)
                  .withMaxResults(maxResults)
                  .withNextToken(paginationToken)
                  .withSortBy(FaceSearchSortBy.TIMESTAMP)
                  );


          VideoMetadata videoMetaData=faceSearchResult.getVideoMetadata();

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


          //Show search results
          List<PersonMatch> matches=
                  faceSearchResult.getPersons();

          for (PersonMatch match: matches) {
              long milliSeconds=match.getTimestamp();
              System.out.print("Timestamp: " + Long.toString(milliSeconds));
              System.out.println(" Person number: " +
match.getPerson().getIndex());
              List <FaceMatch> faceMatches = match.getFaceMatches();
              if (faceMatches != null) {
                  System.out.println("Matches in collection...");
                  for (FaceMatch faceMatch: faceMatches){
                      Face face=faceMatch.getFace();
                      System.out.println("Face Id: "+ face.getFaceId());
                      System.out.println("Similarity: " +
faceMatch.getSimilarity().toString());
```

```
                    System.out.println();
                }
            }
            System.out.println();
        }

        System.out.println();

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

  }
```

Ersetzen Sie in der Funktion `main` die folgenden Zeilen:

```
        StartLabelDetection(bucket, video);

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

mit:

```
        String collection="collection";
        StartFaceSearchCollection(bucket, video, collection);

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

Java V2

Dieser Code stammt aus dem GitHub Beispiel-Repository des AWS Documentation SDK. Das vollständige Beispiel finden Sie hier.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import java.util.List;

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

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetectFaces {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

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

```
        startFaceDetection(rekClient, channel, bucket, video);
        getFaceResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startFaceDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

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

            StartFaceDetectionRequest faceDetectionRequest =
StartFaceDetectionRequest.builder()
                    .jobTag("Faces")
                    .faceAttributes(FaceAttributes.ALL)
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartFaceDetectionResponse startLabelDetectionResult =
rekClient.startFaceDetection(faceDetectionRequest);
            startJobId = startLabelDetectionResult.jobId();

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

    public static void getFaceResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetFaceDetectionResponse faceDetectionResponse = null;
            boolean finished = false;
            String status;
```

```
            int yy = 0;

            do {
                if (faceDetectionResponse != null)
                    paginationToken = faceDetectionResponse.nextToken();

                GetFaceDetectionRequest recognitionRequest =
GetFaceDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {

                    faceDetectionResponse =
rekClient.getFaceDetection(recognitionRequest);
                    status = faceDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
faceDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                // Show face information.
                List<FaceDetection> faces = faceDetectionResponse.faces();
```

```
                for (FaceDetection face : faces) {
                    String age = face.face().ageRange().toString();
                    String smile = face.face().smile().toString();
                    System.out.println("The detected face is estimated to be"
                            + age + " years old.");
                    System.out.println("There is a smile : " + smile);
                }

            } while (faceDetectionResponse != null &&
 faceDetectionResponse.nextToken() != null);

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

Python

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

    # ============== Face Search ===============
    def StartFaceSearchCollection(self,collection):
        response = self.rek.start_face_search(Video={'S3Object':
{'Bucket':self.bucket,'Name':self.video}},
            CollectionId=collection,
            NotificationChannel={'RoleArn':self.roleArn,
 'SNSTopicArn':self.snsTopicArn})

        self.startJobId=response['JobId']

        print('Start Job Id: ' + self.startJobId)


    def GetFaceSearchCollectionResults(self):
        maxResults = 10
        paginationToken = ''

        finished = False
```

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

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

            for personMatch in response['Persons']:
                print('Person Index: ' + str(personMatch['Person']['Index']))
                print('Timestamp: ' + str(personMatch['Timestamp']))

                if ('FaceMatches' in personMatch):
                    for faceMatch in personMatch['FaceMatches']:
                        print('Face ID: ' + faceMatch['Face']['FaceId'])
                        print('Similarity: ' + str(faceMatch['Similarity']))
                print()
            if 'NextToken' in response:
                paginationToken = response['NextToken']
            else:
                finished = True
            print()
```

Ersetzen Sie in der Funktion `main` die folgenden Zeilen:

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

mit:

```
    collection='tests'
    analyzer.StartFaceSearchCollection(collection)

    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetFaceSearchCollectionResults()
```

Wenn Sie zusätzlich zu Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) bereits ein anderes Videobeispiel ausgeführt haben, ist der zu ersetzende Code möglicherweise anders.

5. Ändern Sie den Wert von `collection` in den Namen der Sammlung, die Sie in Schritt 1 erstellt haben.
6. Führen Sie den Code aus. Es wird eine Liste der Personen im Video angezeigt, deren Gesichter mit denen in der Eingabesammlung übereinstimmen. Zusätzlich werden die Tracking-Daten für jede gefundene Person angezeigt.

GetFaceSearch Antwort auf den Vorgang

Das folgende Beispiel ist eine JSON-Antwort von `GetFaceSearch`.

Die Antwort enthält ein Array von im Video erkannten Personen (`Persons`), deren Gesichter mit einem Gesicht in der Eingabesammlung übereinstimmen. Das Array -Element PersonMatch wird jedes Mal erzeugt, wenn die Person im Video erkannt wird. Jedes `PersonMatch` umfasst ein Array von Gesichtstreffern aus der Eingabesammlung, FaceMatch, Informationen über die gefundene Person, PersonDetail, und die Zeit, wann die Person im Video gefunden wurde.


```
{
    "JobStatus": "SUCCEEDED",
    "NextToken": "IJdbzkZfvBRqj8GPV82BPiZKkLOGCqDIsNZG/gQsEE5faTVK9JHOz/
xxxxxxxxxxxxxx",
    "Persons": [
        {
            "FaceMatches": [
                {
                    "Face": {
                        "BoundingBox": {
                            "Height": 0.527472972869873,
                            "Left": 0.33530598878860474,
                            "Top": 0.2161169946193695,
                            "Width": 0.35503000020980835
                        },
                        "Confidence": 99.90239715576172,
                        "ExternalImageId": "image.PNG",
                        "FaceId": "a2f2e224-bfaa-456c-b360-7c00241e5e2d",
                        "ImageId": "eb57ed44-8d8d-5ec5-90b8-6d190daff4c3"
```

```
                },
                "Similarity": 98.40909576416016
            }
        ],
        "Person": {
            "BoundingBox": {
                "Height": 0.8694444298744202,
                "Left": 0.2473958283662796,
                "Top": 0.10092592239379883,
                "Width": 0.49427083134651184
            },
            "Face": {
                "BoundingBox": {
                    "Height": 0.23000000417232513,
                    "Left": 0.42500001192092896,
                    "Top": 0.16333332657814026,
                    "Width": 0.12937499582767487
                },
                "Confidence": 99.97504425048828,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.46415066719055176,
                        "Y": 0.2572723925113678
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.5068183541297913,
                        "Y": 0.23705792427062988
                    },
                    {
                        "Type": "nose",
                        "X": 0.49765899777412415,
                        "Y": 0.28383663296699524
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.487221896648407,
                        "Y": 0.3452930748462677
                    },
                    {
                        "Type": "mouthRight",
                        "X": 0.5142884850502014,
                        "Y": 0.33167609572410583
```

```
                    }
                ],
                "Pose": {
                    "Pitch": 15.966927528381348,
                    "Roll": -15.547388076782227,
                    "Yaw": 11.34195613861084
                },
                "Quality": {
                    "Brightness": 44.80223083496094,
                    "Sharpness": 99.95819854736328
                }
            },
            "Index": 0
        },
        "Timestamp": 0
    },
    {
        "Person": {
            "BoundingBox": {
                "Height": 0.2177777737379074,
                "Left": 0.7593749761581421,
                "Top": 0.13333334028720856,
                "Width": 0.12250000238418579
            },
            "Face": {
                "BoundingBox": {
                    "Height": 0.2177777737379074,
                    "Left": 0.7593749761581421,
                    "Top": 0.13333334028720856,
                    "Width": 0.12250000238418579
                },
                "Confidence": 99.63436889648438,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.8005779385566711,
                        "Y": 0.20915353298187256
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.8391435146331787,
                        "Y": 0.21049551665782928
                    },
                    {
```

```
                        "Type": "nose",
                        "X": 0.8191410899162292,
                        "Y": 0.2523227035999298
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.8093273043632507,
                        "Y": 0.29053622484207153
                    },
                    {
                        "Type": "mouthRight",
                        "X": 0.8366993069648743,
                        "Y": 0.29101791977882385
                    }
                ],
                "Pose": {
                    "Pitch": 3.165884017944336,
                    "Roll": 1.4182015657424927,
                    "Yaw": -11.151537895202637
                },
                "Quality": {
                    "Brightness": 28.910892486572266,
                    "Sharpness": 97.61507415771484
                }
            },
            "Index": 1
        },
        "Timestamp": 0
    },
    {
        "Person": {
            "BoundingBox": {
                "Height": 0.8388888835906982,
                "Left": 0,
                "Top": 0.15833333134651184,
                "Width": 0.2369791716337204
            },
            "Face": {
                "BoundingBox": {
                    "Height": 0.20000000298023224,
                    "Left": 0.029999999329447746,
                    "Top": 0.2199999988079071,
                    "Width": 0.11249999701976776
                },
```

```
                "Confidence": 99.85971069335938,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.06842322647571564,
                        "Y": 0.3010137975215912
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.10543643683195114,
                        "Y": 0.29697132110595703
                    },
                    {
                        "Type": "nose",
                        "X": 0.09569807350635529,
                        "Y": 0.33701086044311523
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.0732642263174057,
                        "Y": 0.3757539987564087
                    },
                    {
                        "Type": "mouthRight",
                        "X": 0.10589495301246643,
                        "Y": 0.3722417950630188
                    }
                ],
                "Pose": {
                    "Pitch": -0.5589138865470886,
                    "Roll": -5.1093974113464355,
                    "Yaw": 18.69594955444336
                },
                "Quality": {
                    "Brightness": 43.052337646484375,
                    "Sharpness": 99.68138885498047
                }
            },
            "Index": 2
        },
        "Timestamp": 0
    }......

],
```

```
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 67301,
        "Format": "QuickTime / MOV",
        "FrameHeight": 1080,
        "FrameRate": 29.970029830932617,
        "FrameWidth": 1920
    }
}
```

Suchen nach Gesichtern in einer Sammlung im Streaming-Video

Sie können Amazon Rekognition Video verwenden, um Gesichter aus einer Sammlung in Streaming-Videos zu entdecken und zu erkennen. Mit Amazon Rekognition Video können Sie einen Stream-Prozessor (CreateStreamProcessor) erstellen, um die Analyse von Streaming-Videos zu starten und zu verwalten.

Um ein bekanntes Gesicht in einem Videostrom zu erkennen (Gesichtssuche), verwendet Amazon Rekognition Video Amazon Kinesis Video Streams, um einen Videostrom zu empfangen und zu verarbeiten. Die Analyseergebnisse werden von Amazon Rekognition Video in einen Kinesis-Datenstrom ausgegeben und dann von Ihrer Client-Anwendung gelesen.

Um Amazon Rekognition Video mit Streaming-Videos verwenden zu können, muss Ihre Anwendung das Folgende implementieren:

- Ein Kinesis-Videostrom zum Senden von Streaming-Videos an Amazon Rekognition Video. Weitere Informationen Sie im Entwicklerleitfaden für Amazon Kinesis Video Streams.
- Einen Amazon-Rekognition-Video-Stromprozessor zur Verwaltung der Analyse des Streaming-Videos. Weitere Informationen finden Sie unter Überblick über den Betrieb des Amazon-Rekognition-Video-Stromprozessors.
- Ein Kinesis-Datenstrom-Benutzer zum Lesen der Analyseergebnisse, die Amazon Rekognition Video an den Kinesis-Datenstrom sendet. Weitere Informationen finden Sie unter Kinesis-Datenströme-Verbraucher.

Dieser Abschnitt enthält Informationen zum Schreiben einer Anwendung, die den Kinesis-Videostrom und andere erforderliche Ressourcen erstellt, Videos in Amazon Rekognition Video streamt und die Analyseergebnisse empfängt.

Themen

- Einrichtung Ihrer Amazon-Rekognition-Video- und Amazon-Kinesis-Ressourcen
- Suche nach Gesichtern in einem Streaming-Video
- Streaming mit einem GStreamer Plugin
- Fehlerbehebung beim Streamen von Videos

Einrichtung Ihrer Amazon-Rekognition-Video- und Amazon-Kinesis-Ressourcen

In den folgenden Verfahren werden die Schritte beschrieben, die Sie ergreifen, um den Kinesis-Videostrom und andere Ressourcen bereitzustellen, die zur Erkennung von Gesichtern in einem Streaming-Video verwendet werden.

Voraussetzungen

Um dieses Verfahren ausführen zu können, müssen Sie den installiert haben. AWS SDK für Java Weitere Informationen finden Sie unter Erste Schritte mit Amazon Rekognition. Die von AWS-Konto Ihnen verwendete Person muss über Zugriffsberechtigungen für die Amazon Rekognition API verfügen. Weitere Informationen finden Sie unter Von Amazon Rekognition definierte Aktionen im IAM-Benutzerhandbuch.

So erkennen Sie Gesichter in einem Videostrom (AWS-SDK)

1. Falls Sie dies noch nicht getan haben, erstellen Sie eine IAM-Servicerolle, um Amazon Rekognition Video Zugriff auf Ihre Kinesis-Videoströme und Ihre Kinesis-Datenströme zu gewähren. Notieren Sie den ARN. Weitere Informationen finden Sie unter Zugriff auf Streams gewähren mit AmazonRekognitionServiceRole.
2. Erstellen Sie eine Sammlung und notieren Sie sich die ID der Sammlung, die Sie verwendet haben.
3. Indizieren Sie die Gesichter, nach denen Sie suchen möchten, in die Sammlung, die Sie in Schritt 2 erstellt haben.
4. Erstellen Sie einen Kinesis-Videostrom und notieren Sie sich den Amazon-Ressourcennamen (ARN) des Streams.
5. Erstellen Sie einen Kinesis-Datenstrom. Stellen Sie dem Stream-Namen den ARN des Streams voran AmazonRekognitionund notieren Sie sich diesen.

Anschließend können Sie den Gesichtssuche-Stromprozessor erstellen und den Stromprozessor mit dem von Ihnen ausgewählten Stromprozessornamen starten.

> **Note**
>
> Sie sollten den Stromprozessor erst starten, nachdem Sie sich vergewissert haben, dass Sie Medien in den Kinesis-Videostrom aufnehmen können.

Streaming-Video von Amazon Rekognition Video

Um Video in Amazon Rekognition Video zu streamen, verwenden Sie das Amazon Kinesis Video Streams SDK, um einen Kinesis-Videostrom zu erstellen und zu verwenden. Die `PutMedia`-Operation schreibt Videodatenfragmente in einen Kinesis-Videostrom, der von Amazon Rekognition Video genutzt wird. Jedes Videodatenfragment ist typischerweise 2–10 Sekunden lang und enthält eine in sich geschlossene Sequenz von Videobildern. Amazon Rekognition Video unterstützt H.264-kodierte Videos, die drei Arten von Frames (I, B und P) enthalten können. Weitere Informationen finden Sie unter Inter Frame. Der erste Frame im Fragment muss ein I-Frame sein. Ein I-Frame kann unabhängig von anderen Frames entschlüsselt werden.

Wenn Videodaten im Kinesis-Videostrom ankommen, weist Kinesis Video Streams dem Fragment eine eindeutige Nummer zu. Ein Beispiel finden Sie unter PutMedia API-Beispiel.

- Wenn Sie von einer Matroska (MKV) -codierten Quelle streamen, verwenden Sie den PutMediaVorgang, um das Quellvideo in den Kinesis-Videostream zu streamen, den Sie erstellt haben. Weitere Informationen finden Sie unter API-Beispiel. PutMedia
- Wenn Sie von einer Gerätekamera streamen, finden Sie weitere Informationen unter Streaming mit einem GStreamer Plugin.

Gewähren von Zugriff auf Ihre Ressourcen für Amazon Rekognition Video

Sie verwenden eine AWS Identity and Access Management (IAM) -Servicerolle, um Amazon Rekognition Video Lesezugriff auf Kinesis-Videostreams zu gewähren. Wenn Sie einen Gesichts-Stromprozessor verwenden, verwenden Sie eine IAM-Servicerolle, um Amazon Rekognition Video Schreibzugriff auf Kinesis-Datenströme zu gewähren. Wenn Sie einen Stromprozessor für die Sicherheitsüberwachung verwenden, verwenden Sie IAM-Rollen, um Amazon Rekognition Video Zugriff auf Ihren Amazon-S3-Bucket und auf ein Amazon-SNS-Thema zu gewähren.

Zugriff für Gesichtssuche-Stromprozessoren gewähren

Sie können eine Berechtigungsrichtlinie erstellen, die Amazon Rekognition Video den Zugriff auf einzelne Kinesis-Videoströme und Kinesis-Datenströme ermöglicht.

Um Amazon Rekognition Video Zugriff auf einen Stromprozessor für die Gesichtssuche zu gewähren

1. Erstellen Sie mit dem IAM JSON Policy Editor eine neue Berechtigungsrichtlinie und verwenden Sie die folgende Richtlinie. Ersetzen Sie `video-arn` durch den ARN des gewünschten Kinesis-Videostroms. Wenn Sie einen Gesichtssuche-Stromprozessor verwenden, ersetzen Sie `data-arn` durch den ARN des gewünschten Kinesis-Datenstroms.

```
{
    "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. Erstellen Sie ein IAM-Servicerolle oder aktualisieren Sie eine bestehende IAM-Servicerolle. Verwenden Sie die folgenden Informationen, um die IAM-Servicerolle zu erstellen:

    1. Wählen Sie Rekognition als Servicenamen.
    2. Wählen Sie Rekognition für den Anwendungsfall der Service-Rolle.
    3. Fügen Sie die Berechtigungsrichtlinie hinzu, die Sie in Schritt 1 erstellt haben.

3. Notieren Sie den ARN der Service-Rolle. Sie benötigen ihn, um Videoanalyse-Operationen zu starten.

Zugriff auf Streams gewähren mit AmazonRekognitionServiceRole

Als alternative Option für die Einrichtung des Zugriffs auf Kinesis-Videoströme und Datenströme können Sie die `AmazonRekognitionServiceRole`-Berechtigungsrichtlinie verwenden. IAM bietet den Anwendungsfall der Rekognition-Servicerolle, die in Verbindung mit der `AmazonRekognitionServiceRole`-Berechtigungsrichtlinie in mehrere Kinesis-Datenströme schreiben und aus allen Ihren Kinesis-Videoströmen lesen kann. Um Amazon Rekognition Video Schreibzugriff auf mehrere Kinesis-Datenstreams zu gewähren, können Sie den Namen der Kinesis-Datenstreams Folgendes voranstellen — zum Beispiel. AmazonRekognition`AmazonRekognitionMyDataStreamName`

Um Amazon Rekognition Video Zugriff auf Ihren Kinesis-Videostrom und Kinesis-Datenstrom zu gewähren

1. Erstellen Sie eine IAM-Servicerolle. Verwenden Sie die folgenden Informationen, um die IAM-Servicerolle zu erstellen:

    1. Wählen Sie Rekognition als Servicenamen.
    2. Wählen Sie Rekognition für den Anwendungsfall der Service-Rolle.
    3. Wählen Sie die AmazonRekognitionServiceRoleBerechtigungsrichtlinie, die Amazon Rekognition Video Schreibzugriff auf Kinesis-Datenstreams mit Präfix AmazonRekognitionund Lesezugriff auf all Ihre Kinesis-Videostreams gewährt.
2. Um sicherzustellen, AWS-Konto dass Sie sicher sind, beschränken Sie den Zugriff von Rekognition auf die Ressourcen, die Sie verwenden. Dies kann erreicht werden, indem Sie Ihrer IAM-Servicerolle eine Vertrauensrichtlinie hinzufügen. Weitere Informationen hierzu finden Sie unter Serviceübergreifende Confused-Deputy-Prävention.
3. Notieren Sie sich den Amazon-Ressourcennamen (ARN) der Service-Rolle. Sie benötigen ihn, um Videoanalyse-Operationen zu starten.

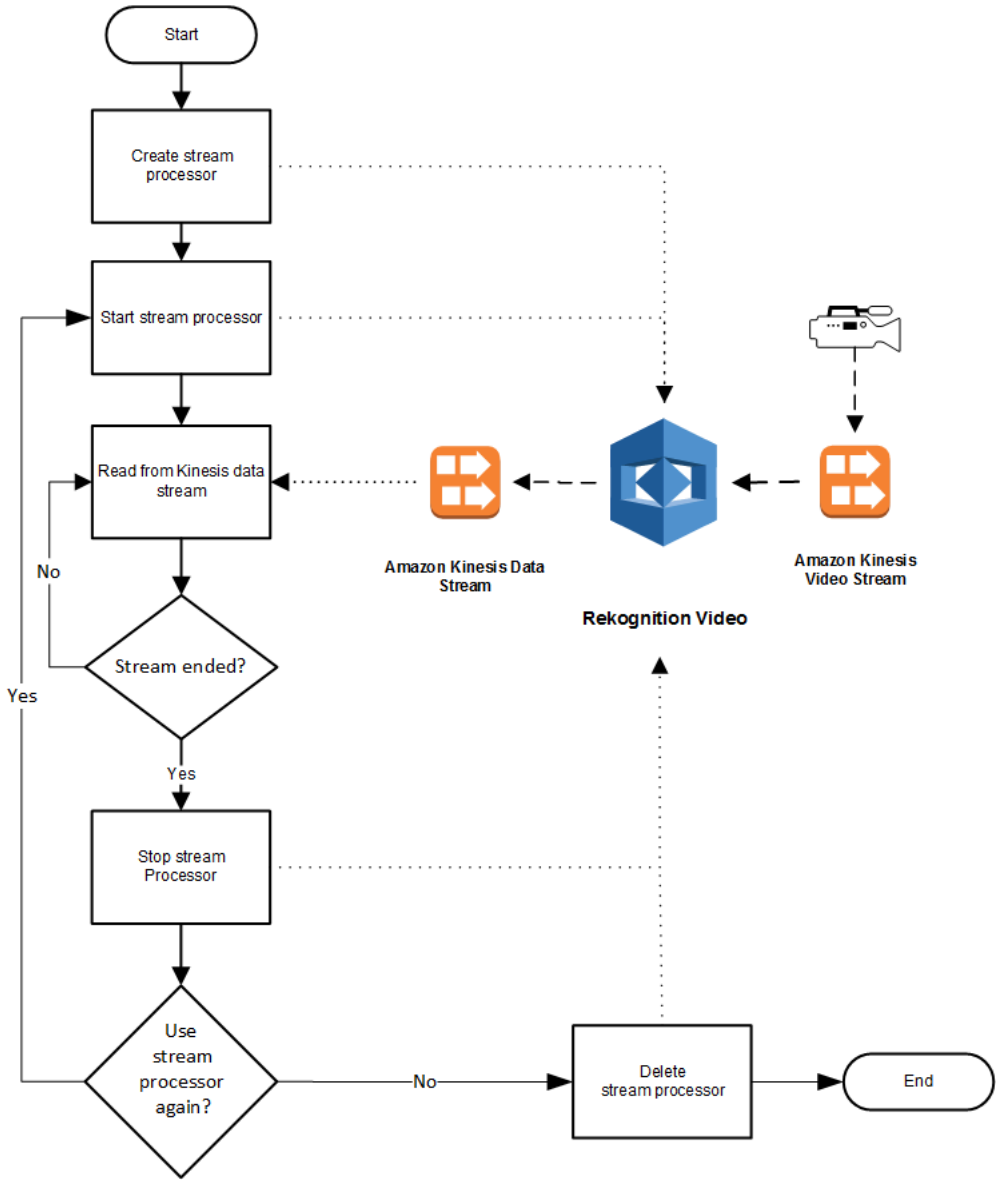
Suche nach Gesichtern in einem Streaming-Video

Amazon Rekognition Video kann Gesichter in einer Sammlung suchen, die mit Gesichtern übereinstimmen, die in einem Streaming-Video erkannt werden. Weitere Informationen über Sammlungen finden Sie unter Gesichtssuche in einer Sammlung.

Themen

- Erstellen des Amazon-Rekognition-Video-Gesichtssuche-Stromprozessors
- Starten des Amazon-Rekognition-Video-Gesichtssuche-Stromprozessors
- Verwendung von Stromprozessoren für die Gesichtssuche (Beispiel Java V2)
- Verwendung von Stromprozessoren für die Gesichtssuche (Beispiel Java V1)
- Lesen von Streaming-Video-Analyseergebnissen
- Lokales Anzeigen von Rekognition-Ergebnissen mit Kinesis Video Streams
- Den JSON-Frame-Datensatz für Kinesis-Gesichtserkennung verstehen

Das folgende Diagramm zeigt, wie Amazon Rekognition Video Gesichter in einem Streaming-Video entdeckt und erkennt.



Erstellen des Amazon-Rekognition-Video-Gesichtssuche-Stromprozessors

Bevor Sie ein Streaming-Video analysieren können, erstellen Sie einen Amazon Rekognition Video Video-Stream-Prozessor () CreateStreamProcessor. Der Stromprozessor enthält Informationen über den Kinesis-Datenstrom und den Kinesis-Videostrom. Enthalten ist auch die ID für die Sammlung der Gesichter, die Sie im bereitgestellten Streaming-Video erkennen möchten. Außerdem geben Sie einen Namen für den Stromprozessor an. Im Folgenden sehen Sie ein JSON-Beispiel für die Anforderung von `CreateStreamProcessor`.

```
{
	"Name": "streamProcessorForCam",
	"Input": {
```

```
                "KinesisVideoStream": {
                        "Arn": "arn:aws:kinesisvideo:us-east-1:nnnnnnnnnnnn:stream/
inputVideo"
                }
        },
        "Output": {
                "KinesisDataStream": {
                        "Arn": "arn:aws:kinesis:us-east-1:nnnnnnnnnnnn:stream/outputData"
                }
        },
        "RoleArn": "arn:aws:iam::nnnnnnnnnnn:role/roleWithKinesisPermission",
        "Settings": {
                "FaceSearch": {
                        "CollectionId": "collection-with-100-faces",
                        "FaceMatchThreshold": 85.5
                }
        }
}
```

Das folgende Beispiel ist eine Antwort von `CreateStreamProcessor`.

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

Starten des Amazon-Rekognition-Video-Gesichtssuche-Stromprozessors

Sie beginnen die Analyse des Streaming-Videos, indem Sie StartStreamProcessor mit dem Streaming-Prozessor-Namen aufrufen, den Sie in `CreateStreamProcessor` festgelegt haben. Im Folgenden sehen Sie ein JSON-Beispiel für die Anforderung von `StartStreamProcessor`.

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

Wenn der Stromprozessor erfolgreich beginnt, wird eine HTTP-200-Antwort zurückgegeben, zusammen mit einem leeren JSON-Text.

Verwendung von Stromprozessoren für die Gesichtssuche (Beispiel Java V2)

Der folgende Beispielcode zeigt, wie verschiedene Streamprozessor-Operationen, wie CreateStreamProcessorund StartStreamProcessor, mit dem AWS SDK for Java Version 2 aufgerufen werden.

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateStreamProcessorRequest;
import software.amazon.awssdk.services.rekognition.model.CreateStreamProcessorResponse;
import software.amazon.awssdk.services.rekognition.model.FaceSearchSettings;
import software.amazon.awssdk.services.rekognition.model.KinesisDataStream;
import software.amazon.awssdk.services.rekognition.model.KinesisVideoStream;
import software.amazon.awssdk.services.rekognition.model.ListStreamProcessorsRequest;
import software.amazon.awssdk.services.rekognition.model.ListStreamProcessorsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.StreamProcessor;
import software.amazon.awssdk.services.rekognition.model.StreamProcessorInput;
import software.amazon.awssdk.services.rekognition.model.StreamProcessorSettings;
import software.amazon.awssdk.services.rekognition.model.StreamProcessorOutput;
import software.amazon.awssdk.services.rekognition.model.StartStreamProcessorRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeStreamProcessorRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeStreamProcessorResponse;

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

```
                Usage:    <role> <kinInputStream> <kinOutputStream> <collectionName>
<StreamProcessorName>

                Where:
                   role - The ARN of the AWS Identity and Access Management (IAM) role
to use. \s
                   kinInputStream - The ARN of the Kinesis video stream.\s
                   kinOutputStream - The ARN of the Kinesis data stream.\s
                   collectionName - The name of the collection to use that contains
content. \s
                   StreamProcessorName - The name of the Stream Processor. \s
                """;

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

       String role = args[0];
       String kinInputStream = args[1];
       String kinOutputStream = args[2];
       String collectionName = args[3];
       String streamProcessorName = args[4];

       Region region = Region.US_EAST_1;
       RekognitionClient rekClient = RekognitionClient.builder()
               .region(region)
               .build();

       processCollection(rekClient, streamProcessorName, kinInputStream,
kinOutputStream, collectionName,
               role);
       startSpecificStreamProcessor(rekClient, streamProcessorName);
       listStreamProcessors(rekClient);
       describeStreamProcessor(rekClient, streamProcessorName);
       deleteSpecificStreamProcessor(rekClient, streamProcessorName);
   }

   public static void listStreamProcessors(RekognitionClient rekClient) {
       ListStreamProcessorsRequest request = ListStreamProcessorsRequest.builder()
               .maxResults(15)
               .build();
```

```
       ListStreamProcessorsResponse listStreamProcessorsResult =
rekClient.listStreamProcessors(request);
       for (StreamProcessor streamProcessor :
listStreamProcessorsResult.streamProcessors()) {
           System.out.println("StreamProcessor name - " + streamProcessor.name());
           System.out.println("Status - " + streamProcessor.status());
       }
   }

   private static void describeStreamProcessor(RekognitionClient rekClient, String
StreamProcessorName) {
       DescribeStreamProcessorRequest streamProcessorRequest =
DescribeStreamProcessorRequest.builder()
               .name(StreamProcessorName)
               .build();

       DescribeStreamProcessorResponse describeStreamProcessorResult = rekClient
               .describeStreamProcessor(streamProcessorRequest);
       System.out.println("Arn - " +
describeStreamProcessorResult.streamProcessorArn());
       System.out.println("Input kinesisVideo stream - "
               + describeStreamProcessorResult.input().kinesisVideoStream().arn());
       System.out.println("Output kinesisData stream - "
               + describeStreamProcessorResult.output().kinesisDataStream().arn());
       System.out.println("RoleArn - " + describeStreamProcessorResult.roleArn());
       System.out.println(
               "CollectionId - "
                       +
describeStreamProcessorResult.settings().faceSearch().collectionId());
       System.out.println("Status - " + describeStreamProcessorResult.status());
       System.out.println("Status message - " +
describeStreamProcessorResult.statusMessage());
       System.out.println("Creation timestamp - " +
describeStreamProcessorResult.creationTimestamp());
       System.out.println("Last update timestamp - " +
describeStreamProcessorResult.lastUpdateTimestamp());
   }

   private static void startSpecificStreamProcessor(RekognitionClient rekClient,
String StreamProcessorName) {
       try {
           StartStreamProcessorRequest streamProcessorRequest =
StartStreamProcessorRequest.builder()
                   .name(StreamProcessorName)
```

```
                    .build();

            rekClient.startStreamProcessor(streamProcessorRequest);
            System.out.println("Stream Processor " + StreamProcessorName + "
started.");

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

    private static void processCollection(RekognitionClient rekClient, String
StreamProcessorName,
                                          String kinInputStream, String
kinOutputStream, String collectionName, String role) {
        try {
            KinesisVideoStream videoStream = KinesisVideoStream.builder()
                    .arn(kinInputStream)
                    .build();

            KinesisDataStream dataStream = KinesisDataStream.builder()
                    .arn(kinOutputStream)
                    .build();

            StreamProcessorOutput processorOutput = StreamProcessorOutput.builder()
                    .kinesisDataStream(dataStream)
                    .build();

            StreamProcessorInput processorInput = StreamProcessorInput.builder()
                    .kinesisVideoStream(videoStream)
                    .build();

            FaceSearchSettings searchSettings = FaceSearchSettings.builder()
                    .faceMatchThreshold(75f)
                    .collectionId(collectionName)
                    .build();

            StreamProcessorSettings processorSettings =
StreamProcessorSettings.builder()
                    .faceSearch(searchSettings)
                    .build();
```

```
            CreateStreamProcessorRequest processorRequest =
 CreateStreamProcessorRequest.builder()
                    .name(StreamProcessorName)
                    .input(processorInput)
                    .output(processorOutput)
                    .roleArn(role)
                    .settings(processorSettings)
                    .build();

            CreateStreamProcessorResponse response =
 rekClient.createStreamProcessor(processorRequest);
            System.out.println("The ARN for the newly create stream processor is "
                    + response.streamProcessorArn());

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

    private static void deleteSpecificStreamProcessor(RekognitionClient rekClient,
 String StreamProcessorName) {
        rekClient.stopStreamProcessor(a -> a.name(StreamProcessorName));
        rekClient.deleteStreamProcessor(a -> a.name(StreamProcessorName));
        System.out.println("Stream Processor " + StreamProcessorName + " deleted.");
    }
}
```

Verwendung von Stromprozessoren für die Gesichtssuche (Beispiel Java V1)

Der folgende Beispielcode zeigt, wie verschiedene Streamprozessor-Operationen wie CreateStreamProcessorund StartStreamProcessormit Java V1 aufgerufen werden. Das Beispiel beinhaltet eine Streamprozessor-Manager-Klasse (StreamManager), die Methoden zum Aufrufen von Streamprozessor-Operationen bereitstellt. Die Starterklasse (Starter) erstellt ein StreamManager Objekt und ruft verschiedene Operationen auf.

So konfigurieren Sie das Beispiel:

1. Legen Sie die Werte der Mitgliedsfelder der Starterklasse auf Ihre gewünschten Werte fest.
2. Entfernen Sie den gewünschten Funktionsaufruf in der Starterklassenfunktion `main`.

Starterklasse

```
//Copyright 2018 Amazon.com, Inc. or its 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 Klasse

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

// Stream manager class. Provides methods for calling
// Stream Processor operations.
package com.amazonaws.samples;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.CreateStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.CreateStreamProcessorResult;
import com.amazonaws.services.rekognition.model.DeleteStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.DeleteStreamProcessorResult;
import com.amazonaws.services.rekognition.model.DescribeStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.DescribeStreamProcessorResult;
import com.amazonaws.services.rekognition.model.FaceSearchSettings;
import com.amazonaws.services.rekognition.model.KinesisDataStream;
import com.amazonaws.services.rekognition.model.KinesisVideoStream;
import com.amazonaws.services.rekognition.model.ListStreamProcessorsRequest;
import com.amazonaws.services.rekognition.model.ListStreamProcessorsResult;
import com.amazonaws.services.rekognition.model.StartStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.StartStreamProcessorResult;
import com.amazonaws.services.rekognition.model.StopStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.StopStreamProcessorResult;
import com.amazonaws.services.rekognition.model.StreamProcessor;
import com.amazonaws.services.rekognition.model.StreamProcessorInput;
import com.amazonaws.services.rekognition.model.StreamProcessorOutput;
import com.amazonaws.services.rekognition.model.StreamProcessorSettings;

public class StreamManager {

    private String streamProcessorName;
    private String kinesisVideoStreamArn;
    private String kinesisDataStreamArn;
    private String roleArn;
    private String collectionId;
    private float matchThreshold;
```

```
    private AmazonRekognition rekognitionClient;


    public StreamManager(String spName,
      String kvStreamArn,
      String kdStreamArn,
      String iamRoleArn,
      String collId,
      Float threshold){
     streamProcessorName=spName;
     kinesisVideoStreamArn=kvStreamArn;
     kinesisDataStreamArn=kdStreamArn;
     roleArn=iamRoleArn;
     collectionId=collId;
     matchThreshold=threshold;
     rekognitionClient=AmazonRekognitionClientBuilder.defaultClient();

    }

    public void createStreamProcessor() {
     //Setup input parameters
        KinesisVideoStream kinesisVideoStream = new
KinesisVideoStream().withArn(kinesisVideoStreamArn);
        StreamProcessorInput streamProcessorInput =
                new StreamProcessorInput().withKinesisVideoStream(kinesisVideoStream);
        KinesisDataStream kinesisDataStream = new
KinesisDataStream().withArn(kinesisDataStreamArn);
        StreamProcessorOutput streamProcessorOutput =
                new StreamProcessorOutput().withKinesisDataStream(kinesisDataStream);
        FaceSearchSettings faceSearchSettings =
                new
FaceSearchSettings().withCollectionId(collectionId).withFaceMatchThreshold(matchThreshold);
        StreamProcessorSettings streamProcessorSettings =
                new StreamProcessorSettings().withFaceSearch(faceSearchSettings);

        //Create the stream processor
        CreateStreamProcessorResult createStreamProcessorResult =
rekognitionClient.createStreamProcessor(
                new
CreateStreamProcessorRequest().withInput(streamProcessorInput).withOutput(streamProcessorOutpu

.withSettings(streamProcessorSettings).withRoleArn(roleArn).withName(streamProcessorName));

        //Display result
```

```
        System.out.println("Stream Processor " + streamProcessorName + " created.");
        System.out.println("StreamProcessorArn - " +
createStreamProcessorResult.getStreamProcessorArn());
    }

    public void startStreamProcessor() {
        StartStreamProcessorResult startStreamProcessorResult =
                rekognitionClient.startStreamProcessor(new
StartStreamProcessorRequest().withName(streamProcessorName));
        System.out.println("Stream Processor " + streamProcessorName + " started.");
    }

    public void stopStreamProcessor() {
        StopStreamProcessorResult stopStreamProcessorResult =
                rekognitionClient.stopStreamProcessor(new
StopStreamProcessorRequest().withName(streamProcessorName));
        System.out.println("Stream Processor " + streamProcessorName + " stopped.");
    }

    public void deleteStreamProcessor() {
        DeleteStreamProcessorResult deleteStreamProcessorResult = rekognitionClient
                .deleteStreamProcessor(new
DeleteStreamProcessorRequest().withName(streamProcessorName));
        System.out.println("Stream Processor " + streamProcessorName + " deleted.");
    }

    public void describeStreamProcessor() {
        DescribeStreamProcessorResult describeStreamProcessorResult = rekognitionClient
                .describeStreamProcessor(new
DescribeStreamProcessorRequest().withName(streamProcessorName));

        //Display various stream processor attributes.
        System.out.println("Arn - " +
describeStreamProcessorResult.getStreamProcessorArn());
        System.out.println("Input kinesisVideo stream - "
                +
describeStreamProcessorResult.getInput().getKinesisVideoStream().getArn());
        System.out.println("Output kinesisData stream - "
                +
describeStreamProcessorResult.getOutput().getKinesisDataStream().getArn());
        System.out.println("RoleArn - " + describeStreamProcessorResult.getRoleArn());
        System.out.println(
                "CollectionId - " +
describeStreamProcessorResult.getSettings().getFaceSearch().getCollectionId());
```

```
        System.out.println("Status - " + describeStreamProcessorResult.getStatus());
        System.out.println("Status message - " +
  describeStreamProcessorResult.getStatusMessage());
        System.out.println("Creation timestamp - " +
  describeStreamProcessorResult.getCreationTimestamp());
        System.out.println("Last update timestamp - " +
  describeStreamProcessorResult.getLastUpdateTimestamp());
    }

    public void listStreamProcessors() {
        ListStreamProcessorsResult listStreamProcessorsResult =
                rekognitionClient.listStreamProcessors(new
  ListStreamProcessorsRequest().withMaxResults(100));

        //List all stream processors (and state) returned from Rekognition
        for (StreamProcessor streamProcessor :
  listStreamProcessorsResult.getStreamProcessors()) {
            System.out.println("StreamProcessor name - " + streamProcessor.getName());
            System.out.println("Status - " + streamProcessor.getStatus());
        }
    }
}
```

Lesen von Streaming-Video-Analyseergebnissen

Sie können die Amazon-Kinesis-Data-Streams-Client-Bibliothek verwenden, um Analyseergebnisse zu verwenden, die an den Amazon-Kinesis-Data-Streams-Ausgabestrom gesendet werden. Weitere Informationen finden Sie unter Lesen von Daten aus einem Kinesis-Datenstrom. Amazon Rekognition Video stellt für jeden analysierten Frame einen JSON-Frame-Datensatz in den Kinesis-Ausgabestrom. Amazon Rekognition Video analysiert nicht jedes Bild, das über den Kinesis-Videostrom an Amazon Rekognition Video weitergegeben wird.

Ein Frame-Datensatz, der an einen Kinesis-Datenstrom gesendet wird, enthält Informationen darüber, in welchem Kinesis-Videostrom-Fragment sich der Frame befindet, wo im Fragment der Frame ist und die Gesichter, die im Frame erkannt werden. Es sind auch Statusinformationen für den Stromprozessor enthalten. Weitere Informationen finden Sie unter Den JSON-Frame-Datensatz für Kinesis-Gesichtserkennung verstehen.

Die Amazon-Kinesis-Video-Streams-Parser-Bibliothek enthält Beispieltests, die Amazon-Rekognition-Video-Ergebnisse verwenden, und integriert sie in den ursprünglichen Kinesis-Videostrom. Weitere

Informationen finden Sie unter Lokales Anzeigen von Rekognition-Ergebnissen mit Kinesis Video Streams.

Amazon Rekognition Video streamt Analyseinformationen von Amazon Rekognition Video in den Kinesis-Datenstrom. Im Folgenden sehen Sie ein JSON-Beispiel für eine einzelne Aufzeichnung.


```
{
  "InputInformation": {
    "KinesisVideo": {
      "StreamArn": "arn:aws:kinesisvideo:us-west-2:nnnnnnnnnnnn:stream/stream-name",
      "FragmentNumber": "91343852333289682796718532614445757584843717598",
      "ServerTimestamp": 1510552593.455,
      "ProducerTimestamp": 1510552593.193,
      "FrameOffsetInSeconds": 2
    }
  },
  "StreamProcessorInformation": {
    "Status": "RUNNING"
  },
  "FaceSearchResponse": [
    {
      "DetectedFace": {
        "BoundingBox": {
          "Height": 0.075,
          "Width": 0.05625,
          "Left": 0.428125,
          "Top": 0.40833333
        },
        "Confidence": 99.975174,
        "Landmarks": [
          {
            "X": 0.4452057,
            "Y": 0.4395594,
            "Type": "eyeLeft"
          },
          {
            "X": 0.46340984,
            "Y": 0.43744427,
            "Type": "eyeRight"
          },
          {
            "X": 0.45960626,
            "Y": 0.4526856,
```

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

Beachten Sie im JSON-Beispiel Folgendes:

- InputInformation— Informationen über den Kinesis-Videostream, der zum Streamen von Videos in Amazon Rekognition Video verwendet wird. Weitere Informationen finden Sie unter InputInformation.
- StreamProcessorInformation— Statusinformationen für den Amazon Rekognition Video Video-Stream-Prozessor. Der einzige mögliche Wert für das Feld `Status` ist RUNNING. Weitere Informationen finden Sie unter StreamProcessorInformation.
- FaceSearchResponse— Enthält Informationen zu Gesichtern im Streaming-Video, die mit Gesichtern in der Eingabesammlung übereinstimmen. FaceSearchResponseenthält ein DetectedFace Objekt, bei dem es sich um ein Gesicht handelt, das im analysierten Videoframe erkannt wurde. Für jedes erkannte Gesicht enthält das Array `MatchedFaces` ein Array mit übereinstimmenden Gesichtsobjekten (MatchedFace), die in der Eingabesammlung gefunden wurden, ebenso wie einen Ähnlichkeitswert.

Zuweisen des Kinesis-Videoströme zum Kinesis-Datenstrom

Möglicherweise möchten Sie die Bilder des Kinesis-Videostroms den analysierten Bildern zuordnen, die an den Kinesis-Datenstrom gesendet werden. Beispielsweise möchten Sie während der Anzeige eines Streaming-Videos vielleicht, dass die Gesichter erkannter Personen eingerahmt werden. Die Begrenzungsrahmen-Koordinaten werden als Teil des Kinesis-Gesichtserkennungsdatensatzes an den Kinesis-Datenstrom gesendet. Damit der Begrenzungsrahmen richtig angezeigt wird, müssen Sie die Zeitinformationen, die zusammen mit der Kinesis-Gesichtserkennungsaufzeichnung gesendet werden, den entsprechenden Frames im Quell-Kinesis-Videostrom zuordnen.

Welche Methode Sie für die Zuweisung des Kinesis-Videostroms zum Kinesis-Datenstrom verwenden, hängt davon ab, ob Live-Medien (z. B. ein Live-Streaming-Video) oder archivierte Medien (z. B. ein gespeichertes Video) gestromt werden.

Zuweisung beim Streamen von Live-Medien

So ordnen Sie einen Kinesis-Videostrom-Frame einem Kinesis-Datenstrom-Frame zu

1. Stellen Sie den Eingabeparameter `FragmentTimeCodeType` der PutMediaOperation auf ein`RELATIVE`.
2. Rufen Sie `PutMedia` an, um Live-Medien in den Kinesis-Videostrom zu übertragen.

3. Wenn Sie einen Kinesis-Gesichtserkennungsdatensatz aus dem Kinesis-Datenstrom erhalten, speichern Sie die Werte von `ProducerTimestamp` und `FrameOffsetInSeconds` aus dem KinesisVideo-Feld.
4. Berechnen Sie den Zeitstempel, der dem Kinesis-Videostrom-Frame entspricht, indem Sie die `ProducerTimestamp`- und `FrameOffsetInSeconds`-Feldwerte addieren.

Zuweisung beim Streamen archivierter Medien

So ordnen Sie einen Kinesis-Videostrom-Frame einem Kinesis-Datenstrom-Frame zu

1. Rufen Sie PutMediaan, um archivierte Medien in den Kinesis-Videostream zu übertragen.
2. Wenn Sie als Antwort auf die `Acknowledgement`-Operation ein `PutMedia`-Objekt erhalten, speichern Sie den `FragmentNumber`-Wert aus dem Feld Nutzlast. `FragmentNumber` ist die Fragmentnummer des MKV-Clusters.
3. Wenn Sie einen Kinesis-Gesichtserkennungsdatensatz aus dem Kinesis-Datenstrom erhalten, speichern Sie den `FrameOffsetInSeconds`-Feldwert aus dem KinesisVideo-Feld.
4. Berechnen Sie die Zuweisung anhand der von Ihnen in den Schritten 2 und 3 gespeicherten Werte von `FrameOffsetInSeconds` und `FragmentNumber`. `FrameOffsetInSeconds` ist der Versatz innerhalb des Fragments mit der spezifischen `FragmentNumber`, die zum für Amazon-Kinesis-Datenstrom gesendet wurde. Weitere Informationen zum Abrufen der Video-Frames für eine bestimmte Fragmentnummer finden Sie im Thema zu Archivierte Medien in Amazon Kinesis Video Streams.

Lokales Anzeigen von Rekognition-Ergebnissen mit Kinesis Video Streams

Sie können die Ergebnisse von Amazon Rekognition Video in Ihrem Feed von Amazon Kinesis Video Streams anhand der Beispieltests der Amazon Kinesis Video Streams Parser Library sehen, die Sie unter — Rekognition Examples finden. KinesisVideo `KinesisVideoRekognitionIntegrationExample`Zeigt Begrenzungsrahmen über erkannten Gesichtern an und rendert das Video lokal durch. JFrame Diese Operation setzt voraus, dass Sie erfolgreich eine Medieneingabe von einer Gerätekamera mit einem Kinesis-Videostrom verbunden und einen Amazon-Rekognition-Stromprozessor gestartet haben. Weitere Informationen finden Sie unter Streaming mit einem GStreamer Plugin.

Schritt 1: Installieren der Kinesis-Videostrom-Parser-Bibliothek

Um ein Verzeichnis zu erstellen und das GitHub-Repository herunterzuladen, führen Sie den folgenden Befehl aus:

```
$ git clone https://github.com/aws/amazon-kinesis-video-streams-parser-library.git
```

Navigieren Sie zum Bibliotheksverzeichnis und führen Sie den folgenden Maven-Befehl aus, um eine Neuinstallation durchzuführen:

```
$ mvn clean install
```

Schritt 2: Konfiguration des Beispieltests zur Integration von Kinesis-Videoströmen und Rekognition

Öffnen Sie die `KinesisVideoRekognitionIntegrationExampleTest.java` Datei. Entfernen Sie das `@Ignore` gleich nach dem Klassen-Header. Füllen Sie die Datenfelder mit den Informationen aus Ihren Amazon-Kinesis- und Amazon-Rekognition-Ressourcen. Weitere Informationen finden Sie unter Einrichtung Ihrer Amazon-Rekognition-Video- und Amazon-Kinesis-Ressourcen. Wenn Sie Video in Ihren Kinesis-Videostrom streamen, entfernen Sie den `inputStream`-Parameter.

Beachten Sie hierzu das folgende Codebeispiel:

```
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();
```

Schritt 3: Ausführen des Beispieltests zur Integration von Kinesis-Videoströmen und Rekognition

Stellen Sie sicher, dass Ihr Kinesis-Videostrom Medieneingaben empfängt, wenn Sie zu ihm streamen, und beginnen Sie mit der Analyse Ihres Streams, während ein Amazon-Rekognition-Video-Stromprozessor läuft. Weitere Informationen finden Sie unter Überblick über den Betrieb des Amazon-Rekognition-Video-Stromprozessors. Führen Sie den `KinesisVideoRekognitionIntegrationExampleTest` Kurs als Test aus. JUnit Nach einer kurzen Verzögerung öffnet sich ein neues Fenster mit einem Video-Feed aus Ihrem Kinesis-Videostrom mit Begrenzungsrahmen, die über erkannten Gesichtern gezogen werden.

> **Note**
>
> Für die Gesichter in der in diesem Beispiel verwendeten Sammlung muss die externe Bild-ID (der Dateiname) in diesem Format angegeben sein, damit Begrenzungsfeldbeschriftungen aussagekräftigen Text anzeigen können: PersonName 1-Trusted, PersonName 2-Intruder, PersonName 3-Neutral usw. Die Beschriftungen können auch farblich gekennzeichnet werden und sind in der Java-Datei anpassbar. FaceType

Den JSON-Frame-Datensatz für Kinesis-Gesichtserkennung verstehen

Amazon Rekognition Video kann Gesichter in einem Streaming-Video erkennen. Für jedes analysierte Bild gibt Amazon Rekognition Video einen JSON-Bilddatensatz an einen Kinesis-Datenstrom aus. Amazon Rekognition Video analysiert nicht jedes Bild, das über den Kinesis-Videostrom an Amazon Rekognition Video weitergegeben wird.

Der JSON-Frame-Datensatz enthält Informationen über den Input- und Output-Stream, den Status des Stromprozessors und Informationen über Gesichter, die im analysierten Frame erkannt wurden. Dieser Abschnitt enthält Referenzinformationen für den JSON-Frame-Datensatz.

Im Folgenden finden Sie die JSON-Syntax für einen Kinesis-Datenstromdatensatz. Weitere Informationen finden Sie unter Arbeiten mit Streaming-Videoereignissen.

> **Note**
>
> Die Amazon-Rekognition-Video-API vergleicht die Gesichter in Ihrem Eingabestrom mit einer Sammlung von Gesichtern und gibt die größten gefundenen Übereinstimmungen zusammen mit einem Ähnlichkeitswert zurück.

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

JSON-Datensatz

Der JSON-Datensatz enthält Informationen über den Frame, der von Amazon Rekognition Video verarbeitet wurde. Der Datensatz enthält Informationen über das Streaming-Video, den Status des analysierten Frames und Informationen über Gesichter, die im analysierten Frame erkannt wurden.

InputInformation

Informationen über den Kinesis-Videostrom, der zum Streamen von Videos in Amazon Rekognition Video verwendet wird.

Typ: InputInformation Objekt

StreamProcessorInformation

Informationen über den Amazon-Rekognition-Video-Stromprozessor. Dies umfasst Statusinformationen über den aktuellen Status des Stromprozessors.

Typ: StreamProcessorInformation Objekt

FaceSearchResponse

Informationen über die erkannten Gesichter in einem Streaming-Video-Frame und die übereinstimmenden Gesichter, die in der Eingabesammlung gefunden wurden.

Typ: FaceSearchResponse Objekt-Array

InputInformation

Informationen über einen Quellvideostrom, der von Amazon Rekognition Video verwendet wird. Weitere Informationen finden Sie unter Arbeiten mit Streaming-Videoereignissen.

KinesisVideo

Typ: KinesisVideo Objekt

KinesisVideo

Informationen über den Kinesis-Videostrom, der das Quellvideo in Amazon Rekognition Video streamt. Weitere Informationen finden Sie unter Arbeiten mit Streaming-Videoereignissen.

StreamArn

Der Amazon-Ressourcenname (ARN) des Kinesis-Videostroms.

Typ: Zeichenfolge

FragmentNumber

Das Fragment des Streaming-Videos, das den Frame enthält, den dieser Datensatz darstellt.

Typ: Zeichenfolge

ProducerTimestamp

Der produzentenseitige Unix-Zeitstempel des Fragments. Weitere Informationen finden Sie unter PutMedia.

Typ: Zahl

ServerTimestamp

Der serverseitige Unix-Zeitstempel des Fragments. Weitere Informationen finden Sie unter PutMedia.

Typ: Zahl

FrameOffsetInSeconds

Der Versatz des Frames (in Sekunden) innerhalb des Fragments.

Typ: Zahl

StreamProcessorInformation

Statusinformationen über den Stromprozessor.

Status

Der aktuelle Status des Stromprozessors. Der einzig mögliche Wert ist RUNNING.

Typ: Zeichenfolge

FaceSearchResponse

Informationen über ein erkanntes Gesicht in einem Streaming-Video-Frame und die Gesichter in einer Eingabesammlung, die mit dem erkannten Gesicht übereinstimmen. Sie geben die Sammlung in einem Aufruf von CreateStreamProcessor an. Weitere Informationen finden Sie unter Arbeiten mit Streaming-Videoereignissen.

DetectedFace

Gesichtsdetails zu einem in einem analysierten Video-Frame erkannten Gesicht.

Typ: DetectedFace Objekt

MatchedFaces

Ein Array von Gesichtsdetails für Gesichter in einer Sammlung, die mit dem in `DetectedFace` erkannten Gesicht übereinstimmen.

Typ: MatchedFace Objekt-Array

DetectedFace

Informationen über ein Gesicht, das in einem Streaming-Video-Frame erkannt wurde. Übereinstimmende Gesichter in der Eingabesammlung stehen im MatchedFace-Objektfeld zur Verfügung.

BoundingBox

Der Begrenzungsrahmen liefert die Koordinaten für ein Gesicht, das in einem analysierten Videobild erkannt wird. Das BoundingBox Objekt hat dieselben Eigenschaften wie das BoundingBox Objekt, das für die Bildanalyse verwendet wird.

Typ: BoundingBox Objekt

Wahrscheinlichkeit

Das Zuverlässigkeitsniveau (1–100) von Amazon Rekognition Video, dass es sich bei dem erkannten Gesicht tatsächlich um ein Gesicht handelt. 1 ist das niedrigste Zuverlässigkeitsniveau, 100 das höchste.

Typ: Zahl

Erkennungszeichen

Ein Array von Gesichtsmerkmalen.

Typ: Merkmal-Objekt-Array

Pose

Gibt die Pose des Gesichts an, die durch Nicken, Neigen und Drehen bestimmt ist.

Typ: Pose-Objekt

Qualität

Gibt die Helligkeit und Schärfe des Gesichtsbilds an.

Typ: ImageQuality Objekt

MatchedFace

Informationen über ein Gesicht, das mit einem in einem analysierten Video-Frame erkannten Gesicht übereinstimmt.

Gesicht

Informationen zur Gesichtsübereinstimmung für ein Gesicht in der Eingabesammlung, das mit dem Gesicht übereinstimmt, das im DetectedFace-Objekt erkannt wurde.

Typ: Gesichts-Objekt

Ähnlichkeit

Das Zuverlässigkeitsniveau (1–100), dem die Gesichter entsprechen. 1 ist das niedrigste Zuverlässigkeitsniveau, 100 das höchste.

Typ: Zahl

Streaming mit einem GStreamer Plugin

Amazon Rekognition Video kann ein Live-Streaming-Video von einer Gerätekamera analysieren. Um auf Medieneingaben von einer Gerätequelle zuzugreifen, müssen Sie installieren GStreamer. GStreamer ist eine Multimedia-Framework-Software eines Drittanbieters, die Medienquellen und Verarbeitungstools in Workflow-Pipelines miteinander verbindet. Sie müssen auch das Amazon Kinesis Video Streams Producer Plug-In für GStreamer installieren. Diese Operation setzt voraus, dass Sie Ihre Amazon-Rekognition-Video- und Amazon-Kinesis-Ressourcen erfolgreich eingerichtet haben. Weitere Informationen finden Sie unter Einrichtung Ihrer Amazon-Rekognition-Video- und Amazon-Kinesis-Ressourcen.

Schritt 1: Installieren von GStreamer

Laden Sie GStreamer, eine Multimedia-Plattform-Software eines Drittanbieters, herunter und installieren Sie es. Sie können eine Paketverwaltungssoftware wie Homebrew (GStreamer on Homebrew) verwenden oder sie direkt von der Freedesktop-Website herunterladen.

Überprüfen Sie die erfolgreiche Installation von GStreamer, indem Sie von Ihrem Befehlszeilenterminal aus einen Video-Feed mit einer Testquelle starten.

```
$ gst-launch-1.0 videotestsrc ! autovideosink
```

Schritt 2: Installieren Sie das Kinesis Video Streams Producer Plug-In

In diesem Abschnitt laden Sie die Amazon Kinesis Video Streams Producer Bibliothek herunter und installieren das Kinesis Video Streams GStreamer Plug-In.

Erstellen Sie ein Verzeichnis und klonen Sie den Quellcode aus dem GitHub-Repository. Stellen Sie sicher, dass Sie den `--recursive`-Parameter angeben.

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

Folgen Sie den Anweisungen der Bibliothek, um das Projekt zu konfigurieren und zu erstellen. Stellen Sie sicher, dass Sie die plattformspezifischen Befehle für Ihr Betriebssystem verwenden. Verwenden Sie den `-DBUILD_GSTREAMER_PLUGIN=ON`-Parameter, wenn Sie `cmake` ausführen, um das Kinesis Video Streams GStreamer Plug-In zu installieren. Für dieses Projekt sind die folgenden zusätzlichen Pakete erforderlich, die in der Installation enthalten sind: GCC oder Clang, Curl, Openssl und Log4CPlus. Wenn Ihr Build aufgrund eines fehlenden Pakets fehlschlägt, überprüfen Sie, ob das Paket installiert ist und sich in Ihrem PATH befindet. Wenn Sie beim Erstellen auf den Fehler „C-kompiliertes Programm kann nicht ausgeführt werden“ stoßen, führen Sie den Build-Befehl erneut aus. Manchmal wird der richtige C-Compiler nicht gefunden.

Überprüfen Sie die Installation des Kinesis Video Streams Plug-Ins, indem Sie den folgenden Befehl ausführen.

```
$ gst-inspect-1.0 kvssink
```

Die folgenden Informationen, wie z. B. die Fabrik- und Plug-In-Details, sollten erscheinen:

```
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/

  ...
```

Schritt 3: GStreamer mit dem Kinesis Video Streams Plug-In ausführen

Bevor Sie mit dem Streaming von einer Gerätekamera zu Kinesis Video Streams beginnen, müssen Sie möglicherweise die Medienquelle in einen akzeptablen Codec für Kinesis Video Streams konvertieren. Um die Spezifikationen und Formatfunktionen der Geräte zu ermitteln, die derzeit an Ihr Gerät angeschlossen sind, führen Sie den folgenden Befehl aus.

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

Um mit dem Streaming zu beginnen, starten Sie GStreamer mit dem folgenden Beispielbefehl und fügen Sie Ihre Anmeldeinformationen und Amazon-Kinesis-Video-Streams-Informationen hinzu. Sie sollten die Zugriffsschlüssel und die Region für die IAM-Servicerolle verwenden, die Sie erstellt haben, während Sie Amazon Rekognition Zugriff auf Ihre Kinesis-Streams gewähren. Weitere Information über IAM-Zugriffsschlüssel finden Sie unter Verwalten von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch. Außerdem können Sie die Parameter für das Videoformat so anpassen, wie es Ihre Nutzung erfordert und auf Ihrem Gerät verfügbar ist.

```
$ 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"
```

Weitere Startbefehle finden Sie unter Beispiel für GStreamer Startbefehle.

> **Note**
>
> Wenn Ihr Startbefehl mit einem Fehler endet, bei dem es sich nicht um eine Verhandlung handelt, überprüfen Sie die Ausgabe des Gerätemonitors und stellen Sie sicher, dass es sich bei den `videoconvert`-Parameterwerten um gültige Funktionen Ihres Geräts handelt.

Nach einigen Sekunden wird in Ihrem Kinesis-Videostrom ein Video-Feed von Ihrer Gerätekamera angezeigt. Um mit der Erkennung und Zuordnung von Gesichtern mit Amazon Rekognition zu beginnen, starten Sie Ihren Amazon-Rekognition-Video-Stromprozessor. Weitere Informationen finden Sie unter Überblick über den Betrieb des Amazon-Rekognition-Video-Stromprozessors.

Fehlerbehebung beim Streamen von Videos

Dieses Thema enthält Informationen zur Fehlerbehebung beim Streamen von Videos mit Amazon Rekognition Video.

Themen

- Ich weiß nicht, ob mein Stromprozessor erfolgreich erstellt wurde
- Ich weiß nicht, ob mein Stromprozessor korrekt konfiguriert ist
- Mein Stromprozessor gibt keine Ergebnisse zurück
- Der Status meines Stromprozessors lautet FAILED
- Mein Stromprozessor gibt nicht die erwarteten Ergebnisse zurück

Ich weiß nicht, ob mein Stromprozessor erfolgreich erstellt wurde

Verwenden Sie den folgenden AWS CLI Befehl, um eine Liste der Stream-Prozessoren und deren aktuellen Status abzurufen.

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

Mit dem folgenden AWS CLI Befehl können Sie weitere Details abrufen. Ersetzen Sie `stream-processor-name` durch den Namen des erforderlichen Stromprozessors.

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

Ich weiß nicht, ob mein Stromprozessor korrekt konfiguriert ist

Wenn Ihr Code nicht die Analyseergebnisse von Amazon Rekognition Video ausgibt, ist Ihr Stromprozessor möglicherweise nicht korrekt konfiguriert. Führen Sie die folgenden Schritte aus, um zu bestätigen, dass Ihr Stromprozessor korrekt konfiguriert ist und Ergebnisse herstellen kann.

So stellen Sie fest, ob Ihre Lösung ordnungsgemäß konfiguriert ist

1. Führen Sie den folgenden Befehl aus, um zu bestätigen, dass Ihr Stromprozessor ausgeführt wird. Ändern Sie `stream-processor-name` in den Namen Ihres Stromprozessors. Der Stromprozessor wird ausgeführt, wenn `Status` den Wert `RUNNING` aufweist. Wenn der Status `RUNNING` lautet und Sie keine Ergebnisse erhalten, siehe Mein Stromprozessor gibt keine Ergebnisse zurück. Wenn der Status `FAILED` lautet, finden Sie unter Der Status meines Stromprozessors lautet FAILED Informationen dazu.

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

2. Wenn Ihr Stream-Prozessor läuft, führen Sie die folgende Bash oder den folgenden PowerShell Befehl aus, um Daten aus dem Kinesis-Ausgabedatenstream zu lesen.

   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. Mit dem Decodierungstool auf der Website „Base64 Decode“ können Sie die Ausgabe in eine für Menschen lesbare Zeichenfolge decodieren. Weitere Informationen finden Sie unter Schritt 3: Rufen Sie den Datensatz ab.

4. Wenn die Befehle funktionieren und im Kinesis-Datenstrom Ergebnisse der Gesichtserkennung angezeigt werden, ist Ihre Lösung ordnungsgemäß konfiguriert. Bei einem Befehlsfehler prüfen Sie die anderen Vorschläge zur Problembehandlung und sehen Sie sich die Informationen unter Gewähren von Zugriff auf Ihre Ressourcen für Amazon Rekognition Videoan.

Alternativ können Sie den AWS Lambda Blueprint "kinesis-process-record" verwenden, um Nachrichten aus dem Kinesis-Datenstrom CloudWatch zur kontinuierlichen Visualisierung zu protokollieren. Dadurch entstehen zusätzliche Kosten für und. AWS Lambda CloudWatch

Mein Stromprozessor gibt keine Ergebnisse zurück

Ihr Stromprozessor gibt möglicherweise aus verschiedenen Gründen keine Ergebnisse zurück.

Grund 1: Der Stromprozessor ist nicht korrekt konfiguriert

Ihr Stromprozessor ist möglicherweise nicht korrekt konfiguriert. Weitere Informationen finden Sie unter Ich weiß nicht, ob mein Stromprozessor korrekt konfiguriert ist.

Grund 2: Ihr Stromprozessor weist nicht den Status RUNNING auf

So beheben Sie Statusfehler des Stromprozessors

1. Überprüfen Sie den Status des Stream-Prozessors mit dem folgenden AWS CLI Befehl.

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

2. Wenn `Status` den Wert `STOPPED`aufweist, starten Sie Ihren Stromprozessor mit dem folgenden Befehl:

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

3. Wenn `Status` den Wert `FAILED` aufweist, finden Sie weitere Informationen unter Der Status meines Stromprozessors lautet FAILED.
4. Wenn `Status` den Wert `STARTING` aufweist, warten Sie 2 Minuten und überprüfen Sie den Status, indem Sie Schritt 1 wiederholen. Wenn der Status noch den Wert `STARTING` aufweist, gehen Sie wie folgt vor:

   a. Löschen Sie den Stromprozessor mit dem folgenden Befehl.

      ```
      aws rekognition delete-stream-processor --name stream-processor-name
      ```

   b. Erstellen Sie einen neuen Stromprozessor mit der gleichen Konfiguration. Weitere Informationen finden Sie unter Arbeiten mit Streaming-Videoereignissen.

   c. Wenn Sie weiterhin Probleme haben, wenden Sie sich an den AWS Support.

5. Wenn `Status` den Wert `RUNNING` aufweist, finden Sie weitere Informationen unter Grund 3: Es sind keine aktiven Daten im Kinesis-Videostrom vorhanden.

Grund 3: Es sind keine aktiven Daten im Kinesis-Videostrom vorhanden

Um zu überprüfen, ob der Kinesis-Videostrom aktive Daten enthält

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Kinesis Video Streams Streams-Konsole unter https://console.aws.amazon.com/kinesisvideo/.
2. Wählen Sie den Kinesis-Videostrom aus, der die Eingabe für den Amazon-Rekognition-Stromprozessor ist.
3. Wenn die Vorschau Keine Daten im Strom angibt, befinden sich keine Daten zur Verarbeitung durch Amazon Rekognition Video im Eingabestrom.

Informationen zur Videoproduktion mit Kinesis Video Streams finden Sie unter Kinesis Video Streams Producer Libraries.

Der Status meines Stromprozessors lautet FAILED

Sie können den Status eines Stream-Prozessors mit dem folgenden AWS CLI Befehl überprüfen.

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

Wenn der Status den Wert FAILED aufweist, überprüfen Sie die Informationen zur Fehlerbehebung für die folgenden Fehlermeldungen.

Fehler: „Der Rolle wird der Zugriff verweigert“

Die vom Stromprozessor verwendete IAM-Rolle ist nicht vorhanden oder Amazon Rekognition Video verfügt nicht über die Berechtigung, die Rolle anzunehmen.

So beheben Sie Probleme mit dem Zugriff auf die IAM-Rolle

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Klicken Sie im linken Navigationsbereich auf Rollen und bestätigen Sie, dass die Rolle vorhanden ist.
3. Wenn die Rolle vorhanden ist, überprüfen Sie, ob für die Rolle die AmazonRekognitionServiceRoleBerechtigungsrichtlinie gilt.
4. Wenn die Rolle nicht vorhanden ist oder nicht über die erforderlichen Berechtigungen verfügt, siehe Gewähren von Zugriff auf Ihre Ressourcen für Amazon Rekognition Video.

5. Starten Sie den Stream-Prozessor mit dem folgenden AWS CLI Befehl.

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

Fehler: „Kein Zugriff auf Kinesis Video oder Zugriff auf Kinesis-Daten verweigert“

Die Rolle hat keinen Zugriff auf die Kinesis-Video-Streams-API-Operationen `GetMedia` und `GetDataEndpoint`. Möglicherweise hat es auch keinen Zugriff auf die Kinesis-Data-Streams-API-Operationen `PutRecord` und `PutRecords`.

So beheben Sie Probleme mit API-Berechtigungen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Öffnen Sie die Rolle und stellen Sie sicher, dass die folgende Berechtigungsrichtlinie angefügt ist.

```
{
    "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. Wenn eine der Berechtigungen fehlt, aktualisieren Sie die Richtlinie. Weitere Informationen finden Sie unter Gewähren von Zugriff auf Ihre Ressourcen für Amazon Rekognition Video.

Fehler: „Stream ***input-video-stream-name*** existiert nicht“

Die Kinesis–Videostrom-Eingabedaten für den Stromprozessor sind nicht vorhanden oder nicht korrekt konfiguriert.

So beheben Sie Probleme mit dem Kinesis-Videostrom

1. Mit dem folgenden Befehl können Sie bestätigen, dass der Stream vorhanden ist.

   ```
   aws kinesisvideo list-streams
   ```

2. Wenn der Stream vorhanden ist, überprüfen Sie Folgendes.

   - Der Amazon-Ressourcenname (ARN) ist identisch mit dem ARN des Eingabestroms für den Stromprozessor.
   - Der Kinesis-Videostrom befindet sich in derselben Region wie der Stromprozessor.

   Wenn der Stream-Prozessor nicht korrekt konfiguriert ist, löschen Sie ihn mit dem folgenden AWS CLI Befehl.

   ```
   aws rekognition delete-stream-processor --name stream-processor-name
   ```

3. Erstellen Sie einen neuen Stromprozessor mit dem beabsichtigten Kinesis Video Stream. Weitere Informationen finden Sie unter Erstellen des Amazon-Rekognition-Video-Gesichtssuche-Stromprozessors.

Fehler: „Sammlung nicht gefunden“

Die vom Stromprozessor für die Gesichtsprüfung verwendete Amazon-Rekognition-Sammlung ist nicht vorhanden, oder die falsche Sammlung wird verwendet.

So bestätigen Sie die Sammlung

1. Verwenden Sie den folgenden AWS CLI Befehl, um festzustellen, ob die erforderliche Sammlung vorhanden ist. Wechseln Sie `region` zu der AWS Region, in der Sie Ihren Stream-Prozessor ausführen.

   ```
   aws rekognition list-collections --region region
   ```

Wenn die erforderliche Sammlung nicht vorhanden ist, erstellen Sie eine neue Sammlung und fügen Sie Gesichtsinformationen hinzu. Weitere Informationen finden Sie unter Gesichtssuche in einer Sammlung.

2. Überprüfen Sie in Ihrem CreateStreamProcessor-Aufruf, dass der Wert des `CollectionId`-Eingabeparameters korrekt ist.
3. Starten Sie den Stream-Prozessor mit dem folgenden AWS CLI Befehl.

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

Fehler: „Stream ***output-kinesis-data-stream-name*** unter Konto ***account-id*** nicht gefunden“

Der Kinesis-Ausgabedatenstream, der vom Stream-Prozessor verwendet wird, ist in Ihrer Region nicht vorhanden AWS-Konto oder befindet sich nicht in derselben AWS Region wie Ihr Stream-Prozessor.

So beheben Sie Probleme mit dem Kinesis-Datenstrom

1. Verwenden Sie den folgenden AWS CLI Befehl, um festzustellen, ob der Kinesis-Datenstrom vorhanden ist. Wechseln Sie `region` zu der AWS Region, in der Sie Ihren Stream-Prozessor verwenden.

```
aws kinesis list-streams --region region
```

2. Wenn der Kinesis-Datenstrom vorhanden ist, überprüfen Sie, ob der Name des Kinesis-Datenstroms dem des Ausgabestroms entspricht, der vom Stromprozessor verwendet wird.
3. Wenn der Kinesis-Datenstream nicht existiert, ist er möglicherweise in einer anderen AWS Region vorhanden. Der Kinesis-Datenstrom muss sich in derselben Region wie der Stromprozessor befinden.
4. Erstellen Sie bei Bedarf einen neuen Kinesis-Datenstrom.

   a. Erstellen Sie einen Kinesis-Datenstrom mit demselben Namen wie dem des Stromprozessors. Weitere Informationen finden Sie unter Schritt 1: Erstellen eines Datenstroms.

   b. Starten Sie den Stream-Prozessor mit dem folgenden AWS CLI Befehl.

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

Mein Stromprozessor gibt nicht die erwarteten Ergebnisse zurück

Wenn Ihr Stromprozessor nicht die erwarteten Gesichtstreffer zurückgibt, verwenden Sie die folgenden Informationen.

- Gesichtssuche in einer Sammlung
- Empfehlungen für die Kameraeinrichtung (Videostreaming)

Pfade von Personen

> **Note**
>
> Hinweis zum Ende des Supports: Am 31. Oktober 2025 wird AWS den Support für Amazon Rekognition People Pathing einstellen. Nach dem 31. Oktober 2025 können Sie die Funktion Rekognition People Pathing nicht mehr nutzen. Weitere Informationen finden Sie in diesem Blogbeitrag.

Amazon Rekognition Video kann den Pfad von Personen in Videos nachverfolgen und Informationen bereitstellen wie:

- Die Position von Personen im Videobild zu dem Zeitpunkt, an dem ihr Pfad nachverfolgt wird.
- Gesichtsmarken wie z. B. die Position des linken Auges, wenn sie entdeckt werden.

Der Amazon-Rekognition-Video-Personen-Pfad in gespeicherten Videos ist eine asynchrone Operation. Rufen StartPersonTrackingSie an, um die Pfadverfolgung von Personen in Videos zu starten. Das Amazon-Simple-Notification-Service-Thema, zu dem Amazon Rekognition Video die Ergebnisse der Objekterkennung und den Abschlussstatus einer Videoanalyse-Operation veröffentlicht. Wenn die Videoanalyse erfolgreich ist, rufen Sie an, GetPersonTrackingum die Ergebnisse der Videoanalyse zu erhalten. Weitere Informationen zum Aufrufen von Amazon-Rekognition-Video-API-Operationen finden Sie unter Amazon-Rekognition-Video-Operationen aufrufen.

Die folgende Prozedur zeigt, wie man den Pfad von Personen durch ein Video, das in einem Amazon-S3-Bucket gespeichert ist, verfolgen kann. Das Beispiel erweitert den Code in Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK), der eine Amazon-Simple-Queue-Service-Warteschlange verwendet, um den Abschlussstatus einer Videoanalyseanforderung zu erhalten.

So erkennen Sie Personen in einem Video, das in einem Amazon-S3-Bucket gespeichert ist (SDK)

1. Führen Sie Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) aus.
2. Fügen Sie den folgenden Code in der Klasse `VideoDetect` ein, die Sie in Schritt 1 erstellt haben.

Java

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

       //
Persons=============================================================================
       private static void StartPersonDetection(String bucket, String video)
 throws Exception{


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

        StartPersonTrackingRequest req = new StartPersonTrackingRequest()
                .withVideo(new Video()
                        .withS3Object(new S3Object()
                            .withBucket(bucket)
                            .withName(video)))
                .withNotificationChannel(channel);



        StartPersonTrackingResult startPersonDetectionResult =
 rek.startPersonTracking(req);
        startJobId=startPersonDetectionResult.getJobId();

       }

       private static void GetPersonDetectionResults() throws Exception{
           int maxResults=10;
           String paginationToken=null;
           GetPersonTrackingResult personTrackingResult=null;

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

```
                personTrackingResult = rek.getPersonTracking(new
GetPersonTrackingRequest()
                     .withJobId(startJobId)
                     .withNextToken(paginationToken)
                     .withSortBy(PersonTrackingSortBy.TIMESTAMP)
                     .withMaxResults(maxResults));

                VideoMetadata
videoMetaData=personTrackingResult.getVideoMetadata();

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


                //Show persons, confidence and detection times
                List<PersonDetection> detectedPersons=
personTrackingResult.getPersons();

                for (PersonDetection detectedPerson: detectedPersons) {

                   long seconds=detectedPerson.getTimestamp()/1000;
                   System.out.print("Sec: " + Long.toString(seconds) + " ");
                   System.out.println("Person Identifier: "  +
detectedPerson.getPerson().getIndex());
                        System.out.println();
                }
           }  while (personTrackingResult !=null &&
personTrackingResult.getNextToken() != null);

       }
```

Ersetzen Sie in der Funktion `main` die folgenden Zeilen:

```
       StartLabelDetection(amzn-s3-demo-bucket, video);

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

mit:

```
        StartPersonDetection(amzn-s3-demo-bucket, video);

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

Java V2

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

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

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

```
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

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

        startPersonLabels(rekClient, channel, bucket, video);
        getPersonDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startPersonLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
```

```
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

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

            StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vidOb)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

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

    public static void getPersonDetectionResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetPersonTrackingResponse personTrackingResult = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (personTrackingResult != null)
                    paginationToken = personTrackingResult.nextToken();

                GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();
```

```
                // Wait until the job succeeds
                while (!finished) {

                    personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                    status = personTrackingResult.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
personTrackingResult.videoMetadata();

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

                List<PersonDetection> detectedPersons =
personTrackingResult.persons();
                for (PersonDetection detectedPerson : detectedPersons) {
                    long seconds = detectedPerson.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    System.out.println("Person Identifier: " +
detectedPerson.person().index());
                    System.out.println();
                }

            } while (personTrackingResult != null &&
personTrackingResult.nextToken() != null);
```

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

Python

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

    # ============== People pathing ===============
    def StartPersonPathing(self):
        response=self.rek.start_person_tracking(Video={'S3Object': {'Bucket':
 self.bucket, 'Name': self.video}},
            NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
 self.snsTopicArn})

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

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

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

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

            for personDetection in response['Persons']:
                print('Index: ' + str(personDetection['Person']['Index']))
                print('Timestamp: ' + str(personDetection['Timestamp']))
```

```
                print()

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

Ersetzen Sie in der Funktion `main` die folgenden Zeilen:

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

mit:

```
    analyzer.StartPersonPathing()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetPersonPathingResults()
```

CLI

Führen Sie den folgenden AWS CLI Befehl aus, um zu starten, dass Personen in einem Video den Pfad eingeben.

```
aws rekognition start-person-tracking --video "{"S3Object":{"Bucket":"amzn-s3-
demo-bucket","Name":"video-name"}}" \
--notification-channel "{"SNSTopicArn":"topic-ARN","RoleArn":"role-ARN"}" \
--region region-name --profile profile-name
```

Aktualisieren Sie die folgenden Werte:

- Ändern Sie `amzn-s3-demo-bucket` und `video-name` in den Amazon-S3-Bucket-Namen und den Dateinamen, die Sie in Schritt 2 angegeben haben.
- Ändern Sie `region-name` in die von Ihnen verwendete AWS-Region.
- Ersetzen Sie den Wert von `profile-name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.
- Ändern Sie `topic-ARN` in den ARN des Amazon-SNS-Themas, das Sie in Schritt 3 von Amazon Rekognition Video konfigurieren erstellt haben.

- Ändern Sie `role-ARN` in den ARN der IAM-Servicerolle, die Sie in Schritt 7 von Amazon Rekognition Video konfigurieren erstellt haben.

Wenn Sie auf einem Windows-Gerät auf die CLI zugreifen, verwenden Sie doppelte Anführungszeichen anstelle von einfachen Anführungszeichen und maskieren Sie die inneren doppelten Anführungszeichen durch einen Backslash (d. h. \), um eventuell auftretende Parserfehler zu beheben. Ein Beispiel finden Sie unten:

```
aws rekognition start-person-tracking --video "{\"S3Object\":{\"Bucket\":\"amzn-
s3-demo-bucket\",\"Name\":\"video-name\"}}"
--notification-channel "{\"SNSTopicArn\":\"topic-ARN\",\"RoleArn\":\"role-ARN
\"}" \
--region region-name --profile profile-name
```

Nachdem Sie das vorangegangene Codebeispiel ausgeführt haben, kopieren Sie die zurückgegebene `jobID` und geben Sie sie an den folgenden `GetPersonTracking`-Befehl weiter, um Ihre Ergebnisse zu erhalten, und ersetzen Sie `job-id-number` durch `jobID`, die Sie zuvor erhalten haben:

```
aws rekognition get-person-tracking --job-id job-id-number
```

> **Note**
>
> Wenn Sie zusätzlich zu Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) bereits ein anderes Videobeispiel ausgeführt haben, ist der zu ersetzende Code möglicherweise anders.

3. Führen Sie den Code aus. Die eindeutigen Kennungen für getrackte Personen werden zusammen mit der Zeit angezeigt, in Sekunden, in denen der Pfad der Personen getrackt wurde.

GetPersonTracking Reaktion auf den Vorgang

`GetPersonTracking` gibt ein Array, `Persons`, von PersonDetection-Objekten zurück, die Details über die im Video erkannten Personen und den Zeitpunkt der Verfolgung der Pfade enthalten.

Sie können `Persons` mithilfe der `SortBy`-Eingabeparameter sortieren. Legen Sie `TIMESTAMP` fest, um die Elemente nach dem Zeitpunkt zu sortieren, zu dem die Pfade der Personen im Video nachverfolgt werden. Geben Sie `INDEX` an, um nach den Personen zu sortieren, die im Video getrackt werden. Innerhalb jeder Gruppe von Ergebnissen für eine Person werden die Elemente nach absteigender Zuverlässigkeit der Pfaderkennung sortiert. Standardmäßig wird das zurückgegebene Array `Persons` nach `TIMESTAMP` sortiert. Nachfolgend finden Sie ein Beispiel einer JSON-Antwort von `GetPersonDetection`. Nach der Zeit sortierte Ergebnisse erfolgen in Millisekunden ab dem Beginn des Videos, in dem die Pfade der Personen aufgespürt werden. In der Antwort ist Folgendes zu beachten:

- Personeninformationen – Das `PersonDetection`-Arrayelement enthält Informationen über die entdeckte Person. Zum Beispiel die Zeit, zu der die Person erkannt wurde (`Timestamp`), die Position der Person im Videobild zum Zeitpunkt der Erkennung (`BoundingBox`) und wie zuversichtlich Amazon Rekognition Video ist, dass die Person korrekt erkannt wurde (`Confidence`).

  Gesichtsmerkmale werden nicht bei jedem Zeitstempel zurückgegeben, wenn der Pfad der Person getrackt wird. Außerdem kann es unter Umständen vorkommen, dass der Körper einer getrackten Person nicht sichtbar ist, so dass nur ihre Gesichtsposition zurückgegeben wird.
- Seiteninformationen – Das Beispiel zeigt eine Seite mit Informationen der Personenerkennung. Sie können festlegen, wie viele Personenelemente zurückgegeben werden sollen, durch den Eingabeparameter `MaxResults` von `GetPersonTracking`. Wenn mehr Ergebnisse als `MaxResults` vorhanden sind, gibt `GetPersonTracking` einen Token zurück (`NextToken`), der dazu verwendet wird, die nächste Seite mit Ergebnissen zu erhalten. Weitere Informationen finden Sie unter Analyseergebnisse von Amazon Rekognition Video abrufen.
- Index – Eine eindeutige Kennung für die Erkennung der Person im gesamten Video.
- Video-Informationen – Die Antwort enthält Informationen über das Videoformat (`VideoMetadata`) in jeder Seite mit Informationen, die von `GetPersonDetection` zurückgegeben werden.

```
{
    "JobStatus": "SUCCEEDED",
```

```
    "NextToken": "AcDymG0fSSoaI6+BBYpka5wVlqttysSPP8VvWcujMDluj1QpFo/vf
+mrMoqBGk8eUEiFlllR6g==",
    "Persons": [
        {
            "Person": {
                "BoundingBox": {
                    "Height": 0.8787037134170532,
                    "Left": 0.00572916679084301,
                    "Top": 0.12129629403352737,
                    "Width": 0.21666666865348816
                },
                "Face": {
                    "BoundingBox": {
                        "Height": 0.20000000298023224,
                        "Left": 0.029999999329447746,
                        "Top": 0.2199999988079071,
                        "Width": 0.11249999701976776
                    },
                    "Confidence": 99.85971069335938,
                    "Landmarks": [
                        {
                            "Type": "eyeLeft",
                            "X": 0.06842322647571564,
                            "Y": 0.3010137975215912
                        },
                        {
                            "Type": "eyeRight",
                            "X": 0.10543643683195114,
                            "Y": 0.29697132110595703
                        },
                        {
                            "Type": "nose",
                            "X": 0.09569807350635529,
                            "Y": 0.33701086044311523
                        },
                        {
                            "Type": "mouthLeft",
                            "X": 0.0732642263174057,
                            "Y": 0.3757539987564087
                        },
                        {
                            "Type": "mouthRight",
                            "X": 0.10589495301246643,
                            "Y": 0.3722417950630188
```

```
                    }
                ],
                "Pose": {
                    "Pitch": -0.5589138865470886,
                    "Roll": -5.1093974113464355,
                    "Yaw": 18.69594955444336
                },
                "Quality": {
                    "Brightness": 43.052337646484375,
                    "Sharpness": 99.68138885498047
                }
            },
            "Index": 0
        },
        "Timestamp": 0
    },
    {
        "Person": {
            "BoundingBox": {
                "Height": 0.9074074029922485,
                "Left": 0.24791666865348816,
                "Top": 0.09259258955717087,
                "Width": 0.375
            },
            "Face": {
                "BoundingBox": {
                    "Height": 0.23000000417232513,
                    "Left": 0.42500001192092896,
                    "Top": 0.16333332657814026,
                    "Width": 0.12937499582767487
                },
                "Confidence": 99.97504425048828,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.46415066719055176,
                        "Y": 0.2572723925113678
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.5068183541297913,
                        "Y": 0.23705792427062988
                    },
                    {
```

```
                            "Type": "nose",
                            "X": 0.49765899777412415,
                            "Y": 0.28383663296699524
                        },
                        {
                            "Type": "mouthLeft",
                            "X": 0.487221896648407,
                            "Y": 0.3452930748462677
                        },
                        {
                            "Type": "mouthRight",
                            "X": 0.5142884850502014,
                            "Y": 0.33167609572410583
                        }
                    ],
                    "Pose": {
                        "Pitch": 15.966927528381348,
                        "Roll": -15.547388076782227,
                        "Yaw": 11.34195613861084
                    },
                    "Quality": {
                        "Brightness": 44.80223083496094,
                        "Sharpness": 99.95819854736328
                    }
                },
                "Index": 1
            },
            "Timestamp": 0
        }.....

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

Erkennung von persönlicher Schutzausrüstung

Amazon Rekognition kann persönliche Schutzausrüstung (PSA) erkennen, die von Personen auf einem Bild getragen wird. Sie können diese Informationen verwenden, um die Sicherheitsmaßnahmen am Arbeitsplatz zu verbessern. Mithilfe der PSA-Erkennung können Sie beispielsweise feststellen, ob Arbeiter auf einer Baustelle Kopfbedeckungen tragen oder ob medizinisches Personal Gesichtsbedeckungen und Handbedeckungen trägt. Die folgende Abbildung zeigt einige Arten von persönlicher Schutzausrüstung, die erkannt werden können.

Um PPE in einem Bild zu erkennen, rufen Sie die DetectProtectiveEquipmentAPI auf und übergeben ein Eingabebild. Die Antwort ist eine JSON-Struktur, die Folgendes umfasst.

- Die auf dem Bild erkannten Personen.
- Die Körperteile, an denen persönliche Schutzausrüstung getragen wird (Gesicht, Kopf, linke Hand und rechte Hand).

- Die Arten von PSA, die an Körperteilen festgestellt wurden (Gesichtsbedeckung, Handbedeckung und Kopfbedeckung).
- Bei erkannten PSA-Artikeln ein Indikator dafür, ob die PSA den entsprechenden Körperteil bedeckt oder nicht.

Sie erhalten Begrenzungsrahmen für die Positionen von Personen und PSA-Artikel, die auf dem Bild erkannt wurden.

Optional können Sie eine Zusammenfassung der auf einem Bild erkannten PSA-Artikel und Personen anfordern. Weitere Informationen finden Sie unter Zusammenfassung der PSA, die in einem Bild erkannt wurde.

> **Note**
>
> Die Amazon-Rekognition-PSA-Erkennung führt weder Gesichtserkennung noch Gesichtsvergleiche durch und kann die erkannten Personen nicht identifizieren.

Arten von PSA

DetectProtectiveEquipmenterkennt die folgenden Arten von persönlicher Schutzausrüstung. Wenn Sie andere Arten von PSA in Bildern erkennen möchten, sollten Sie erwägen, Amazon Rekognition Custom Labels zu verwenden, um ein benutzerdefiniertes Modell zu trainieren. Weitere Informationen finden Sie unter Amazon Rekognition Custom Labels.

Gesichtsbedeckung

`DetectProtectiveEquipment` kann gängige Gesichtsbedeckungen wie chirurgische Gesichtsbedeckungen, N95-Masken und Masken aus Stoff erkennen.

Handbedeckung

`DetectProtectiveEquipment` kann Handbedeckungen wie Operationshandschuhe und Schutzhandschuhe erkennen.

Kopfbedeckung

`DetectProtectiveEquipment` kann Schutzhelme erkennen.

Die API gibt an, dass in einem Bild ein Kopf, eine Hand oder eine Gesichtsbedeckung erkannt wurde. Die API gibt keine Informationen zum Typ einer bestimmten Abdeckung zurück. Zum Beispiel „Operationshandschuh“ für die Art einer Handbedeckung.

Zuverlässigkeit der PSA-Erkennung

Amazon Rekognition macht eine Vorhersage über das Vorhandensein von PSA, Personen und Körperteilen in einem Bild. Die API liefert eine Punktzahl (50–100), die angibt, wie sicher Amazon Rekognition von der Genauigkeit einer Vorhersage ist.

> **Note**
>
> Wenn Sie die `DetectProtectiveEquipment`-Operation nutzen möchten, um eine Entscheidung zu treffen, die sich auf die Rechte, den Datenschutz oder den Zugang zu Diensten einer Person auswirkt, empfehlen wir Ihnen, das Ergebnis zur Überprüfung und Bestätigung an einen Mitarbeiter weiterzuleiten, bevor Sie Maßnahmen ergreifen.

Zusammenfassung der PSA, die in einem Bild erkannt wurde

Optional können Sie eine Zusammenfassung der auf einem Bild erkannten PSA-Artikel und Personen anfordern. Sie können eine Liste der erforderlichen Schutzausrüstung (Gesichtsbedeckung, Handbedeckung oder Kopfbedeckung) und eine Mindestvertrauensschwelle (z. B. 80 %) angeben. Die Antwort umfasst eine konsolidierte Zusammenfassung pro Bild anhand der Identifikationsnummer (ID) der Personen mit der erforderlichen persönlichen Schutzausrüstung, der Personen ohne die erforderliche persönliche Schutzausrüstung und der Personen, bei denen keine Entscheidung getroffen werden konnte.

Anhand der Zusammenfassung können Sie schnell Fragen wie „Wie viele Personen tragen keine Gesichtsbedeckungen?“ oder Tragen alle persönliche Schutzausrüstung? beantworten. Jede erkannte Person in der Zusammenfassung hat eine eindeutige ID. Mithilfe der ID können Sie Informationen wie die Position des Begrenzungsrahmens einer Person ermitteln, die keine persönliche Schutzausrüstung trägt.

> **Note**
>
> Die ID wird für jede Bildanalyse nach dem Zufallsprinzip generiert und ist bei Bildern oder mehreren Analysen desselben Bilds nicht einheitlich.

Sie können Gesichtsbedeckungen, Kopfbedeckungen, Handbedeckungen oder eine Kombination Ihrer Wahl zusammenfassen. Informationen zur Angabe der erforderlichen PSA-Typen finden Sie unter Spezifizierung der Anforderungen für die Zusammenfassung. Sie können auch ein Mindestzuverlässigkeitsniveau (50–100) angeben, das erfüllt sein muss, damit Erkennungen in die Zusammenfassung aufgenommen werden.

Weitere Informationen zur Antwort auf die Zusammenfassung von `DetectProtectiveEquipment` finden Sie unter Die DetectProtectiveEquipment Antwort verstehen.

Tutorial: Eine AWS Lambda Funktion erstellen, die Bilder mit PPE erkennt

Sie können eine AWS Lambda Funktion erstellen, die persönliche Schutzausrüstung (PSA) in Bildern erkennt, die sich in einem Amazon S3 S3-Bucket befinden. Dieses Java V2-Tutorial finden Sie im AWS GitHub Dokumentations-SDK-Beispiel-Repository.

Grundlegendes zur API zur Erkennung persönlicher Schutzausrüstung

Die folgenden Informationen beschreiben die DetectProtectiveEquipmentAPI. Beispielcode finden Sie unter Erkennung von persönlicher Schutzausrüstung in einem Bild.

Bereitstellung eines Bilds

Sie können das Eingabebild (JPG- oder PNG-Format) entweder als Bild-Bytes bereitstellen oder auf ein in einem Amazon-S3-Bucket gespeichertes Bild verweisen.

Wir empfehlen, Bilder zu verwenden, bei denen das Gesicht der Person zur Kamera zeigt.

Wenn Ihr Eingabebild nicht auf 0 Grad gedreht ist, empfehlen wir, es vor dem Senden an `DetectProtectiveEquipment` auf 0 Grad zu drehen. Bilder im JPG-Format können

Ausrichtungsinformationen in den Metadaten des Exchangeable Image File Format (Exif) enthalten. Sie können diese Informationen verwenden, um einen Code zu schreiben, der Ihr Bild dreht. Weitere Informationen finden Sie unter Exif Version 2.32. Bilder im PNG-Format enthalten keine Informationen zur Bildausrichtung.

Um ein Bild aus einem Amazon S3 S3-Bucket zu übergeben, verwenden Sie einen Benutzer mit mindestens AmazonS3-RechtenReadOnlyAccess . Verwenden Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen zum Aufruf von `DetectProtectiveEquipment.`

Im folgenden Eingabe-JSON-Beispiel wird das Bild in einem Amazon-S3-Bucket übergeben. Weitere Informationen finden Sie unter Arbeiten mit Bildern. Im Beispiel wird eine Zusammenfassung aller PSA-Typen (Kopfbedeckung, Handbedeckung und Gesichtsbedeckung) mit einer Erkennungssicherheit (`MinConfidence`) von mindestens 80 % angefordert. Sie sollten einen `MinConfidence`-Wert zwischen 50 und 100 % angeben, da Vorhersagen nur dann von `DetectProtectiveEquipment` zurückgegeben werden, wenn die Erkennungssicherheit zwischen 50 und 100 % liegt. Wenn Sie einen Wert unter 50 % angeben, sind die Ergebnisse identisch, wenn Sie einen Wert von 50 % angeben. Weitere Informationen finden Sie unter Spezifizierung der Anforderungen für die Zusammenfassung.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "worker.jpg"
        }
    },
    "SummarizationAttributes": {
        "MinConfidence": 80,
        "RequiredEquipmentTypes": [
            "FACE_COVER",
            "HAND_COVER",
            "HEAD_COVER"
        ]
    }
}
```

Wenn Sie eine große Sammlung von Bildern verarbeiten, empfiehlt sich der Einsatz von AWS Batch zum Verarbeiten von Aufrufen an `DetectProtectiveEquipment` in Stapeln im Hintergrund.

Spezifizierung der Anforderungen für die Zusammenfassung

Sie können optional den Eingabeparameter `SummarizationAttributes` (ProtectiveEquipmentSummarizationAttributes) verwenden, um zusammenfassende Informationen zu den in einem Bild erkannten PSA-Typen anzufordern.

Verwenden Sie das `RequiredEquipmentTypes`-Array-Feld, um die PSA-Typen anzugeben, die zusammengefasst werden sollen. Fügen Sie in das Array einen oder mehrere `FACE_COVER` `HAND_COVER` oder `HEAD_COVER` ein.

Verwenden Sie das `MinConfidence`-Feld, um ein Mindestmaß an Erkennungssicherheit (50–100) anzugeben. Die Zusammenfassung enthält keine Personen, Körperteile, Körperteilabdeckung und PSA-Elemente, die mit einer Wahrscheinlichkeit von weniger als `MinConfidence` erkennt werden.

Weitere Informationen zur Antwort auf die Zusammenfassung von `DetectProtectiveEquipment` finden Sie unter Die DetectProtectiveEquipment Antwort verstehen.

Die DetectProtectiveEquipment Antwort verstehen

`DetectProtectiveEquipment` gibt ein Array von Personen zurück, die im Eingabebild erkannt wurden. Für jede Person werden Informationen über die erkannten Körperteile und die erkannten persönlichen Schutzausrüstungen zurückgegeben. Der JSON-Code für das folgende Bild eines Arbeiters, der eine Kopf-, Hand- und Gesichtsbedeckung trägt, lautet wie folgt.

Beachten Sie in den JSON-Daten Folgendes:

- Erkannte Personen: `Persons` ist eine Reihe von Personen, die auf dem Bild erkannt wurden (einschließlich Personen, die keine persönliche Schutzausrüstung tragen). `DetectProtectiveEquipment` kann persönliche Schutzausrüstung bei bis zu 15 Personen erkennen, die auf einem Bild erkannt wurden. Jedes ProtectiveEquipmentPersonObjekt in der Reihe enthält eine Personen-ID, einen Begrenzungsrahmen für die Person, erkannte Körperteile und erkannte persönliche Schutzausrüstung. Der Wert von `Confidence` in `ProtectiveEquipmentPerson` gibt die prozentuale Sicherheit von Amazon Rekognition an, dass das Begrenzungsfeld eine Person enthält.
- Körperteile — `BodyParts` ist eine Reihe von Körperteilen (ProtectiveEquipmentBodyPart), die an einer Person erkannt wurden (einschließlich Körperteile, die nicht durch persönliche Schutzausrüstung abgedeckt sind). Jeder `ProtectiveEquipmentBodyPart` enthält den Namen (`Name`) des erkannten Körperteils. `DetectProtectEquipment` kann Gesicht, Kopf, linke und

rechte Körperteile erkennen. Das `Confidence`-Feld in `ProtectiveEquipmentBodyPart` gibt das prozentuale Vertrauen an, das Amazon Rekognition in Bezug auf die Erkennungsgenauigkeit des Körperteils hat.

- PSA-Artikel: Das Array `EquipmentDetections` in einem `ProtectiveEquipmentBodyPart`-Objekt enthält eine Reihe von erkannten PSA-Artikeln. Jedes EquipmentDetectionObjekt enthält die folgenden Felder.
  - `Type` – der Typ der gefundenen PSA.
  - `BoundingBox` – ein Begrenzungsrahmen um die erkannte PSA.
  - `Confidence` – Das Vertrauen von Amazon Rekognition, dass der Begrenzungsrahmen die erkannte persönliche Schutzausrüstung enthält.
  - `CoversBodyPart` – Zeigt an, ob sich die erkannte persönliche Schutzausrüstung an der entsprechenden Körperstelle befindet.

  Das CoversBodyPartFeld `Value` ist ein boolescher Wert, der angibt, ob sich die erkannte PSA am entsprechenden Körperteil befindet. Das Feld `Confidence` gibt das Vertrauen in die Vorhersage an. Sie können `CoversBodyPart` verwenden, um Fälle herauszufiltern, in denen sich die erkannte persönliche Schutzausrüstung zwar auf dem Bild, aber nicht auf der Person befindet.

  > **Note**
  >
  > `CoversBodyPart` bedeutet bzw. impliziert nicht, dass die Person durch die Schutzausrüstung angemessen geschützt ist oder dass die Schutzausrüstung selbst ordnungsgemäß getragen wurde.

- Übersichtsinformationen: `Summary` enthält die im `SummarizationAttributes`-Eingabeparameter angegebenen zusammenfassenden Informationen. Weitere Informationen finden Sie unter Spezifizierung der Anforderungen für die Zusammenfassung.

  `Summary`ist ein Objekt des Typs ProtectiveEquipmentSummary, das die folgenden Informationen enthält.

  - `PersonsWithRequiredEquipment`— Eine Gruppe IDs von Personen, wobei jede Person die folgenden Kriterien erfüllt.
    - Die Person trägt die gesamte im `SummarizationAttributes`-Eingabeparameter angegebene persönliche Schutzausrüstung.

  - Die `Confidence` für Person (`ProtectiveEquipmentPerson`), Körperteil (`ProtectiveEquipmentBodyPart`) und PSA (`EquipmentDetection`) ist gleich oder größer als die angegebene minimale Vertrauensschwelle (`MinConfidence`).
  - Der Wert von `CoversBodyPart` für alle PSA-Artikel ist wahr.
- `PersonsWithoutRequiredEquipment`— Eine Reihe IDs von Personen, die eines der folgenden Kriterien erfüllen.
  - Der `Confidence`-Wert für Person (`ProtectiveEquipmentPerson`), Körperteil (`ProtectiveEquipmentBodyPart`) und Körperteilbedeckung (`CoversBodyPart`) liegt über dem angegebenen Mindestzuverlässigkeitsschwellenwert (`MinConfidence`), aber der Person fehlt ein oder mehrere angegebene PSA-Artikel (`SummarizationAttributes`).
  - Der Wert von `CoversBodyPart` ist falsch für jede angegebene persönliche Schutzausrüstung (`SummarizationAttributes`), deren `Confidence`-Wert über dem angegebenen Mindestzuverlässigkeitsschwellenwert (`MinConfidence`) liegt. Die Person verfügt außerdem über sämtliche angegebene PSA (`SummarizationAttributes`) und die `Confidence`-Werte für Person (`ProtectiveEquipmentPerson`), Körperteil (`ProtectiveEquipmentBodyPart`) und Schutzausrüstung (`EquipmentDetection`) sind größer oder gleich der minimalen Vertrauensschwelle (`MinConfidence`).
- `PersonsIndeterminate`— Eine Reihe von erkannten Personen, bei denen IDs der `Confidence` Wert für die Person (`ProtectiveEquipmentPerson`), den Körperteil (), die Schutzausrüstung (`ProtectiveEquipmentBodyPartEquipmentDetection`) oder den `CoversBodyPart` booleschen Wert unter dem angegebenen Mindestkonfidenzschwellenwert (`MinConfidence`) liegt.

Verwenden Sie die Array-Größe, um die Anzahl für eine bestimmte Zusammenfassung zu ermitteln. Die Größe von `PersonsWithRequiredEquipment` gibt beispielsweise Auskunft über die Anzahl der Personen, bei denen festgestellt wurde, dass sie den angegebenen PSA-Typ tragen.

Sie können die Personen-ID verwenden, um weitere Informationen über eine Person herauszufinden, z. B. den Standort der Person im Begrenzungsrahmen. Die Personen-ID ist dem ID-Feld eines Objekts (`ProtectiveEquipmentPerson`) zugeordnet, das in `Persons` (Array von `ProtectiveEquipmentPerson`) zurückgegeben wurde. Sie können dann den Begrenzungsrahmen und andere Informationen aus dem entsprechenden `ProtectiveEquipmentPerson`-Objekt abrufen.

```
{
   "ProtectiveEquipmentModelVersion": "1.0",
   "Persons": [
       {
           "BodyParts": [
               {
                   "Name": "FACE",
                   "Confidence": 99.99861145019531,
                   "EquipmentDetections": [
                       {
                           "BoundingBox": {
                               "Width": 0.14528800547122955,
                               "Height": 0.14956723153591156,
                               "Left": 0.4363413453102112,
                               "Top": 0.34203192591667175
                           },
                           "Confidence": 99.90001678466797,
                           "Type": "FACE_COVER",
                           "CoversBodyPart": {
                               "Confidence": 98.0676498413086,
                               "Value": true
                           }
                       }
                   ]
               },
               {
                   "Name": "LEFT_HAND",
                   "Confidence": 96.9786376953125,
                   "EquipmentDetections": [
                       {
                           "BoundingBox": {
                               "Width": 0.14495663344860077,
                               "Height": 0.12936046719551086,
                               "Left": 0.5114737153053284,
                               "Top": 0.5744519829750061
                           },
                           "Confidence": 83.72270965576172,
                           "Type": "HAND_COVER",
                           "CoversBodyPart": {
                               "Confidence": 96.9288558959961,
                               "Value": true
                           }
```

```
                    }
                ]
            },
            {
                "Name": "RIGHT_HAND",
                "Confidence": 99.82939147949219,
                "EquipmentDetections": [
                    {
                        "BoundingBox": {
                            "Width": 0.20971858501434326,
                            "Height": 0.20528452098369598,
                            "Left": 0.2711356580257416,
                            "Top": 0.6750612258911133
                        },
                        "Confidence": 95.70789337158203,
                        "Type": "HAND_COVER",
                        "CoversBodyPart": {
                            "Confidence": 99.85433197021484,
                            "Value": true
                        }
                    }
                ]
            },
            {
                "Name": "HEAD",
                "Confidence": 99.9999008178711,
                "EquipmentDetections": [
                    {
                        "BoundingBox": {
                            "Width": 0.24350935220718384,
                            "Height": 0.34623199701309204,
                            "Left": 0.43011072278022766,
                            "Top": 0.01103297434747219
                        },
                        "Confidence": 83.88762664794922,
                        "Type": "HEAD_COVER",
                        "CoversBodyPart": {
                            "Confidence": 99.96485900878906,
                            "Value": true
                        }
                    }
                ]
            }
        ],
```

```
            "BoundingBox": {
                "Width": 0.7403100728988647,
                "Height": 0.9412225484848022,
                "Left": 0.02214839495718479,
                "Top": 0.03134796395897865
            },
            "Confidence": 99.98855590820312,
            "Id": 0
        }
    ],
    "Summary": {
        "PersonsWithRequiredEquipment": [
            0
        ],
        "PersonsWithoutRequiredEquipment": [],
        "PersonsIndeterminate": []
    }
}
```

Erkennung von persönlicher Schutzausrüstung in einem Bild

Um persönliche Schutzausrüstung (PSA) von Personen auf einem Bild zu erkennen, verwenden Sie den API-Vorgang DetectProtectiveEquipmentohne Speicherung.

Sie können das Eingabebild als Bild-Byte-Array (base64-codierte Bild-Bytes) oder als Amazon-S3-Objekt bereitstellen, indem Sie entweder AWS-SDK oder AWS Command Line Interface (AWS CLI) verwenden. In diesen Beispielen wird ein Bild, das in einem Amazon-S3-Bucket gespeichert ist, verwendet. Weitere Informationen finden Sie unter Arbeiten mit Bildern.

So erkennen Sie PSA an Personen in einem Bild

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`- und `AmazonS3ReadOnlyAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Laden Sie ein Bild (das eine oder mehrere Personen mit PSA enthält) in Ihren S3-Bucket hoch.

Weitere Anleitungen finden Sie unter Upload eines Objekts in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service.

3. Verwenden Sie die folgenden Beispiele zum Aufrufen der `DetectProtectiveEquipment`-Operation. Informationen zur Anzeige von Begrenzungsrahmen in einem Bild finden Sie unter Anzeigen von Begrenzungsrahmen.

   Java

   In diesem Beispiel werden Informationen zu den PSA-Gegenständen angezeigt, die bei Personen erkannt wurden, die auf einem Bild erkannt wurden.

   Ändern Sie den Wert von `amzn-s3-demo-bucket` in den Namen des Amazon S3-Buckets, das Ihr Bild enthält. Ändern Sie den Wert von `photo` in Ihren Bilddateinamen.

```
//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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
//snippet-start:[rekognition.java2.detect_ppe.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
```

```
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import
 software.amazon.awssdk.services.rekognition.model.DetectProtectiveEquipmentRequest;
import
 software.amazon.awssdk.services.rekognition.model.DetectProtectiveEquipmentResponse;
import software.amazon.awssdk.services.rekognition.model.EquipmentDetection;
import
 software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentBodyPart;
import
 software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentSummarizationAttri
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
 software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentPerson;
import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.List;
//snippet-end:[rekognition.java2.detect_ppe.import]

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

 public static void main(String[] args) {

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

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

    String sourceImage = args[0];
    String bucketName = args[1];
    Region region = Region.US_WEST_2;
    S3Client s3 = S3Client.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
        .build();

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

    displayGear(s3, rekClient, sourceImage, bucketName) ;
    s3.close();
    rekClient.close();
    System.out.println("This example is done!");
}

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

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

    try {
        ProtectiveEquipmentSummarizationAttributes summarizationAttributes =
ProtectiveEquipmentSummarizationAttributes.builder()
            .minConfidence(80F)
            .requiredEquipmentTypesWithStrings("FACE_COVER", "HAND_COVER",
"HEAD_COVER")
            .build();

        SdkBytes sourceBytes = SdkBytes.fromInputStream(is);
```

```
        software.amazon.awssdk.services.rekognition.model.Image souImage =
Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectProtectiveEquipmentRequest request =
DetectProtectiveEquipmentRequest.builder()
            .image(souImage)
            .summarizationAttributes(summarizationAttributes)
            .build();

        DetectProtectiveEquipmentResponse result =
rekClient.detectProtectiveEquipment(request);
        List<ProtectiveEquipmentPerson> persons = result.persons();
        for (ProtectiveEquipmentPerson person: persons) {
            System.out.println("ID: " + person.id());
            List<ProtectiveEquipmentBodyPart> bodyParts=person.bodyParts();
            if (bodyParts.isEmpty()){
                System.out.println("\tNo body parts detected");
            } else
                for (ProtectiveEquipmentBodyPart bodyPart: bodyParts) {
                    System.out.println("\t" + bodyPart.name() + ". Confidence:
" + bodyPart.confidence().toString());
                    List<EquipmentDetection>
equipmentDetections=bodyPart.equipmentDetections();

                    if (equipmentDetections.isEmpty()){
                        System.out.println("\t\tNo PPE Detected on " +
bodyPart.name());
                    } else {
                        for (EquipmentDetection item: equipmentDetections) {
                            System.out.println("\t\tItem: " + item.type() + ".
Confidence: " + item.confidence().toString());
                            System.out.println("\t\tCovers body part: "
                                    + item.coversBodyPart().value().toString()
+ ". Confidence: " + item.coversBodyPart().confidence().toString());

                            System.out.println("\t\tBounding Box");
                            BoundingBox box =item.boundingBox();
                            System.out.println("\t\tLeft: "
+box.left().toString());
                            System.out.println("\t\tTop: " +
box.top().toString());
```

```
                            System.out.println("\t\tWidth: " +
box.width().toString());
                            System.out.println("\t\tHeight: " +
box.height().toString());
                            System.out.println("\t\tConfidence: " +
item.confidence().toString());
                            System.out.println();
                        }
                    }
                }
        }
        System.out.println("Person ID Summary\n-----------------");

        displaySummary("With required equipment",
result.summary().personsWithRequiredEquipment());
        displaySummary("Without required equipment",
result.summary().personsWithoutRequiredEquipment());
        displaySummary("Indeterminate",
result.summary().personsIndeterminate());

    } catch (RekognitionException e) {
        e.printStackTrace();
        System.exit(1);
    }
}

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

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

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

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

```
        return null;
    }

    static void displaySummary(String summaryType,List<Integer> idList) {
        System.out.print(summaryType + "\n\tIDs  ");
        if (idList.size()==0) {
            System.out.println("None");
        } else {
            int count=0;
            for (Integer id: idList ) {
                if (count++ == idList.size()-1) {
                    System.out.println(id.toString());
                } else {
                    System.out.print(id.toString() + ", ");
                }
            }
        }
        System.out.println();
    }
    // snippet-end:[rekognition.java2.detect_ppe.main]
}
```

AWS CLI

Dieser AWS CLI Befehl fordert eine PPE-Zusammenfassung an und zeigt die JSON-Ausgabe für den `detect-protective-equipment` CLI-Vorgang an.

Ändern Sie `amzn-s3-demo-bucket` in den Namen eines Amazon-S3-Buckets, der ein Bild enthält. Ändern Sie `input.jpg` in den Namen des Bilds, den Sie verwenden möchten.

```
aws rekognition detect-protective-equipment \
  --image "S3Object={Bucket=amzn-s3-demo-bucket,Name=input.jpg}" \
  --summarization-attributes
 "MinConfidence=80,RequiredEquipmentTypes=['FACE_COVER','HAND_COVER','HEAD_COVER']"
```

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `detect-protective-equipment` CLI-Vorgang an.

Ändern Sie `amzn-s3-demo-bucket` in den Namen eines Amazon-S3-Buckets, der ein Bild enthält. Ändern Sie `input.jpg` in den Namen des Bilds, den Sie verwenden möchten.

```
aws rekognition detect-protective-equipment \
```

```
  --image "S3Object={Bucket=amzn-s3-demo-bucket,Name=input.jpg}"
```

Python

In diesem Beispiel werden Informationen zu den PSA-Gegenständen angezeigt, die bei Personen erkannt wurden, die auf einem Bild erkannt wurden.

Ändern Sie den Wert von `amzn-s3-demo-bucket` in den Namen des Amazon S3-Buckets, das Ihr Bild enthält. Ändern Sie den Wert von `photo` in Ihren Bilddateinamen. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
# Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def detect_ppe(photo, bucket):

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

    response = client.detect_protective_equipment(Image={'S3Object': {'Bucket':
 bucket, 'Name': photo}},

 SummarizationAttributes={'MinConfidence': 80,

 'RequiredEquipmentTypes': ['FACE_COVER',

                     'HAND_COVER',

                     'HEAD_COVER']})

    print('Detected PPE for people in image ' + photo)
    print('\nDetected people\n---------------')
    for person in response['Persons']:

        print('Person ID: ' + str(person['Id']))
        print('Body Parts\n----------')
        body_parts = person['BodyParts']
        if len(body_parts) == 0:
```

```
            print('No body parts found')
        else:
            for body_part in body_parts:
                print('\t' + body_part['Name'] + '\n\t\tConfidence: ' +
 str(body_part['Confidence']))
                print('\n\t\tDetected PPE\n\t\t------------')
                ppe_items = body_part['EquipmentDetections']
                if len(ppe_items) == 0:
                    print('\t\tNo PPE detected on ' + body_part['Name'])
                else:
                    for ppe_item in ppe_items:
                        print('\t\t' + ppe_item['Type'] + '\n\t\t\tConfidence: '
 + str(ppe_item['Confidence']))
                        print('\t\tCovers body part: ' + str(
                            ppe_item['CoversBodyPart']['Value']) + '\n\t\t
\tConfidence: ' + str(
                            ppe_item['CoversBodyPart']['Confidence']))
                        print('\t\tBounding Box:')
                        print('\t\t\tTop: ' + str(ppe_item['BoundingBox']
['Top']))
                        print('\t\t\tLeft: ' + str(ppe_item['BoundingBox']
['Left']))
                        print('\t\t\tWidth: ' + str(ppe_item['BoundingBox']
['Width']))
                        print('\t\t\tHeight: ' + str(ppe_item['BoundingBox']
['Height']))
                        print('\t\t\tConfidence: ' +
 str(ppe_item['Confidence']))
            print()
        print()

    print('Person ID Summary\n----------------')
    display_summary('With required equipment', response['Summary']
['PersonsWithRequiredEquipment'])
    display_summary('Without required equipment', response['Summary']
['PersonsWithoutRequiredEquipment'])
    display_summary('Indeterminate', response['Summary']
['PersonsIndeterminate'])

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

# Display summary information for supplied summary.
def display_summary(summary_type, summary):
```

```
    print(summary_type + '\n\tIDs: ', end='')
    if (len(summary) == 0):
        print('None')
    else:
        for num, id in enumerate(summary, start=0):
            if num == len(summary) - 1:
                print(id)
            else:
                print(str(id) + ', ', end='')

def main():
    photo = 'photo-name'
    bucket = 'amzn-s3-demo-bucket'
    person_count = detect_ppe(photo, bucket)
    print("Persons detected: " + str(person_count))

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

Beispiel: Zeichnen von Begrenzungsrahmen um Gesichtsbedeckungen

Die folgenden Beispiele zeigen Ihnen, wie Sie Begrenzungsrahmen um Gesichtsbedeckungen zeichnen, die bei Personen entdeckt wurden. Ein Beispiel, das Amazon DynamoDB verwendet AWS Lambda , finden Sie im AWS Documentation SDK Examples GitHub Repository.

Um Gesichtsbedeckungen zu erkennen, verwenden Sie den API-Vorgang DetectProtectiveEquipmentohne Speicherung. Das Bild wird aus dem lokalen Dateisystem geladen. Sie stellen das Eingabebild an `DetectProtectiveEquipment` als Bild-Byte-Array (base64-kodierte Bild-Bytes) zur Verfügung. Weitere Informationen finden Sie unter Arbeiten mit Bildern.

Im Beispiel wird ein Begrenzungsrahmen um die erkannten Gesichtsbedeckungen herum angezeigt. Der Begrenzungsrahmen ist grün, wenn die Gesichtsbedeckung den Körperteil vollständig bedeckt. Andernfalls wird ein roter Begrenzungsrahmen angezeigt. Als Warnung wird innerhalb des Begrenzungsrahmens der Gesichtsbedeckung ein gelber Begrenzungsrahmen angezeigt, wenn die Erkennungssicherheit unter dem angegebenen Zuverlässigkeitswert liegt. Wenn keine Gesichtsbedeckung erkannt wird, wird ein roter Begrenzungsrahmen um die Person herum gezeichnet.

Die Bildausgabe sieht folgendermaßen oder ähnlich aus.

So zeigen Sie Begrenzungsrahmen auf erkannten Gesichtsbedeckungen an

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `DetectProtectiveEquipment`-Operation. Informationen zur Anzeige von Begrenzungsrahmen in einem Bild finden Sie unter Anzeigen von Begrenzungsrahmen.

   Java

   In der Funktion `main` ändern Sie Folgendes:

- Der Wert von `photo` zum Pfad- und Dateinamen einer lokalen Bilddatei (PNG oder JPEG).
- Der Wert von `confidence` bis zum gewünschten Zuverlässigkeitsniveau (50–100).

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


import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import com.amazonaws.util.IOUtils;

import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.BoundingBox;
import
 com.amazonaws.services.rekognition.model.DetectProtectiveEquipmentRequest;
import com.amazonaws.services.rekognition.model.DetectProtectiveEquipmentResult;
import com.amazonaws.services.rekognition.model.EquipmentDetection;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.ProtectiveEquipmentBodyPart;
import com.amazonaws.services.rekognition.model.ProtectiveEquipmentPerson;

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

    private static final long serialVersionUID = 1L;

    BufferedImage image;
```

```
    static int scale;
    DetectProtectiveEquipmentResult result;
    float confidence=80;

    public PPEBoundingBox(DetectProtectiveEquipmentResult ppeResult,
BufferedImage bufImage, float requiredConfidence) throws Exception {
        super();
        scale = 2; // increase to shrink image size.

        result = ppeResult;
        image = bufImage;

        confidence=requiredConfidence;
    }
    // Draws the bounding box around the detected faces.
    public void paintComponent(Graphics g) {
        float left = 0;
        float top = 0;
        int height = image.getHeight(this);
        int width = image.getWidth(this);
        int offset=20;

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

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

        // Iterate through detected persons and display bounding boxes.
        List<ProtectiveEquipmentPerson> persons = result.getPersons();

        for (ProtectiveEquipmentPerson person: persons) {
            BoundingBox boxPerson = person.getBoundingBox();
            left = width * boxPerson.getLeft();
            top = height * boxPerson.getTop();
            Boolean foundMask=false;

            List<ProtectiveEquipmentBodyPart> bodyParts=person.getBodyParts();

            if (bodyParts.isEmpty()==false)
             {
                    //body parts detected

                    for (ProtectiveEquipmentBodyPart bodyPart: bodyParts) {
```

```
                        List<EquipmentDetection>
equipmentDetections=bodyPart.getEquipmentDetections();

                        for (EquipmentDetection item: equipmentDetections) {

                            if (item.getType().contentEquals("FACE_COVER"))
                            {
                                // Draw green or red bounding box depending on
mask coverage.
                                foundMask=true;
                                BoundingBox box =item.getBoundingBox();
                                left = width * box.getLeft();
                                top = height * box.getTop();
                                Color maskColor=new Color( 0, 212, 0);

                                if (item.getCoversBodyPart().getValue()==false)
{
                                    // red bounding box
                                    maskColor=new Color( 255, 0, 0);
                                }
                                g2d.setColor(maskColor);
                                g2d.drawRect(Math.round(left / scale),
Math.round(top / scale),
                                        Math.round((width * box.getWidth()) /
scale), Math.round((height * box.getHeight())) / scale);

                                // Check confidence is > supplied confidence.
                                if (item.getCoversBodyPart().getConfidence()<
confidence)
                                {
                                    // Draw a yellow bounding box inside face
mask bounding box
                                    maskColor=new Color( 255, 255, 0);
                                    g2d.setColor(maskColor);
                                    g2d.drawRect(Math.round((left + offset) /
scale),
                                            Math.round((top + offset) / scale),
                                            Math.round((width *
box.getWidth())- (offset * 2 ))/ scale,
                                            Math.round((height *
box.getHeight()) -( offset* 2)) / scale);
                                }
```

```
                            }
                        }

                    }

                }

            // Didn't find a mask, so draw person bounding box red
            if (foundMask==false) {

                left = width * boxPerson.getLeft();
                top = height * boxPerson.getTop();
                g2d.setColor(new Color(255, 0, 0));
                g2d.drawRect(Math.round(left / scale), Math.round(top / scale),
                        Math.round(((width) * boxPerson.getWidth()) / scale),
Math.round((height * boxPerson.getHeight())) / scale);
            }
         }

   }


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

       String photo = "photo";

       float confidence =80;


       int height = 0;
       int width = 0;

       BufferedImage image = null;
       ByteBuffer imageBytes;

       // Get image bytes for call to DetectProtectiveEquipment
       try (InputStream inputStream = new FileInputStream(new File(photo))) {
           imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
       }

       //Get image for display
       InputStream imageBytesStream;
       imageBytesStream = new ByteArrayInputStream(imageBytes.array());
```

```
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        image=ImageIO.read(imageBytesStream);
        ImageIO.write(image, "jpg", baos);
        width = image.getWidth();
        height = image.getHeight();

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


        // Call DetectProtectiveEquipment
        DetectProtectiveEquipmentRequest request = new
 DetectProtectiveEquipmentRequest()
                .withImage(new Image()
                        .withBytes(imageBytes));

        DetectProtectiveEquipmentResult result =
 rekognitionClient.detectProtectiveEquipment(request);


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

    }
}
```

Java V2

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.*;
```

```
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import
 software.amazon.awssdk.services.rekognition.model.DetectProtectiveEquipmentRequest;
import software.amazon.awssdk.services.rekognition.model.EquipmentDetection;
import
 software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentBodyPart;
import
 software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentPerson;
import
 software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentSummarizationAttri
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
 software.amazon.awssdk.services.rekognition.model.DetectProtectiveEquipmentResponse;
//snippet-end:[rekognition.java2.display_mask.import]

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

 DetectProtectiveEquipmentResponse result;
 static BufferedImage image;
 static int scale;
 float confidence;
```

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

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

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

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

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

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

 // snippet-start:[rekognition.java2.display_mask.main]
public static void displayGear(S3Client s3,
                                    RekognitionClient rekClient,
                                    String sourceImage,
                                    String bucketName) {
    float confidence = 80;
    byte[] data = getObjectBytes(s3, bucketName, sourceImage);
    InputStream is = new ByteArrayInputStream(data);

    try {
```

```
        ProtectiveEquipmentSummarizationAttributes summarizationAttributes =
 ProtectiveEquipmentSummarizationAttributes.builder()
             .minConfidence(70F)
             .requiredEquipmentTypesWithStrings("FACE_COVER")
             .build();

        SdkBytes sourceBytes = SdkBytes.fromInputStream(is);
        image = ImageIO.read(sourceBytes.asInputStream());

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

        DetectProtectiveEquipmentRequest request =
 DetectProtectiveEquipmentRequest.builder()
             .image(souImage)
             .summarizationAttributes(summarizationAttributes)
             .build();

        DetectProtectiveEquipmentResponse result =
 rekClient.detectProtectiveEquipment(request);
        JFrame frame = new JFrame("Detect PPE");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        PPEBoundingBoxFrame panel = new PPEBoundingBoxFrame(result, image,
 confidence);
        panel.setPreferredSize(new Dimension(image.getWidth() / scale,
 image.getHeight() / scale));
        frame.setContentPane(panel);
        frame.pack();
        frame.setVisible(true);

    } catch (RekognitionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

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

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

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

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

public PPEBoundingBoxFrame(DetectProtectiveEquipmentResponse ppeResult,
BufferedImage bufImage, float requiredConfidence) {
    super();
    scale = 1; // increase to shrink image size.
    result = ppeResult;
    image = bufImage;
    confidence=requiredConfidence;
}

// Draws the bounding box around the detected masks.
public void paintComponent(Graphics g) {
    float left = 0;
    float top = 0;
    int height = image.getHeight(this);
    int width = image.getWidth(this);
    int offset=20;

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

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

    // Iterate through detected persons and display bounding boxes.
    List<ProtectiveEquipmentPerson> persons = result.persons();
    for (ProtectiveEquipmentPerson person: persons) {
```

```
        List<ProtectiveEquipmentBodyPart> bodyParts=person.bodyParts();
        if (!bodyParts.isEmpty()){
            for (ProtectiveEquipmentBodyPart bodyPart: bodyParts) {
                List<EquipmentDetection>
 equipmentDetections=bodyPart.equipmentDetections();
                for (EquipmentDetection item: equipmentDetections) {

                    String myType = item.type().toString();
                    if (myType.compareTo("FACE_COVER") ==0) {

                        // Draw green bounding box depending on mask coverage.
                        BoundingBox box =item.boundingBox();
                        left = width * box.left();
                        top = height * box.top();
                        Color maskColor=new Color( 0, 212, 0);

                        if (item.coversBodyPart().equals(false)) {
                            // red bounding box.
                            maskColor=new Color( 255, 0, 0);
                        }
                        g2d.setColor(maskColor);
                        g2d.drawRect(Math.round(left / scale), Math.round(top /
 scale),
                                Math.round((width * box.width()) / scale),
 Math.round((height * box.height())) / scale);

                        // Check confidence is > supplied confidence.
                        if (item.coversBodyPart().confidence() < confidence) {
                            // Draw a yellow bounding box inside face mask
 bounding box.
                            maskColor=new Color( 255, 255, 0);
                            g2d.setColor(maskColor);
                            g2d.drawRect(Math.round((left + offset) / scale),
                                    Math.round((top + offset) / scale),
                                    Math.round((width * box.width())- (offset *
 2 ))/ scale,
                                    Math.round((height * box.height()) -
( offset* 2)) / scale);
                        }
                    }
                }
            }
        }
```

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

Python

In der Funktion `main` ändern Sie Folgendes:

- Der Wert von `photo` zum Pfad- und Dateinamen einer lokalen Bilddatei (PNG oder JPEG).
- Der Wert von `confidence` bis zum gewünschten Zuverlässigkeitsniveau (50–100).
- Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
#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

Ändern Sie im folgenden CLI-Beispiel den Wert der unten aufgeführten Argumente:

- Der Wert von `photo` zum Pfad- und Dateinamen einer lokalen Bilddatei (PNG oder JPEG).
- Der Wert von `confidence` bis zum gewünschten Zuverlässigkeitsniveau (50–100).
- Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
                                aws rekognition detect-protective-equipment
 --image "{"S3Object":{"Bucket":"amzn-s3-demo-bucket","Name":"image-name"}}" --
profile profile-name \
                                --summarization-attributes
 "{"MinConfidence":MinConfidenceNumber,"RequiredEquipmentTypes":["FACE_COVER"]}"
```

Wenn Sie auf einem Windows-Gerät auf die CLI zugreifen, verwenden Sie doppelte Anführungszeichen anstelle von einfachen Anführungszeichen und maskieren Sie die inneren doppelten Anführungszeichen durch einen Backslash (d. h. \), um eventuell auftretende Parserfehler zu beheben. Sehen Sie sich zum Beispiel Folgendes an:

```
                                aws rekognition detect-protective-equipment  --
image "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-bucket\",\"Name\":\"image-name
\"}}" \
                                --profile profile-name --summarization-
attributes "{\"MinConfidence\":MinConfidenceNumber,\"RequiredEquipmentTypes\":
[\"FACE_COVER\"]}"
```

Erkennen von Prominenten

Amazon Rekognition macht es Kunden leicht, mithilfe von Machine Learning Zehntausende bekannter Persönlichkeiten in Bildern und Videos automatisch zu erkennen. Die von der API zur Erkennung von Prominenten bereitgestellten Metadaten reduzieren den wiederholten manuellen Aufwand, der erforderlich ist, um Inhalte zu taggen und sie leicht durchsuchbar zu machen, erheblich.

Aufgrund der raschen Verbreitung von Bild- und Videoinhalten haben Medienunternehmen oft Schwierigkeiten, ihre Medienkataloge in großem Umfang zu organisieren, zu durchsuchen und zu nutzen. Nachrichtensender und Sportsender müssen Bilder und Videos häufig schnell finden, um auf aktuelle Ereignisse reagieren und relevante Programme erstellen zu können. Unzureichende Metadaten erschweren diese Aufgaben, aber mit Amazon Rekognition können Sie große Mengen neuer oder archivierter Inhalte automatisch taggen, sodass sie leicht nach einer umfassenden Auswahl an internationalen, weithin bekannten Prominenten wie Schauspielern, Sportlern und Erstellern von Online-Inhalten durchsucht werden können.

Die Amazon-Rekognition-Erkennung von Prominenten ist ausschließlich für den Einsatz in Fällen konzipiert, in denen Sie davon ausgehen, dass ein Bild oder Video eine bekannte Berühmtheit enthält. Weitere Informationen zum Erkennen von Gesichtern, die nicht zu den Prominenten gehören, finden Sie unter Gesichtssuche in einer Sammlung.

> **Note**
>
> Wenn Sie ein Star sind und nicht in diese Funktion aufgenommen werden möchten, wenden Sie sich an den AWS Support oder senden Sie eine E-Mail an `<rekognition-celebrity-opt-out@amazon .com>`.

Themen

- Prominentenerkennung im Vergleich zur Gesichtssuche
- Erkennen von Prominenten in einem Bild
- Erkennen von Prominenten in einem gespeicherten Video
- Abrufen von Informationen über einen Prominenten

Prominentenerkennung im Vergleich zur Gesichtssuche

Amazon Rekognition bietet Funktionen sowohl zur Prominentenerkennung als auch zur Gesichtserkennung. Bezüglich Anwendungsfälle und bewährter Methoden gibt es einige wichtige Unterschiede zwischen diesen Funktionen.

Prominentenerkennung verfügt bei der Auslieferung bereits über die Fähigkeit, Hunderttausende bekannter Persönlichkeiten in Bereichen wie Sport, Medien, Politik und der Geschäftswelt erkennen zu können. Diese Funktionalität wurde entwickelt, sodass Sie großer Mengen an Bildern oder Videos leichter durchsuchen können, um eine kleine Zusammenstellung zu identifizieren, in der wahrscheinlich ein bestimmter Prominenter enthalten ist. Sie ist nicht dazu bestimmt, Gesichter anderer Personen abzugleichen, bei denen es sich nicht um Prominente handelt. In Situationen, in denen die Genauigkeit des Prominentenabgleichs wichtig ist, wird empfohlen, einen menschlichen Operatoren mit dem Durchsuchen dieser kleineren Menge markierter Inhalte zu betrauen, um ein hohes Maß an Genauigkeit sicherzustellen und entsprechendes menschliches Ermessen anzuwenden. Prominentenerkennung darf nicht auf eine Art und Weise angewendet werden, die zu einer Beeinträchtigung von Bürgerrechten führen würde.

Die Gesichtserkennung ist dagegen eine allgemeinere Funktionalität, mit der Sie eigene Gesichtssammlungen mit eigenen Gesichtsvektoren erstellen können, um Identitäten zu prüfen oder nach anderen Personen, nicht nur nach Prominenten, zu suchen. Die Gesichtserkennung kann für Anwendungen wie Authentifizierung des Gebäudezutritts, öffentliche Sicherheit und soziale Medien eingesetzt werden. In allen diesen Fällen wird empfohlen, dass Sie bewährte Methoden, angemessene Vertrauensschwellen (einschließlich 99 % für Anwendungsfälle, die die öffentliche Sicherheit betreffen) und eine menschliche Überprüfung in Situationen verwenden, in denen die Genauigkeit des Abgleichs wichtig ist.

Weitere Informationen finden Sie unter Gesichtssuche in einer Sammlung.

Erkennen von Prominenten in einem Bild

Um Prominente in Bildern zu erkennen und um zusätzliche Informationen über sie zu erhalten, verwenden Sie die nicht speichernde API-Operation RecognizeCelebrities. Sie können beispielsweise in sozialen Medien oder Nachrichten- und Unterhaltungsindustrien, in denen das Sammeln von Informationen oft zeitkritisch sein kann, mit der Operation `RecognizeCelebrities`bis zu 64 Prominente in einem Bild identifizieren und Links zu Webseiten von Prominenten zurückgeben, sofern diese verfügbar sind. Amazon Rekognition merkt sich nicht, in welchem Bild die Prominenten entdeckt wurden. Ihre Anwendung muss diese Informationen speichern.

Wenn Sie die zusätzlichen Informationen für einen Prominenten nicht gespeichert haben, die von `RecognizeCelebrities` zurückgegeben wurden, diese aber erhalten möchten, ohne das Bild erneut zu analysieren, verwenden Sie GetCelebrityInfo. Um `GetCelebrityInfo` aufzurufen, benötigen Sie die eindeutige Kennung, die Amazon Rekognition jedem Prominenten zuweist. Die Kennung wird als Teil der `RecognizeCelebrities`-Antwort für jeden erkannten Prominenten zurückgegeben.

Wenn Sie eine große Sammlung von Bildern für die Erkennung von Prominenten verarbeiten, empfiehlt sich der Einsatz von AWS Batch zum Verarbeiten von Aufrufen an `RecognizeCelebrities` in Stapeln im Hintergrund. Wenn du deiner Sammlung ein neues Bild hinzufügst, kannst du eine AWS Lambda Funktion verwenden, um Prominente zu erkennen, indem du anrufst, `RecognizeCelebrities` während das Bild in einen S3-Bucket hochgeladen wird.

Anrufen RecognizeCelebrities

Sie können das Eingabebild als Bild-Byte-Array (Base64-kodierte Bildbytes) oder als Amazon S3 S3-Objekt bereitstellen, indem Sie entweder das AWS Command Line Interface (AWS CLI) oder das AWS-SDK verwenden. In diesem AWS CLI Verfahren laden Sie ein Bild im JPG- oder PNG-Format in einen S3-Bucket hoch. In den AWS SDK-Verfahren verwenden Sie ein Bild, das aus Ihrem lokalen Dateisystem geladen wurde. Weitere Informationen zu Eingabebild-Empfehlungen finden Sie unter Arbeiten mit Bildern.

Zum Ausführen dieses Verfahrens benötigen Sie eine Bilddatei mit einem oder mehreren Gesichtern von Prominenten.

Erkennen von Prominenten in einem Bild

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`- und `AmazonS3ReadOnlyAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `RecognizeCelebrities`-Operation.

Java

Dieses Beispiel zeigt Informationen über die Prominenten an, die in einem Bild erkannt werden.

Ändern Sie den Wert von `photo` in den Pfad und Dateinamen einer Bilddatei mit einem oder mehreren prominenten Gesichtern.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
//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

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `recognize-celebrities` CLI-Vorgang an.

Ändern Sie `amzn-s3-demo-bucket` in den Namen eines Amazon-S3-Buckets, der ein Bild enthält. Ändern Sie `input.jpg` in den Dateinamen eines Bildes mit einem oder mehreren prominenten Gesichtern.

Ersetzen Sie den Wert von `profile_name` mit dem Namen Ihres Entwicklerprofils.

```
aws rekognition recognize-celebrities \
  --image "S3Object={Bucket=amzn-s3-demo-bucket,Name=input.jpg}"
```

Wenn Sie auf einem Windows-Gerät auf die CLI zugreifen, verwenden Sie doppelte Anführungszeichen anstelle von einfachen Anführungszeichen und maskieren Sie die inneren doppelten Anführungszeichen durch einen Backslash (d. h. \), um eventuell auftretende Parserfehler zu beheben. Sehen Sie sich zum Beispiel Folgendes an:

```
                                    aws rekognition recognize-celebrities --
image \
                                    "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --profile profile-name
```

Python

Dieses Beispiel zeigt Informationen über die Prominenten an, die in einem Bild erkannt werden.

Ändern Sie den Wert von `photo` in den Pfad und Dateinamen einer Bilddatei mit einem oder mehreren prominenten Gesichtern.

Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
#Copyright 2018 Amazon.com, Inc. or its 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

Dieses Beispiel zeigt Informationen über die Prominenten an, die in einem Bild erkannt werden.

Ändern Sie den Wert von `photo` in den Pfad und Dateinamen einer Bilddatei mit einem oder mehreren prominenten Gesichtern. Ändern Sie den Wert von `amzn-s3-demo-bucket` in den Namen des S3-Buckets, der die angegebene Bilddatei enthält. Ändern Sie den Wert von `REGION` in den Namen der Region, die Ihrem Benutzer zugeordnet ist. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
// 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

Dieses Beispiel zeigt Informationen über die Prominenten an, die in einem Bild erkannt werden.

Ändern Sie den Wert von `photo` in den Pfad und Dateinamen einer Bilddatei mit einem oder mehreren prominenten Gesichtern (JPG- oder 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. Notieren Sie den Wert eines der IDs abgebildeten Prominenten. Sie brauchen sie in Abrufen von Informationen über einen Prominenten.

RecognizeCelebrities Operationsanfrage

Die Eingabe in `RecognizeCelebrities` ist ein Bild. In diesem Beispiel wird das Bild als Bild-Bytes übergeben. Weitere Informationen finden Sie unter Arbeiten mit Bildern.

```
{
    "Image": {
        "Bytes": "/AoSiyvFpm....."
    }
}
```

RecognizeCelebrities Antwort auf die Operation

Es folgt ein Beispiel für die Eingabe und Ausgabe im JSON-Format für `RecognizeCelebrities`.

`RecognizeCelebrities` gibt ein Array von erkannten Prominenten und ein Array von nicht erkannten Gesichtern zurück. Beachten Sie im Beispiel Folgendes:

- Erkannte Prominente – `Celebrities` ist ein Array erkannter Prominenter. Jedes Celebrity-Objekt im Array enthält den Namen des Prominenten und eine Liste mit URLs Verweisen auf verwandte Inhalte, z. B. den IMDB- oder Wikidata-Link des Prominenten. Amazon Rekognition gibt ein ComparedFaceObjekt zurück, anhand dessen Ihre Anwendung ermitteln kann, wo sich das Gesicht des Prominenten auf dem Bild befindet, sowie eine eindeutige Kennung für den Star. Verwenden Sie die eindeutige Kennung zum Abrufen von Prominenten-Informationen zu einem späteren Zeitpunkt mit der API-Operation GetCelebrityInfo.
- Nicht erkannte Gesichter – `UnrecognizedFaces`ist ein Array von Gesichtern, die keinen Prominenten entsprechen. Jedes ComparedFace-Objekt im Array enthält einen Begrenzungsrahmen (sowie andere Informationen), die Sie für die Lokalisierung des Gesichts im Bild verwenden können.

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

Erkennen von Prominenten in einem gespeicherten Video

Die Prominenten-Erkennung mit Amazon Rekognition Video ist eine asynchrone Operation. Verwenden Sie, um Prominente in einem gespeicherten Video zu erkennen, um die StartCelebrityRecognitionVideoanalyse zu starten. Das Amazon-Simple-Notification-Service-Thema, zu dem Amazon Rekognition Video die Ergebnisse der Objekterkennung und den Abschlussstatus einer Videoanalyse-Operation veröffentlicht. Wenn die Videoanalyse erfolgreich ist, rufen Sie GetCelebrityRecognition auf, um die Analyseergebnisse abzurufen. Weitere Informationen zum Starten der Videoanalyse und zum Abrufen der Ergebnisse finden Sie unter Amazon-Rekognition-Video-Operationen aufrufen.

Diese Prozedur erweitert den Code in Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK), der eine Amazon-SQS-Warteschlange verwendet, um den Abschlussstatus einer Videoanalyseanforderung zu erhalten. Zum Ausführen dieser Prozedur benötigen Sie eine Videodatei mit einem oder mehreren Gesichtern von Prominenten.

So erkennen Sie Prominente in einem Video, das in einem Amazon-S3-Bucket (SDK) gespeichert ist

1. Führen Sie Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) aus.
2. Fügen Sie den folgenden Code in der Klasse `VideoDetect` ein, die Sie in Schritt 1 erstellt haben.

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);

      }
```

Ersetzen Sie in der Funktion `main` die Zeile:

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

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

mit:

```
        StartCelebrityDetection(amzn-s3-demo-bucket, video);

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

Java V2

Dieser Code stammt aus dem GitHub Beispiel-Repository des AWS Documentation SDK. Das vollständige Beispiel finden Sie hier.

```
//snippet-start:[rekognition.java2.recognize_video_celebrity.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;
//snippet-end:[rekognition.java2.recognize_video_celebrity.import]
```

```
/**
*  To run this code example, ensure that you perform the Prerequisites as stated
 in the Amazon Rekognition Guide:
*  https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-
sqs.html
*
* Also, ensure that set up your development environment, including your
 credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/

public class RecognizeCelebritiesVideo {

private static String startJobId ="";

public static void main(String[] args) {

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

   if (args.length != 4) {
       System.out.println(usage);
       System.exit(1);
   }

   String bucket = args[0];
   String video = args[1];
   String topicArn = args[2];
   String roleArn = args[3];
   Region region = Region.US_EAST_1;
```

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

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

  StartCelebrityDetection(rekClient, channel, bucket, video);
  GetCelebrityDetectionResults(rekClient);
  System.out.println("This example is done!");
  rekClient.close();
}

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

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

       StartCelebrityRecognitionRequest recognitionRequest =
 StartCelebrityRecognitionRequest.builder()
           .jobTag("Celebrities")
           .notificationChannel(channel)
           .video(vidOb)
           .build();

       StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
 rekClient.startCelebrityRecognition(recognitionRequest);
       startJobId = startCelebrityRecognitionResult.jobId();

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

```
        System.exit(1);
    }
}

public static void GetCelebrityDetectionResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetCelebrityRecognitionResponse recognitionResponse = null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (recognitionResponse !=null)
                paginationToken = recognitionResponse.nextToken();

            GetCelebrityRecognitionRequest recognitionRequest =
 GetCelebrityRecognitionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {
                recognitionResponse =
 rekClient.getCelebrityRecognition(recognitionRequest);
                status = recognitionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData=recognitionResponse.videoMetadata();
```

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

            List<CelebrityRecognition> celebs= recognitionResponse.celebrities();
            for (CelebrityRecognition celeb: celebs) {
                long seconds=celeb.timestamp()/1000;
                System.out.print("Sec: " + seconds + " ");
                CelebrityDetail details=celeb.celebrity();
                System.out.println("Name: " + details.name());
                System.out.println("Id: " + details.id());
                System.out.println();
            }

        } while (recognitionResponse.nextToken() != null);

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

Python

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

    # ============== Celebrities ===============
    def StartCelebrityDetection(self):
        response=self.rek.start_celebrity_recognition(Video={'S3Object':
 {'Bucket': self.bucket, 'Name': self.video}},
            NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
 self.snsTopicArn})

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

    def GetCelebrityDetectionResults(self):
```

```
        maxResults = 10
        paginationToken = ''
        finished = False

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

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

            for celebrityRecognition in response['Celebrities']:
                print('Celebrity: ' +
                    str(celebrityRecognition['Celebrity']['Name']))
                print('Timestamp: ' + str(celebrityRecognition['Timestamp']))
                print()

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

Ersetzen Sie in der Funktion `main` die folgenden Zeilen:

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

mit:

```
    analyzer.StartCelebrityDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetCelebrityDetectionResults()
```

Node.JS

Ersetzen Sie im folgenden Codebeispiel für Node.Js den Wert von `amzn-s3-demo-bucket` durch den Namen des S3-Buckets, der Ihr Video enthält, und den Wert von `videoName`

durch den Namen der Videodatei. Außerdem müssen Sie den Wert von `roleArn` durch den ARN ersetzen, der Ihrer IAM-Servicerolle zugeordnet ist. Ersetzen Sie abschließend den Wert von `region` durch den Namen der Betriebsregion, die Ihrem Konto zugeordnet ist. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

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

// Import required AWS SDK clients and commands for Node.js
import { CreateQueueCommand, GetQueueAttributesCommand, GetQueueUrlCommand,
  SetQueueAttributesCommand, DeleteQueueCommand, ReceiveMessageCommand,
 DeleteMessageCommand } from  "@aws-sdk/client-sqs";
import {CreateTopicCommand, SubscribeCommand, DeleteTopicCommand } from "@aws-
sdk/client-sns";
import  { SQSClient } from "@aws-sdk/client-sqs";
import  { SNSClient } from "@aws-sdk/client-sns";
import  { RekognitionClient, StartLabelDetectionCommand,
 GetLabelDetectionCommand,
  StartCelebrityRecognitionCommand, GetCelebrityRecognitionCommand} from "@aws-
sdk/client-rekognition";
import { stdout } from "process";
import {fromIni} from '@aws-sdk/credential-providers';

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

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

```
var startJobId = ""

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

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

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

```
          Effect: "Allow",
          Principal: {AWS: "*"},
          Action: "SQS:SendMessage",
          Resource: queueArn,
          Condition: {
            ArnEquals: {
              'aws:SourceArn': topicArn
            }
          }
        }
      ]
    };

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

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

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

const getCelebrityRecognitionResults = async(startJobId) =>{
  try {
      //Initiate label detection and update value of startJobId with returned
 Job ID
```

```
      var maxResults = 10
      var paginationToken = ''
      var finished = false

      while (finished == false){
          var response = await rekClient.send(new
 GetCelebrityRecognitionCommand({JobId: startJobId, MaxResults: maxResults,
              NextToken: paginationToken}))
          console.log(response.VideoMetadata.Codec)
          console.log(response.VideoMetadata.DurationMillis)
          console.log(response.VideoMetadata.Format)
          console.log(response.VideoMetadata.FrameRate)
          response.Celebrities.forEach(celebrityRecognition => {
              console.log(`Celebrity: ${celebrityRecognition.Celebrity.Name}`)
              console.log(`Timestamp: ${celebrityRecognition.Timestamp}`)
              console.log()
          })
          // Searh for pagination token, if found, set variable to next token
          if (String(response).includes("NextToken")){
              paginationToken = response.NextToken

          }else{
              finished = true
          }
      }
    } catch (err) {
      console.log("Error", err);
    }
  };

// Checks for status of job completion
const getSQSMessageSuccess = async(sqsQueueUrl, startJobId) => {
  try {
    // Set job found and success status to false initially
    var jobFound = false
    var succeeded = false
    var dotLine = 0
    // while not found, continue to poll for response
    while (jobFound == false){
      var sqsReceivedResponse = await sqsClient.send(new
 ReceiveMessageCommand({QueueUrl:sqsQueueUrl,
        MaxNumberOfMessages:'ALL', MaxNumberOfMessages:10}));
      if (sqsReceivedResponse){
        var responseString = JSON.stringify(sqsReceivedResponse)
```

```
        if (!responseString.includes('Body')){
          if (dotLine < 40) {
            console.log('.')
            dotLine = dotLine + 1
          }else {
            console.log('')
            dotLine = 0
          };
          stdout.write('', () => {
            console.log('');
          });
          await new Promise(resolve => setTimeout(resolve, 5000));
          continue
        }
      }

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

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

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

runCelebRecognitionAndGetResults()
```

CLI

Führen Sie den folgenden AWS CLI Befehl aus, um mit der Erkennung von Prominenten in einem Video zu beginnen.

```
aws rekognition start-celebrity-recognition --video "{"S3Object":
{"Bucket":"amzn-s3-demo-bucket","Name":"video-name"}}" \
--notification-channel "{"SNSTopicArn":"topic-arn","RoleArn":"role-arn"}" \
--region region-name --profile profile-name
```

Aktualisieren Sie die folgenden Werte:

- Ändern Sie `amzn-s3-demo-bucket` und `video-name` in den Amazon-S3-Bucket-Namen und den Dateinamen, die Sie in Schritt 2 angegeben haben.
- Ändern Sie `region-name` in die von Ihnen verwendete AWS-Region.
- Ersetzen Sie den Wert von `profile-name` mit dem Namen Ihres Entwicklerprofils.
- Ändern Sie `topic-ARN` in den ARN des Amazon-SNS-Themas, das Sie in Schritt 3 von Amazon Rekognition Video konfigurieren erstellt haben.
- Ändern Sie `role-ARN` in den ARN der IAM-Servicerolle, die Sie in Schritt 7 von Amazon Rekognition Video konfigurieren erstellt haben.

Wenn Sie auf einem Windows-Gerät auf die CLI zugreifen, verwenden Sie doppelte Anführungszeichen anstelle von einfachen Anführungszeichen und maskieren Sie die inneren doppelten Anführungszeichen durch einen Backslash (d. h. \), um eventuell auftretende Parserfehler zu beheben. Ein Beispiel finden Sie unten:

```
aws rekognition start-celebrity-recognition --video "{\"S3Object\":{\"Bucket\":
\"amzn-s3-demo-bucket\",\"Name\":\"video-name\"}}" \
--notification-channel "{\"SNSTopicArn\":\"topic-arn\",\"RoleArn\":\"role-arn
\"}" \
--region region-name --profile profile-name
```

Nachdem Sie das vorangegangene Codebeispiel ausgeführt haben, kopieren Sie die zurückgegebene `jobID` und geben Sie sie an den folgenden `GetCelebrityRecognition`-Befehl weiter, um Ihre Ergebnisse zu erhalten, und ersetzen Sie `job-id-number` durch `jobID`, die Sie zuvor erhalten haben:

```
aws rekognition get-celebrity-recognition --job-id job-id-number --profile
 profile-name
```

> **Note**
>
> Wenn Sie zusätzlich zu Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) bereits ein anderes Videobeispiel ausgeführt haben, ist der zu ersetzende Code möglicherweise anders.

3. Führen Sie den Code aus. Die Informationen über die Prominenten im Video werden angezeigt.

GetCelebrityRecognition Reaktion auf den Vorgang

Nachfolgend finden Sie ein Beispiel einer JSON-Antwort. Die Antwort enthält die folgenden Attribute:

- Erkannte Prominente – `Celebrities` ist ein Array von Prominenten mit den Zeit(en), wann sie in einem Video erkannt wurden. Ein CelebrityRecognition-Objekt wird jedes Mal erzeugt, wenn der Prominente im Video erkannt wird. Jedes `CelebrityRecognition` enthält Informationen über einen erkannten Prominenten (CelebrityDetail) und den Zeitpunkt (`Timestamp`), an dem der Prominente im Video erkannt wird. `Timestamp` wird in Millisekunden ab dem Anfang des Videos gemessen.
- CelebrityDetail— Enthält Informationen über eine anerkannte Berühmtheit. Es enthält den Namen der Berühmtheit (`Name`), die Kennung (`ID`), das bekannte Geschlecht der Berühmtheit (`KnownGender`) und eine Liste von URLs Verweisen auf verwandte Inhalte (`Urls`). Es beinhaltet auch das Vertrauen, das Amazon Rekognition Video in die Genauigkeit der Erkennung hat, und Details zum Gesicht des Prominenten. FaceDetail Wenn Sie diese Inhalte später benötigen, können Sie `ID` mit getCelebrityInfo verwenden.
- VideoMetadata— Informationen über das Video, das analysiert wurde.

```
{
    "Celebrities": [
        {
            "Celebrity": {
                "Confidence": 0.699999988079071,
                "Face": {
                    "BoundingBox": {
                        "Height": 0.20555555820465088,
                        "Left": 0.029374999925494194,
                        "Top": 0.22333332896232605,
```

```
                    "Width": 0.11562500149011612
                },
                "Confidence": 99.89837646484375,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.06857934594154358,
                        "Y": 0.30842265486717224
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.10396526008844376,
                        "Y": 0.300625205039978
                    },
                    {
                        "Type": "nose",
                        "X": 0.0966852456331253,
                        "Y": 0.34081998467445374
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.075217105448246,
                        "Y": 0.3811396062374115
                    },
                    {
                        "Type": "mouthRight",
                        "X": 0.10744428634643555,
                        "Y": 0.37407416105270386
                    }
                ],
                "Pose": {
                    "Pitch": -0.9784082174301147,
                    "Roll": -8.808176040649414,
                    "Yaw": 20.28228759765625
                },
                "Quality": {
                    "Brightness": 43.312068939208984,
                    "Sharpness": 99.9305191040039
                }
            },
            "Id": "XXXXXX",
            "KnownGender": {
                "Type": "Female"
            },
```

```
                "Name": "Celeb A",
                "Urls": []
            },
            "Timestamp": 367
        },......
    ],
    "JobStatus": "SUCCEEDED",
    "NextToken": "XfXnZKiyMOGDhzBzYUhS5puM+g1IgezqFeYpv/H/+5noP/LmM57FitUAwSQ5D6G4AB/
PNwolrw==",
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 67301,
        "FileExtension": "mp4",
        "Format": "QuickTime / MOV",
        "FrameHeight": 1080,
        "FrameRate": 29.970029830932617,
        "FrameWidth": 1920
    }
}
```

Abrufen von Informationen über einen Prominenten

Mit diesen Prozeduren erhalten Sie Informationen über Prominente, indem Sie die API-Operation getCelebrityInfo verwenden. Der Prominente wird durch die Prominenten-ID identifiziert, die in einem vorherigen Aufruf von RecognizeCelebrities zurückgegeben wurde.

Anrufen GetCelebrityInfo

Diese Verfahren benötigen die ID eines Prominenten, den Amazon Rekognition kennt. Verwenden Sie die Prominenten-ID, die Sie in Erkennen von Prominenten in einem Bild notiert haben.

Abrufen von Informationen eines Prominenten (SDK)

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`- und `AmazonS3ReadOnlyAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Verwenden Sie die folgenden Beispiele zum Aufrufen der `GetCelebrityInfo`-Operation.

   Java

   Dieses Beispiel zeigt den Namen und die Informationen zu einem Prominenten an.

   Ersetze es `id` durch einen der Prominenten, die in IDs angezeigt werdenErkennen von Prominenten in einem Bild.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityInfoRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityInfoResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

                Usage:    <id>

                Where:
                   id - The id value of the celebrity. You can use the
 RecognizeCelebrities example to get the ID value.\s
                """;

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

        String id = args[0];
        Region region = Region.US_WEST_2;
```

```
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        getCelebrityInfo(rekClient, id);
        rekClient.close();
    }

    /**
     * Retrieves information about a celebrity identified in an image.
     *
     * @param rekClient the Amazon Rekognition client used to make the API call
     * @param id the unique identifier of the celebrity
     * @throws RekognitionException if there is an error retrieving the
 celebrity information
     */
    public static void getCelebrityInfo(RekognitionClient rekClient, String id)
 {
        try {
            GetCelebrityInfoRequest info = GetCelebrityInfoRequest.builder()
                    .id(id)
                    .build();

            GetCelebrityInfoResponse response =
 rekClient.getCelebrityInfo(info);
            System.out.println("celebrity name: " + response.name());
            System.out.println("Further information (if available):");
            for (String url : response.urls()) {
                System.out.println(url);
            }

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

AWS CLI

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `get-celebrity-info` CLI-Vorgang an. Ersetze es `ID` durch einen der Prominenten, die in IDs angezeigt

werdenErkennen von Prominenten in einem Bild. Ersetzen Sie den Wert von `profile-name` mit dem Namen Ihres Entwicklerprofils.

```
aws rekognition get-celebrity-info --id celebrity-id --profile profile-name
```

Python

Dieses Beispiel zeigt den Namen und die Informationen zu einem Prominenten an.

Ersetze es `id` durch einen der Prominenten, die in IDs angezeigt werdenErkennen von Prominenten in einem Bild. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
# Copyright 2018 Amazon.com, Inc. or its 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

Dieses Beispiel zeigt den Namen und die Informationen zu einem Prominenten an.

Ersetze es `id` durch einen der Prominenten, die in IDs angezeigt werdenErkennen von Prominenten in einem Bild.

```
//Copyright 2018 Amazon.com, Inc. or its 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 Betriebsanfrage

Es folgt ein Beispiel für die Eingabe und Ausgabe im JSON-Format für `GetCelebrityInfo`.

Die Eingabe in `GetCelebrityInfo` ist die ID für des erforderlichen Prominenten.

```
{
    "Id": "nnnnnnn"
}
```

GetCelebrityInfo Antwort auf die Operation

`GetCelebrityInfo` gibt ein Array (`Urls`) von Links zu Informationen über den angeforderten Prominenten zurück.

```
{
    "Name": "Celebrity Name",
    "Urls": [
        "www.imdb.com/name/nmnnnnnnn"
    ]
}
```

Inhalte moderieren

Sie können Amazon Rekognition verwenden, um unangemessene, unerwünschte oder anstößige Inhalte zu erkennen. Sie können die Rekognition-Moderation APIs in sozialen Medien, Rundfunkmedien, Werbung und E-Commerce-Situationen einsetzen, um ein sichereres Nutzererlebnis zu schaffen, Werbetreibenden Markensicherheitsgarantien zu bieten und lokale und globale Vorschriften einzuhalten.

Heutzutage verlassen sich viele Unternehmen ausschließlich auf menschliche Moderatoren, um Inhalte Dritter oder benutzergenerierter Inhalte zu überprüfen, während andere lediglich auf Beschwerden von Benutzern reagieren, um anstößige oder unangemessene Bilder, Anzeigen oder Videos zu entfernen. Menschliche Moderatoren allein können jedoch nicht skalieren, um diese Anforderungen in ausreichender Qualität oder Geschwindigkeit zu erfüllen, was zu einer schlechten Benutzererfahrung, hohen Skalierungskosten oder sogar zu einem Verlust des Rufs der Marke führt. Durch die Verwendung von Rekognition für die Bild- und Videomoderation können menschliche Moderatoren eine viel kleinere Menge an Inhalten überprüfen, typischerweise 1–5 % des Gesamtvolumens, die bereits durch Machine Learning gekennzeichnet wurden. Auf diese Weise können sie sich auf wertvollere Aktivitäten konzentrieren und trotzdem eine umfassende Moderationsabdeckung zu einem Bruchteil ihrer derzeitigen Kosten erreichen. Um menschliche Belegschaften einzurichten und menschliche Überprüfungsaufgaben durchzuführen, können Sie Amazon Augmented AI verwenden, das bereits in Rekognition integriert ist.

Sie können die Genauigkeit des Deep-Learning-Modells für Moderation mit der Funktion „Benutzerdefinierte Moderation“ verbessern. Mit der benutzerdefinierten Moderation trainieren Sie einen benutzerdefinierten Moderationsadapter, indem Sie Ihre Bilder hochladen und diese Bilder mit Anmerkungen versehen. Der geschulte Adapter kann dann dem DetectModerationLabelsBetrieb zur Verfügung gestellt werden, um dessen Leistung auf Ihren Bildern zu verbessern. Weitere Informationen finden Sie unter Verbesserung der Genauigkeit mit benutzerdefinierter Moderation.

Labels, die von der Moderation von Rekognition-Inhalten unterstützt werden

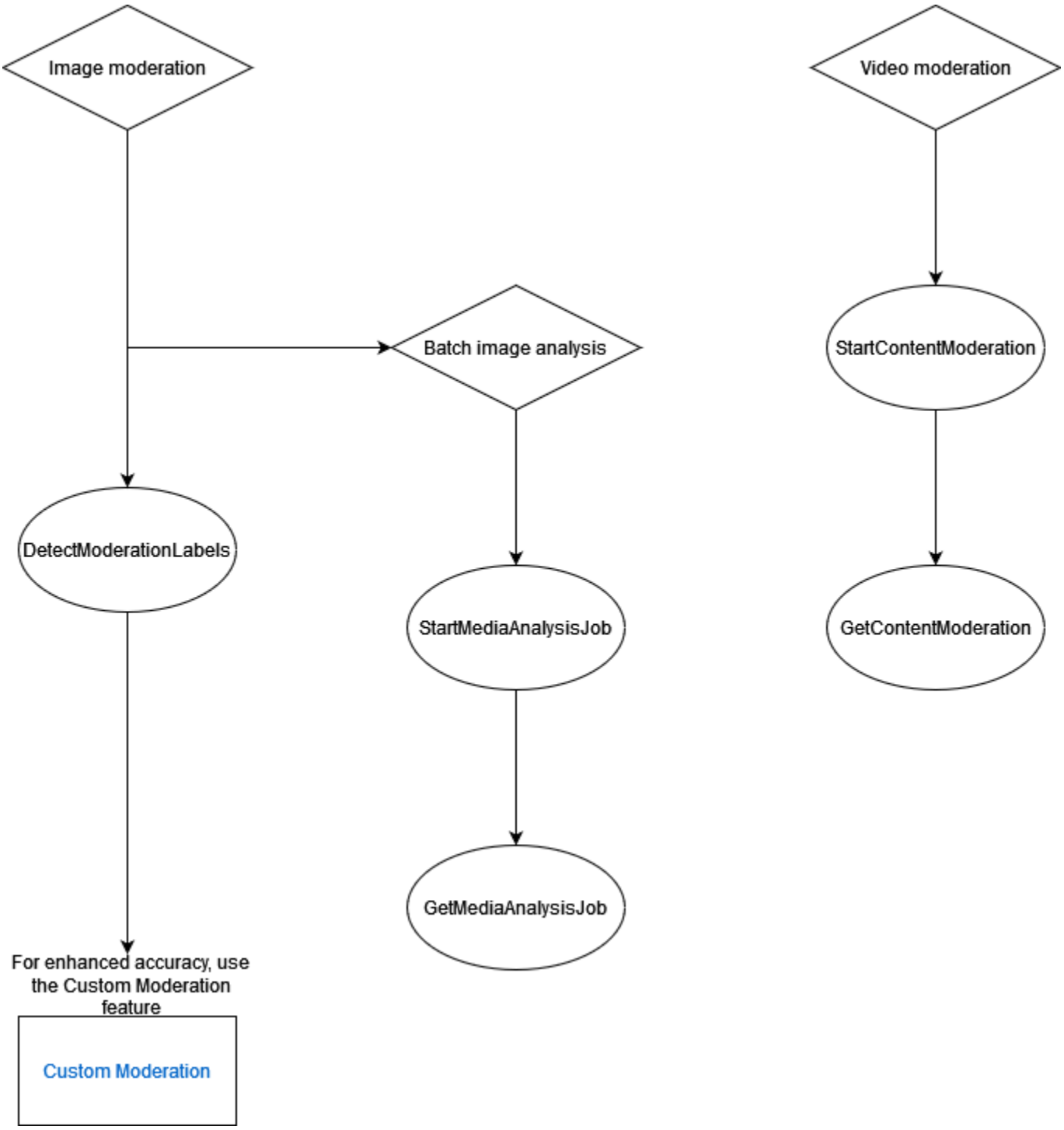
- Um eine Liste der Moderationslabels herunterzuladen, klicken Sie hier.

Themen

- Verwendung der Bild- und Videomoderation APIs
- Content Moderation Version 7 testen und die API-Antwort transformieren

- Erkennen unangemessener Bilder
- Erkennung unangemessener gespeicherter Videos
- Verbesserung der Genauigkeit mit benutzerdefinierter Moderation
- Überprüfen unangemessener Inhalte mit Amazon Augmented AI

Das folgende Diagramm zeigt die Reihenfolge der Aufrufvorgänge, abhängig von Ihren Zielen bei der Verwendung der Bild- oder Videokomponenten der Inhaltsmoderation:



Verwendung der Bild- und Videomoderation APIs

In der Amazon Rekognition Image API können Sie unangemessene, unerwünschte oder anstößige Inhalte synchron DetectModerationLabelsund asynchron mithilfe von AND-Vorgängen erkennen. StartMediaAnalysisJobGetMediaAnalysisJob Sie können die Amazon Rekognition

Video Video-API verwenden, um solche Inhalte asynchron zu erkennen, indem Sie die StartContentModerationOperationen und verwenden. GetContentModeration

Labelkategorien

Amazon Rekognition verwendet eine dreistufige hierarchische Taxonomie, um Kategorien von unangemessenen, unerwünschten oder anstößigen Inhalten zu kennzeichnen. Jedes Label mit Taxonomiestufe 1 (L1) hat eine Reihe von Taxonomie-Labels der Stufe 2 (L2), und einige Labels der Taxonomiestufe 2 können Taxonomie-Labels der Stufe 3 (L3) haben. Dies ermöglicht eine hierarchische Klassifizierung des Inhalts.

Für jedes erkannte Moderationslabel gibt die API auch den zurück`TaxonomyLevel`, der die Ebene (1, 2 oder 3) enthält, zu der das Label gehört. Ein Bild kann beispielsweise gemäß der folgenden Kategorisierung gekennzeichnet werden:

L1: Nicht explizite Nacktheit intimer Bereiche und Küssen, L2: Nicht explizite Nacktheit, L3: Implizite Nacktheit.

> **Note**
>
> Wir empfehlen, L1- oder L2-Kategorien zu verwenden, um Ihre Inhalte zu moderieren, und L3-Kategorien nur, um bestimmte Konzepte zu entfernen, die Sie nicht moderieren möchten (d. h. um Inhalte zu erkennen, die Sie aufgrund Ihrer Moderationsrichtlinie möglicherweise nicht als unangemessene, unerwünschte oder anstößige Inhalte einstufen möchten).

Die folgende Tabelle zeigt die Beziehungen zwischen den Kategorieebenen und die möglichen Bezeichnungen für jede Ebene. Um eine Liste der Moderationslabels herunterzuladen, klicken Sie hier.

Kategorie auf oberster Ebene (L1)	Kategorie der zweiten Ebene (L2)	Kategorie der dritten Stufe (L3)	Definitionen
Explizit	Explizite Nacktheit	Exponierte männliche Genitalien	Männliche Genitalien des Menschen, einschließlich des Penis (ob aufrecht oder schlaff), des

	Hodensacks und aller erkennbaren Schamhaare. Dieser Begriff gilt für sexuelle Aktivitäten oder visuelle Inhalte, bei denen männliche Genitalien ganz oder teilweise abgebildet sind.
Exponierte weibliche Genitalien	Äußere Teile des weiblichen Fortpflanzungssystems, das die Vulva, die Vagina und alle sichtbaren Schamhaare umfasst. Dieser Begriff gilt für Szenarien mit sexueller Aktivität oder visuellen Inhalten, in denen diese Aspekte der weiblichen Anatomie entweder vollständig oder teilweise dargestellt werden.

	Freiliegendes Gesäß oder Anus	Gesäß oder Anus von Menschen, einschließlich Fällen, in denen das Gesäß nackt oder durch durchsichtige Kleidung erkennbar ist. Die Definition gilt insbesondere für Situationen, in denen das Gesäß oder der Anus direkt und vollständig sichtbar sind, ausgenommen Fälle, in denen jegliche Form von Unterwäsche oder Kleidung eine vollständige oder teilweise Abdeckung bietet.
	Freiliegende weibliche Brustwarze	Menschliche weibliche Brustwarzen, einschließlich vollständig sichtbarer und teilweise sichtbarer Aerola (Bereich um die Brustwarzen) und Brustwarzen.

Explizite sexuelle Aktivität	N/A	Darstellung von tatsächlichen oder simulierten sexuellen Handlungen, die menschlichen Geschlechtsverkehr, Oralsex sowie männliche Genitalstimulation und weibliche Genitalstimulation durch andere Körperteile und Gegenstände umfassen. Der Begriff umfasst auch Ejakulation oder Scheidenflüssigkeiten an Körperteilen sowie erotische Praktiken oder Rollenspiele mit Bondage, Disziplin, Dominanz und Unterwerfung sowie Sadomasochismus.
Sexspielzeug	N/A	Gegenstände oder Geräte, die zur sexuellen Stimulation oder zum sexuellen Vergnügen verwendet werden, z. B. Dildo, Vibrator, Buttplug, Beats usw.

Unexplizite Nacktheit intimer Körperteile und Küssen	Nicht explizite Nacktheit	Nackter Rücken	Hinterer Teil des Menschen, in dem der Großteil der Haut vom Hals bis zum Ende der Wirbelsäule sichtbar ist. Dieser Begriff gilt nicht, wenn der Rücken der Person teilweise oder vollständig verschlossen ist.
		Freiliegende männliche Brustwarze	Männliche Brustwarzen beim Menschen, einschließlich teilweise sichtbarer Brustwarzen.
		Teilweise freiliegendes Gesäß	Teilweise freiliegendes menschliches Gesäß. Dieser Begriff umfasst einen aufgrund kurzer Kleidung teilweise sichtbaren Bereich des Gesäßes oder der Pobacken oder den teilweise sichtbaren oberen Teil der Analspalte. Der Begriff gilt nicht für Fälle, in denen das Gesäß vollständig nackt ist.

		Teilweise exponierte weibliche Brust	Teilweise exponierte weibliche Brust, bei der ein Teil der weiblichen Brust sichtbar oder unbedeckt ist, ohne dass die gesamte Brust sichtbar ist. Dieser Begriff gilt, wenn der Bereich der inneren Brustfalte sichtbar ist oder wenn die untere Brustfalte sichtbar ist, wenn die Brustwarze vollständig bedeckt oder verdeckt ist.
		Implizite Nacktheit	Eine nackte Person, entweder oben ohne oder ohne Boden, wobei intime Bereiche wie Gesäß, Brustwarzen oder Genitalien bedeckt, verdeckt oder nicht vollständig sichtbar sind.
	Verstopfte Intimbereiche	Verstopfte weibliche Brustwarze	Visuelle Darstellung einer Situation, in der die Brustwarzen einer Frau durch undurchsichtige Kleidung oder Deckungen bedeckt sind, ihre Form jedoch deutlich sichtbar ist.

		Verstopfte männliche Genitalien	Visuelle Darstellung einer Situation, in der die Genitalien oder der Penis eines Mannes durch undurchsichtige Kleidung oder undurchsichtige Hüllen bedeckt sind, ihre Form jedoch deutlich sichtbar ist. Dieser Begriff gilt, wenn sich die verstopften Genitalien auf dem Bild in Nahaufnahme befinden.
	Auf die Lippen küssen	N/A	Darstellung der Lippen einer Person, die Kontakt mit den Lippen einer anderen Person aufnehmen.
Bademode oder Unterwäsche	Weibliche Bademode oder Unterwäsche	N/A	Menschliche Kleidung für weibliche Badebekleidung (z. B. einteilige Badeanzüge, Bikinis, Tankinis usw.) und Damenunterwäsche (z. B. BHs, Höschen, Slips, Dessous, Tangas usw.)

	Männliche Bademode oder Unterwäsche	N/A	Menschliche Kleidung für männliche Badebekleidung (z. B. Badehosen , Boardshorts, Badeslips usw.) und männliche Unterwäsche (z. B. Slips, Boxershorts usw.)
Gewalt	Waffen	N/A	Instrumente oder Geräte, die verwendet werden, um Lebewesen , Strukturen oder Systemen Schaden zuzufügen. Dazu gehören Schusswaffen (z. B. Waffen, Gewehre, Maschinengewehre usw.), scharfe Waffen (z. B. Schwerter, Messer usw.), Sprengstoffe und Munition (z. B. Raketen, Bomben, Kugeln usw.).

Graphische Gewalt	Waffengewalt	Der Einsatz von Waffen, um sich selbst, anderen Personen oder Eigentum Schaden zuzufügen, zu schädigen, zu verletzen oder zu töten.
	Körperliche Gewalt	Der Akt, anderen Personen oder Eigentum Schaden zuzufügen (z. B. Schlagen, Kämpfen, Haarziehen usw.) oder eine andere Gewalttat, an der eine Menschenmenge oder mehrere Personen beteiligt sind.
	Selbstverletzung	Der Akt, sich selbst Schaden zuzufügen , häufig durch Schneiden von Körperteilen wie Armen oder Beinen, an denen Schnitte normalerweise sichtbar sind.

		Blut und Blut	Visuelle Darstellung von Gewalt an einer Person, einer Gruppe von Personen oder Tieren mit offenen Wunden, Blutvergießen und verstümmelten Körperteilen.
		Explosionen und Explosionen	Darstellung eines heftigen und zerstörerischen Ausbruchs intensiver Flammen mit dichtem Rauch oder Staub und Rauch, die aus dem Boden austreten.
Optisch störend	Tod und Abmagerung	Ausgemergelte Körper	Extrem dünne und unterernährte menschliche Körper mit starker körperlicher Erschöpfung und Erschöpfung von Muskel- und Fettgewebe.
		Leichen	Menschliche Leichen in Form von verstümmelten Körpern, hängenden Leichen oder Skeletten.

	Stürzt ab	Flugzeugabsturz	Zwischenfälle mit Luftfahrzeugen wie Flugzeugen, Hubschraubern oder anderen fliegenden Fahrzeugen, die zu Schäden, Verletzungen oder zum Tod führen. Dieser Begriff gilt, wenn Teile der Luftfahrzeuge sichtbar sind.
Drogen und Tabak	Produkte	Pillen	Kleine, feste, oft runde oder ovale Tische oder Kapseln. Dieser Begriff bezieht sich auf Pillen, die einzeln, in einer Flasche oder einer durchsichtigen Packung angeboten werden, und gilt nicht für die visuelle Darstellung einer Person, die Tabletten einnimmt.

	Drogen und Tabakwaren und -konsum	Rauchen	Der Vorgang des Einatmens , Ausatmens und Entzünden s brennender Substanzen wie Zigaretten, Zigarren, E-Zigaretten, Shisha oder Joint.
Alkohol	Alkoholkonsum	Alkohol trinken	Das Trinken alkoholis cher Getränke aus Flaschen oder Gläsern mit Alkohol oder Spirituosen.
	Alkoholische Getränke	N/A	Nahaufnahme von einer oder mehreren Flaschen Alkohol oder Spirituosen, Gläsern oder Bechern mit Alkohol oder Spirituos en und Gläsern oder Bechern mit Alkohol oder Spirituosen, die von einer Person in der Hand gehalten werden. Dieser Begriff gilt nicht für eine Person, die aus Flaschen oder Gläsern mit Alkohol oder Spirituosen trinkt.

Unhöfliche Gesten	Mittelfinger	N/A	Visuelle Darstellung einer Handbewegung, bei der der Mittelfinger nach oben gestreckt wird, während die anderen Finger nach unten geklappt sind.
Glücksspiel	N/A	N/A	Die Teilnahme an Glücksspielen, um einen Preis in Casinos zu gewinnen, z. B. Spielkarten, Blackjacks, Roulette, Spielautomaten in Casinos usw.
Hasssymbole	Nazipartei	N/A	Visuelle Darstellung von Symbolen, Flaggen oder Gesten im Zusammenhang mit der NSDAP.
	Weiße Vorherrschaft	N/A	Visuelle Darstellung von Symbolen oder Kleidungsstücken, die mit dem Ku-Klux-Klan (KKK) in Verbindung stehen, und von Bildern mit Flaggen der Konföderierten.
	Extremistisch	N/A	Bilder mit Flaggen extremistischer und terroristischer Gruppen.

Nicht für jedes Label in der Kategorie L2 wird ein Label in der Kategorie L3 unterstützt. Darüber hinaus sind die L3-Labels unter den L2-Labels „Produkte“ und „Arzneimittel- und Tabakzubehör und -gebrauch“ nicht vollständig. Diese L2-Labels decken Konzepte ab, die über die genannten L3-Labels hinausgehen. In solchen Fällen werden in der API-Antwort nur L2-Labels zurückgegeben.

Sie bestimmen, inwieweit Inhalte für Ihre Anwendung akzeptabel sind. Vielleicht sind beispielsweise anzügliche Bilder akzeptabel, Bilder mit Nacktheit jedoch nicht. Verwenden Sie zum Filtern von Bildern das Label-Array ModerationLabel, das von `DetectModerationLabels` (Bilder) und `GetContentModeration` (Videos) zurückgegeben wird.

Inhaltstyp

Die API kann auch den animierten oder illustrierten Inhaltstyp identifizieren, und der Inhaltstyp wird als Teil der Antwort zurückgegeben:

- Animierte Inhalte umfassen Videospiele und Animationen (z. B. Zeichentrick, Comics, Manga, Anime).
- Zu den illustrierten Inhalten gehören Zeichnen, Malen und Skizzen.

Wahrscheinlichkeit

Sie können den von Amazon Rekognition verwendeten Zuverlässigkeitsschwellenwert zum Erkennen unangemessener Inhalte durch Angabe des `MinConfidence`-Eingabeparameters festlegen. Für unangemessenen Inhalte, die mit einem niedrigeren Zuverlässigkeitswert als `MinConfidence` erkannt werden, werden keine Labels zurückgegeben.

Die Angabe eines Werts unter 50% führt wahrscheinlich zu einer hohen Anzahl falsch-positiver Ergebnisse (d. h. höherer Erinnerungsvermögen, geringere Genauigkeit). `MinConfidence` Andererseits führt die Angabe eines Werts `MinConfidence` über 50% wahrscheinlich zu einer geringeren Anzahl falsch-positiver Ergebnisse (d. h. geringerer Erinnerungswert, höhere Genauigkeit). Wenn Sie keinen Wert für `MinConfidence` festlegen, gibt Amazon Rekognition Labels für unangemessenen Inhalte zurück, die mit einer Zuverlässigkeit von mindestens 50 % erkannt werden.

Das `ModerationLabel`-Array enthält Labels in den vorangegangenen Kategorien und einen geschätzten Zuverlässigkeitswert für die Genauigkeit der erkannten Inhalte. Es werden ein Label der obersten Kategorie sowie alle erkannten Labels der zweiten Stufe zurückgegeben. Amazon Rekognition kann beispielsweise „Explicit Nudity” (explizite Nacktheit) mit einer hohen

Zuverlässigkeitsbewertung als Label der obersten Kategorie zurückgeben. Dies könnte für Ihre Filteranforderungen ausreichend sein. Gegebenenfalls können Sie jedoch auch den Zuverlässigkeitswert eines Labels der zweiten Stufe (z. B. „Grafische männliche Nacktheit") verwenden, um eine genauere Filterung zu erhalten. Ein Beispiel finden Sie unter Erkennen unangemessener Bilder.

Versionsverwaltung

Amazon Rekognition Image und Amazon Rekognition Video geben beide die Version des Moderationserkennungsmodells zurück, das zur Erkennung unangemessener Inhalte verwendet wird (`ModerationModelVersion`).

Sortieren und Aggregieren

Beim Abrufen von Ergebnissen mit GetContentModeration können Sie Ihre Ergebnisse sortieren und aggregieren.

Sortierreihenfolge: Das Array der zurückgegebenen Label wird nach Zeit sortiert. Um nach dem Label zu sortieren, geben Sie `NAME` im Eingabeparameter `SortBy` für `GetContentModeration` an. Wenn das Label mehrfach im Video erscheint, gibt es mehrere Instances des `ModerationLabel`-Elements.

Labelinformationen — Das ModerationLabels Array-Element enthält ein `ModerationLabel` Objekt, das wiederum den Labelnamen und das Vertrauen von Amazon Rekognition in die Genauigkeit des erkannten Labels enthält. Der Zeitstempel ist die Zeit, zu der das `ModerationLabel` erkannt wurde, definiert als die Anzahl der Millisekunden, die seit dem Start des Videos vergangen sind. Bei nach Video `SEGMENTS` aggregierten Ergebnissen werden die `StartTimestampMillis`-, `EndTimestampMillis`- und `DurationMillis`-Strukturen zurückgegeben, die jeweils die Startzeit, die Endzeit und die Dauer eines Segments definieren.

Aggregation: Gibt an, wie Ergebnisse aggregiert werden, wenn sie zurückgegeben werden. Standardmäßig wird nach `TIMESTAMPS` aggregiert. Sie können sich auch für die Aggregation nach `SEGMENTS` entscheiden, wodurch die Ergebnisse über ein Zeitfenster aggregiert werden. Es werden nur Label zurückgegeben, die während der Segmente erkannt wurden.

Status des benutzerdefinierten Moderationsadapters

Benutzerdefinierte Moderationsadapter können einen der folgenden Status haben: TRAINING_IN_PROGRESS, TRAINING_COMPLETED, TRAINING_FAILED, DELETING,

DEPRECATED oder EXPIRED. Eine vollständige Erläuterung dieser Adapterstatus finden Sie unter Adapter verwalten.

> **Note**
>
> Amazon Rekognition ist keine Autorität auf dem Gebiet unangemessener oder anstößiger Inhalte und erhebt auch nicht den Anspruch, ein umfassender Filter zu sein. Darüber hinaus erkennt die Bild- und Videomoderation APIs nicht, ob ein Bild illegale Inhalte wie CSAM enthält.

Content Moderation Version 7 testen und die API-Antwort transformieren

Rekognition hat das Modell für maschinelles Lernen für die Bild- und Videokomponenten der Etikettenerkennungsfunktion von Content Moderation von Version 6.1 auf Version 7 aktualisiert. Dieses Update verbesserte die Gesamtgenauigkeit und führte mehrere neue Kategorien ein und änderte andere.

Wenn Sie derzeit Videoanwendungen von Version 6.1 verwenden, empfehlen wir Ihnen, die folgenden Maßnahmen zu ergreifen, um einen reibungslosen Übergang zu Version 7 zu gewährleisten:

1. Laden Sie ein privates AWS-SDK (siehethe section called “AWS SDK und Nutzerhandbuch für Content Moderation, Version 7 ”) herunter und verwenden Sie es, um die StartContentModeration API aufzurufen.
2. Sehen Sie sich die aktualisierte Liste der Labels und Konfidenzwerte an, die in der API-Antwort oder in der Konsole zurückgegeben wurden. Passen Sie die Nachverarbeitungslogik Ihrer Anwendung bei Bedarf entsprechend an.
3. Ihr Konto bleibt bis zum 13. Mai 2024 auf Version 6.1. Wenn Sie Version 6.1 über den 13. Mai 2024 hinaus verwenden möchten, wenden Sie sich bis zum 30. April 2024 an das AWS-Supportteam, um eine Verlängerung zu beantragen. Wir können Ihr Konto so verlängern, dass es bis zum 10. Juni 2024 weiterhin Version 6.1 verwendet. Wenn wir bis zum 30. April 2024 nichts von Ihnen hören, wird Ihr Konto ab dem 13. Mai 2024 automatisch auf Version 7.0 migriert.

AWS SDK und Nutzerhandbuch für Content Moderation, Version 7

Laden Sie das SDK herunter, das der von Ihnen ausgewählten Entwicklungssprache entspricht, und lesen Sie das entsprechende Benutzerhandbuch.

Link zum SDK	Installations- und Bedienungsanleitung
Java-1.X	Anleitung - Java 1.pdf
Java-2.X	Anleitung - Java 2.pdf
JavaScript v2	Leitfaden - JavaScript v2.pdf
JavaScript v3	Leitfaden - JavaScript v3.pdf
Python	Leitfaden — Python & AWS CLI.pdf
Ruby	Leitfaden - RubyV3.pdf
go_v1	Anleitung - GO V1.pdf
go_v2	Anleitung - GO V2.pdf
DotNet	Leitfaden - .NET.pdf
php	Anleitung - PHP.pdf

Labelzuordnungen für die Versionen 6.1 bis 7

In Version 7 der Inhaltsmoderation wurden neue Labelkategorien hinzugefügt und bereits bestehende Labelnamen geändert. Beziehen Sie sich the section called “ Labelkategorien ” bei der Entscheidung, wie 6.1-Labels 7 Labels zugeordnet werden sollen, auf die Taxonomie-Tabelle unter.

Im folgenden Abschnitt finden Sie einige Beispiele für Label-Zuordnungen. Wir empfehlen Ihnen, diese Zuordnungen und die Labeldefinitionen zu überprüfen, bevor Sie die erforderlichen Aktualisierungen auf der Grundlage der Nachverarbeitungslogik Ihrer Anwendung vornehmen.

L1-Zuordnungsschema

Wenn Sie eine Nachverarbeitungslogik verwenden, die nur nach der Kategorie der obersten Ebene (L1) filtert (z. B.`Explicit Nudity`, `Violence` usw.)`Suggestive`, finden Sie Informationen zur Aktualisierung Ihres Codes in der folgenden Tabelle.

V6.1 L1	V7 L1
Explizite Nacktheit	Explizit
Anzüglich	Unexplizite Nacktheit intimer Bereiche und Küssen
	Bademode oder Unterwäsche
Gewalt	Gewalt
Optisch störend	Optisch störend
Unhöfliche Gesten	Unhöfliche Gesten
Drogen	Drogen und Tabak
Tabak	Drogen und Tabak
Alkohol	Alkohol
Glücksspiel	Glücksspiel
Hasssymbole	Hasssymbole

L2-Zuordnungsschema

Wenn Sie eine Nachverarbeitungslogik verwenden, die sowohl nach L1- als auch nach L2-Kategorien (z. B. `Explicit Nudity / Nudity, Suggestive / Female Swimwear Or Underwear Violence / Weapon Violence` usw.) filtert, können Sie Ihren Code anhand der folgenden Tabelle aktualisieren.

V6.1 L1	V6.1 L2	V7 L1	V7 L2	V7 L3	V 7 ContentTypes

Explizite Nacktheit	Nacktheit	Explizit	Explizite Nacktheit	Freiliegende weibliche Brustwarze	
				Freiliegendes Gesäß oder Anus	
	Explizite männliche Nacktheit	Explizit	Explizite Nacktheit	Exponierte männliche Genitalien	
	Explizite weibliche Nacktheit	Explizit	Explizite Nacktheit	Exponiert e weibliche Genitalien	
	Sexuelle Aktivität	Explizit	Explizite sexuelle Aktivität		
	Illustrierte explizite Nacktheit	Explizit	Explizite Nacktheit		Zuordnung zu „Animiert “ und „Illustri ert“
	Illustrierte explizite Nacktheit	Explizit	Explizite sexuelle Aktivität		Zuordnung zu „Animiert “ und „Illustri ert“
	Erwachsen enspielzeug	Explizit	Sexspielzeug		
Anzüglich	Frauen in Badebekle idung oder Unterwäsche	Bademode oder Unterwäsche	Weibliche Bademode oder Unterwäsche		

	Männer in Badebekleidung oder Unterwäsche	Bademode oder Unterwäsche	Männliche Bademode oder Unterwäsche	
	Teilweise Nacktheit	Unexplizite Nacktheit intimer Bereiche und Küssen	Nicht explizite Nacktheit	Implizite Nacktheit
	Mann mit nackter Brust	Unexplizite Nacktheit intimer Bereiche und Küssen	Nicht explizite Nacktheit	Freiliegende männliche Brustwarze
	Freizügige Kleidung	Unexplizite Nacktheit intimer Bereiche und Küssen	Nicht explizite Nacktheit	
		Nicht explizite Nacktheit intimer Bereiche und Küssen	Verstopfte intime Bereiche	
	Sexuelle Situationen	Unexplizite Nacktheit intimer Bereiche und Küssen	Auf die Lippen küssen	
Gewalt	Grafische Gewalt oder Blut	Gewalt	Grafische Gewalt	Blut und Blut

	Körperliche Gewalt	Gewalt	Graphische Gewalt	Körperliche Gewalt
	Waffengewalt	Gewalt	Graphische Gewalt	Waffengewalt
	Waffen	Gewalt	Waffen	
	Selbstverletzung	Gewalt	Graphische Gewalt	Selbstverletzung
Optisch störend	Ausgemergelte Körper	Optisch störend	Tod und Abmagerung	Ausgemergelte Körper
	Leichen	Optisch störend	Tod und Abmagerung	Leichen
	Hinrichtung	Optisch störend	Tod und Abmagerung	Leichen
	Flugzeugabsturz	Optisch störend	Stürzt ab	Flugzeugabsturz
	Explosionen und Sprengungen	Gewalt	Grafische Gewalt	Explosionen und Explosionen
Unhöfliche Gesten	Mittelfinger	Unhöfliche Gesten	Mittelfinger	
Drogen	Pharmazeutische Produkte	Drogen und Tabak	Produkte	
	Drogenkonsum	Drogen und Tabak	Drogen- und Tabakzubehör und -konsum	

	Pillen	Drogen und Tabak	Produkte	Pillen
	Drogenutensilien	Drogen und Tabak	Drogen- und Tabakzubehör und -konsum	
Tabak	Tabakprodukte	Drogen und Tabak	Produkte	
	Rauchen	Drogen und Tabak	Drogen- und Tabakzubehör und -konsum	Rauchen
Alkohol	Alkohol trinken	Alkohol	Alkoholkonsum	Alkohol trinken
	Alkoholische Getränke	Alkohol	Alkoholische Getränke	
Glücksspiel	Glücksspiel	Glücksspiel		
Hasssymbole	Nazipartei	Hasssymbole	Nazipartei	
	Weiße Vorherrschaft	Hasssymbole	Weiße Vorherrschaft	
	Extremistisch	Hasssymbole	Extremistisch	

Erkennen unangemessener Bilder

Mit diesem DetectModerationLabelsVorgang können Sie feststellen, ob ein Bild unangemessenen oder anstößigen Inhalt enthält. Eine Liste der Moderationsbezeichnungen in Amazon Rekognition finden Sie unter Verwenden der Bild- und Videomoderation. APIs

Erkennen von unangemessenen Inhalten in einem Bild

Das Bild muss entweder im PNG- oder JPG-Format vorliegen. Sie können das Eingabebild als Array von Bild-Bytes (base64-kodierte Bild-Bytes) bereitstellen oder ein Amazon-S3-Objekt festlegen. In diesen Prozeduren laden Sie ein Bild (JPEG oder PNG) in Ihren S3-Bucket hoch.

Um diese Verfahren ausführen zu können, müssen Sie das AWS CLI oder das entsprechende SDK installiert haben. AWS Weitere Informationen finden Sie unter Erste Schritte mit Amazon Rekognition. Das AWS-Konto, das Sie verwenden, muss über die Zugriffsberechtigungen für die Amazon-Rekognition-API verfügen. Weitere Informationen finden Sie unter Von Amazon Rekognition definierte Aktionen.

Um Moderationslabels in einem Bild zu erkennen (SDK)

1. Wenn Sie dies noch nicht getan haben:

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`- und `AmazonS3ReadOnlyAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Laden Sie ein Bild in Ihren S3-Bucket hoch.

   Weitere Anleitungen finden Sie unter Upload eines Objekts in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service.

3. Verwenden Sie die folgenden Beispiele zum Aufrufen der `DetectModerationLabels`-Operation.

   Java

   In diesem Beispiel werden die Namen der erkannten Labels unangemessener Inhalte, Wahrscheinlichkeitsgrade und das übergeordnete Label für erkannte Moderationslabels ausgegeben.

   Ersetzen Sie die Werte von `amzn-s3-demo-bucket` und `photo` durch den S3-Bucket-Namen und den Bilddateinamen, die Sie in Schritt 2 verwendet haben.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
//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

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `detect-moderation-labels` CLI-Vorgang an.

Ersetzen Sie `amzn-s3-demo-bucket` und `input.jpg` mit dem S3-Bucket-Namen und dem Bilddateinamen, die Sie in Schritt 2 verwendet haben. Ersetzen Sie den Wert von `profile_name` mit dem Namen Ihres Entwicklerprofils. Um einen Adapter zu verwenden, geben Sie den ARN der Projektversion für den `project-version`-Parameter an.

```
aws rekognition detect-moderation-labels --image "{S3Object:{Bucket:<amzn-s3-
demo-bucket>,Name:<image-name>}}" \
--profile profile-name \
--project-version "ARN"
```

Wenn Sie auf einem Windows-Gerät auf die CLI zugreifen, verwenden Sie doppelte Anführungszeichen anstelle von einfachen Anführungszeichen und maskieren Sie die inneren doppelten Anführungszeichen durch einen Backslash (d. h. \), um eventuell auftretende Parserfehler zu beheben. Sehen Sie sich zum Beispiel Folgendes an:

```
aws rekognition detect-moderation-labels --image "{\"S3Object\":{\"Bucket\":
\"amzn-s3-demo-bucket\",\"Name\":\"image-name\"}}" \
--profile profile-name
```

Python

In diesem Beispiel werden die Namen der erkannten unangemessenen oder anstößigen Inhaltslabels, die Vertrauensstufen und das übergeordnete Label für erkannte unangemessene Inhaltslabels ausgegeben.

Ersetzen Sie in der `main`-Funktion die Werte von `amzn-s3-demo-bucket` und `photo` durch den S3-Bucket-Namen und den Bilddateinamen, die Sie in Schritt 2 verwendet haben. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

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

import boto3

def moderate_image(photo, bucket):

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

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

    print('Detected labels for ' + photo)
    for label in response['ModerationLabels']:
        print (label['Name'] + ' : ' + str(label['Confidence']))
        print (label['ParentName'])
    return len(response['ModerationLabels'])

def main():

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

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

.NET

In diesem Beispiel werden die Namen der erkannten Labels unangemessener oder anstößiger Inhalte, Wahrscheinlichkeitsgrade und das übergeordnete Label für erkannte Moderationslabels ausgegeben.

Ersetzen Sie die Werte von `amzn-s3-demo-bucket` und `photo` durch den S3-Bucket-Namen und den Bilddateinamen, die Sie in Schritt 2 verwendet haben.

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

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

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

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        DetectModerationLabelsRequest detectModerationLabelsRequest = new
 DetectModerationLabelsRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket
                },
            },
            MinConfidence = 60F
        };

        try
```

```
        {
            DetectModerationLabelsResponse detectModerationLabelsResponse =
 rekognitionClient.DetectModerationLabels(detectModerationLabelsRequest);
            Console.WriteLine("Detected labels for " + photo);
            foreach (ModerationLabel label in
 detectModerationLabelsResponse.ModerationLabels)
                Console.WriteLine("Label: {0}\n Confidence: {1}\n Parent: {2}",
                    label.Name, label.Confidence, label.ParentName);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```

DetectModerationLabels Operationsanforderung

Die Eingabe in `DetectModerationLabels` ist ein Bild. In dieser Beispiel-JSON-Eingabe wird das Quellbild aus einem Amazon-S3-Bucket geladen. ist `MinConfidence` das minimale Vertrauen, das Amazon Rekognition Image in die Genauigkeit des erkannten Labels haben muss, damit es in der Antwort zurückgegeben wird.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "input.jpg"
        }
    },
    "MinConfidence": 60
}
```

DetectModerationLabels Antwort auf die Operation

`DetectModerationLabels` kann Eingabebilder aus einem S3-Bucket abrufen oder als Image-Bytes zur Verfügung stellen. Das folgende Beispiel zeigt die Antwort auf eine Anforderung an `DetectModerationLabels`.

In der folgenden Beispiel-JSON-Antwort sollte Folgendes beachtet werden:

- Informationen zur Erkennung unangemessener Bilder: Das Beispiel zeigt eine Liste mit Labels für unangemessene oder anstößige Inhalte, die im Bild gefunden wurden. Die Liste umfasst das auf dem Bild erkannte Label der Oberkategorie sowie jedes Label von Kategorien zweiter Stufe.

  Label – Jedes Label verfügt über einen Namen, einen geschätzten Zuverlässigkeitswert von Amazon Rekognition in Bezug auf die Exaktheit des Labels sowie den Namen des übergeordneten Labels. Der übergeordnete Name für ein übergeordnetes Label ist "".

  Zuverlässigkeitswert für das Label – Jedes Label verfügt über einen Zuverlässigkeitswert zwischen 0 und 100. Dieser gibt an, zu wie viel Prozent Amazon Rekognition sicher ist, dass das Label korrekt ist. Sie geben den erforderlichen Zuverlässigkeitswert für ein Label an, das bei der Erstellung der Antwort durch die API-Operation gelten soll.


```
{
    "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"
        }
    ]
```

```
}
```

Erkennung unangemessener gespeicherter Videos

Die Erkennung unangemessener oder anstößiger Inhalte in gespeicherten Videos mit Amazon Rekognition Video ist eine asynchrone Operation. Rufen Sie an, um unangemessene oder anstößige Inhalte zu erkennen StartContentModeration. Das Amazon-Simple-Notification-Service-Thema, zu dem Amazon Rekognition Video die Ergebnisse der Objekterkennung und den Abschlussstatus einer Videoanalyse-Operation veröffentlicht. Wenn die Videoanalyse erfolgreich ist, rufen Sie GetContentModeration auf, um die Analyseergebnisse abzurufen. Weitere Informationen zum Starten der Videoanalyse und zum Abrufen der Ergebnisse finden Sie unter Amazon-Rekognition-Video-Operationen aufrufen. Eine Liste der Moderationsbezeichnungen in Amazon Rekognition finden Sie unter Verwenden der Bild- und Videomoderation. APIs

Dieses Verfahren erweitert den Code in Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK), der eine Amazon-Simple-Queue-Service-Warteschlange verwendet, um den Abschlussstatus einer Videoanalyseanforderung zu erhalten.

So erkennen Sie unangemessene oder anstößige Inhalte in einem Video, das in einem Amazon-S3-Bucket (SDK) gespeichert ist

1. Führen Sie Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) aus.
2. Fügen Sie den folgenden Code in der Klasse `VideoDetect` ein, die Sie in Schritt 1 erstellt haben.

   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);

        }
```

Ersetzen Sie in der Funktion `main` die folgenden Zeilen:

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

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

mit:

```
        StartUnsafeContentDetection(amzn-s3-demo-bucket, video);

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

Java V2

Dieser Code stammt aus dem AWS Documentation SDK Examples Repository. GitHub Das vollständige Beispiel finden Sie hier.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import
 software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

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

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
```

```
                    roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

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

        startModerationDetection(rekClient, channel, bucket, video);
        getModResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startModerationDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {

        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

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

```
            StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
                    .jobTag("Moderation")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartContentModerationResponse startModDetectionResult = rekClient
                    .startContentModeration(modDetectionRequest);
            startJobId = startModDetectionResult.jobId();

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

    public static void getModResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetContentModerationResponse modDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (modDetectionResponse != null)
                    paginationToken = modDetectionResponse.nextToken();

                GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    modDetectionResponse =
rekClient.getContentModeration(modRequest);
                    status = modDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
```

```
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
 null.
                VideoMetadata videoMetaData =
 modDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
 videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<ContentModerationDetection> mods =
 modDetectionResponse.moderationLabels();
                for (ContentModerationDetection mod : mods) {
                    long seconds = mod.timestamp() / 1000;
                    System.out.print("Mod label: " + seconds + " ");
                    System.out.println(mod.moderationLabel().toString());
                    System.out.println();
                }

            } while (modDetectionResponse != null &&
 modDetectionResponse.nextToken() != null);

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

Python

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

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

    # ============== Unsafe content ===============
    def StartUnsafeContent(self):
        response=self.rek.start_content_moderation(Video={'S3Object': {'Bucket':
 self.bucket, 'Name': self.video}},
            NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
 self.snsTopicArn})

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

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

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

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

            for contentModerationDetection in response['ModerationLabels']:
                print('Label: ' +
                    str(contentModerationDetection['ModerationLabel']['Name']))
                print('Confidence: ' +
                    str(contentModerationDetection['ModerationLabel']
['Confidence']))
                print('Parent category: ' +
                    str(contentModerationDetection['ModerationLabel']
['ParentName']))
                print('Timestamp: ' +
 str(contentModerationDetection['Timestamp']))
                print()
```

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

Ersetzen Sie in der Funktion `main` die folgenden Zeilen:

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

mit:

```
    analyzer.StartUnsafeContent()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetUnsafeContentResults()
```

> **Note**
>
> Wenn Sie zusätzlich zu Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) bereits ein anderes Videobeispiel ausgeführt haben, ist der zu ersetzende Code möglicherweise anders.

3. Führen Sie den Code aus. Eine Liste der im Video erkannten unangemessenen Inhaltslabels wird angezeigt.

GetContentModeration Antwort auf den Vorgang

Die `GetContentModeration`-Antwort ist ein Array, `ModerationLabels`, von ContentModerationDetection-Objekten. Das Array enthält ein Element für jede Erkennung eines Labels unangemessener Inhalte. ModerationLabelEnthält innerhalb eines `ContentModerationDetectionObject` Objekts Informationen zu einem erkannten Objekt mit unangemessenem oder anstößigem Inhalt. `Timestamp`ist der Zeitpunkt in Millisekunden ab dem Start des Videos, zu dem das Label erkannt wurde. Die Labels sind hierarchisch in der Weise organisiert, in der die Labels von der Analyse von Bildern mit unangemessenen Inhalten erkannt wurden. Weitere Informationen finden Sie unter Inhalte moderieren.

Im Folgenden finden Sie ein Beispiel für eine Antwort von `GetContentModeration`, sortiert nach `NAME` und aggregiert nach `TIMESTAMPS`.


```
{
    "JobStatus": "SUCCEEDED",
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 54100,
        "Format": "QuickTime / MOV",
        "FrameRate": 30.0,
        "FrameHeight": 462,
        "FrameWidth": 884,
        "ColorRange": "LIMITED"
    },
    "ModerationLabels": [
        {
            "Timestamp": 36000,
            "ModerationLabel": {
                "Confidence": 52.451576232910156,
                "Name": "Alcohol",
                "ParentName": "",
                "TaxonomyLevel": 1
            },
            "ContentTypes": [
                {
                    "Confidence": 99.9999008178711,
                    "Name": "Animated"
                }
            ]
        },
        {
            "Timestamp": 36000,
            "ModerationLabel": {
                "Confidence": 52.451576232910156,
                "Name": "Alcoholic Beverages",
                "ParentName": "Alcohol",
                "TaxonomyLevel": 2
            },
            "ContentTypes": [
                {
                    "Confidence": 99.9999008178711,
                    "Name": "Animated"
                }
```

```
            ]
        }
    ],
    "ModerationModelVersion": "7.0",
    "JobId": "a1b2c3d4...",
    "Video": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "video-name.mp4"
        }
    },
    "GetRequestMetadata": {
        "SortBy": "TIMESTAMP",
        "AggregateBy": "TIMESTAMPS"
    }
}
```

Im Folgenden finden Sie ein Beispiel für eine Antwort von `GetContentModeration`, sortiert nach `NAME` und aggregiert nach `SEGMENTS`.

```
{
    "JobStatus": "SUCCEEDED",
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 54100,
        "Format": "QuickTime / MOV",
        "FrameRate": 30.0,
        "FrameHeight": 462,
        "FrameWidth": 884,
        "ColorRange": "LIMITED"
    },
    "ModerationLabels": [
        {
            "Timestamp": 0,
            "ModerationLabel": {
                "Confidence": 0.0003000000142492354,
                "Name": "Alcohol Use",
                "ParentName": "Alcohol",
                "TaxonomyLevel": 2
            },
            "StartTimestampMillis": 0,
            "EndTimestampMillis": 29520,
            "DurationMillis": 29520,
```

```
            "ContentTypes": [
                {
                    "Confidence": 99.9999008178711,
                    "Name": "Illustrated"
                },
                {
                    "Confidence": 99.9999008178711,
                    "Name": "Animated"
                }
            ]
        }
    ],
    "ModerationModelVersion": "7.0",
    "JobId": "a1b2c3d4...",
    "Video": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "video-name.mp4"
        }
    },
    "GetRequestMetadata": {
        "SortBy": "TIMESTAMP",
        "AggregateBy": "SEGMENTS"
    }
}
```

Verbesserung der Genauigkeit mit benutzerdefinierter Moderation

Mit der DetectModerationLabelsAPI von Amazon Rekognition können Sie Inhalte erkennen, die unangemessen, unerwünscht oder anstößig sind. Mit der Funktion Rekognition Custom Moderation können Sie die Genauigkeit von mithilfe DetectModerationLabelsvon Adaptern verbessern. Adapter sind modulare Komponenten, die zu einem bestehenden Deep-Learning-Modell von Rekognition hinzugefügt werden können, wodurch dessen Funktionen für die Aufgaben, für die es trainiert wurde, erweitert werden. Indem Sie einen Adapter erstellen und ihn für den DetectModerationLabelsBetrieb bereitstellen, können Sie eine höhere Genauigkeit bei den Aufgaben der Inhaltsmoderation erreichen, die sich auf Ihren speziellen Anwendungsfall beziehen.

Wenn Sie das Inhaltsmoderationsmodell von Rekognition für bestimmte Moderationslabels anpassen, müssen Sie ein Projekt erstellen und einen Adapter anhand einer Reihe von Bildern trainieren, die Sie bereitstellen. Anschließend können Sie die Leistung des Adapters iterativ überprüfen und

den Adapter so anpassen, dass er den gewünschten Genauigkeitsgrad erreicht. Projekte werden verwendet, um die verschiedenen Versionen von Adaptern zu enthalten.

Sie können die Rekognition-Konsole verwenden, um Projekte und Adapter zu erstellen. Alternativ können Sie ein AWS SDK und das dazugehörige verwenden, um ein Projekt APIs zu erstellen, einen Adapter zu trainieren und Ihre Adapter zu verwalten.

Adapter erstellen und verwenden

Adapter sind modulare Komponenten, die zum bestehenden Deep-Learning-Modell von Rekognition hinzugefügt werden können, wodurch dessen Funktionen für die Aufgaben, für die es trainiert wurde, erweitert werden. Durch das Training eines Deep-Learning-Modells mit Adaptern können Sie eine höhere Genauigkeit bei Bildanalyseaufgaben erreichen, die sich auf Ihren speziellen Anwendungsfall beziehen.

Um einen Adapter zu erstellen und zu verwenden, müssen Sie Rekognition Trainings- und Testdaten zur Verfügung stellen. Sie können dies auf zwei Arten erreichen:

- Massenanalyse und -überprüfung: Sie können einen Trainingsdatensatz erstellen, indem Sie Bilder massenweise analysieren, die Rekognition analysiert und ihnen Label zuweist. Anschließend können Sie die generierten Anmerkungen für Ihre Bilder überprüfen und die Vorhersagen überprüfen oder korrigieren. Weitere Informationen zur Funktionsweise der Massenanalyse von Bildern finden Sie unter Massenanalyse.
- Manuelles Kommentieren: Bei diesem Ansatz erstellen Sie Ihre Trainingsdaten, indem Sie Bilder hochladen und kommentieren. Sie erstellen Ihre Testdaten, indem Sie entweder Bilder hochladen und mit Anmerkungen versehen oder sie automatisch aufteilen.

Wählen Sie eines der folgenden Themen aus, um mehr zu erfahren:

Themen

- Massenanalyse und -überprüfung
- Manuelle Kommentierung

Massenanalyse und -überprüfung

Bei diesem Ansatz laden Sie eine große Anzahl von Bildern hoch, die Sie als Trainingsdaten verwenden möchten, und verwenden dann Rekognition, um Vorhersagen für diese Bilder zu

erhalten, wodurch ihnen automatisch Label zugewiesen werden. Sie können diese Vorhersagen als Ausgangspunkt für Ihren Adapter verwenden. Sie können die Genauigkeit der Vorhersagen überprüfen und dann den Adapter auf der Grundlage der verifizierten Vorhersagen trainieren. Dies kann mit der AWS Konsole geschehen.

Massenanalyse und benutzerdefinierte Moderation

Laden Sie Bilder für die Massenanalyse hoch

Um einen Trainingsdatensatz für Ihren Adapter zu erstellen, laden Sie Bilder in großen Mengen hoch, damit Rekognition Label vorhersagen kann. Um optimale Ergebnisse zu erzielen, sollten Sie für das Training so viele Bilder wie möglich bis zu einer Obergrenze von 10.000 bereitstellen und sicherstellen, dass die Bilder für alle Aspekte Ihres Anwendungsfalls repräsentativ sind.

Wenn Sie die AWS Konsole verwenden, können Sie Bilder direkt von Ihrem Computer hochladen oder einen Amazon Simple Storage Service-Bucket bereitstellen, in dem Ihre Bilder gespeichert werden. Wenn Sie Rekognition jedoch APIs mit einem SDK verwenden, müssen Sie eine Manifestdatei bereitstellen, die auf Bilder verweist, die in einem Amazon Simple Storage Service-Bucket gespeichert sind. Weitere Informationen finden Sie unter Massenanalyse.

Überprüfen der Vorhersagen

Sobald Sie Ihre Bilder auf die Rekognition-Konsole hochgeladen haben, generiert Rekognition Label für sie. Anschließend können Sie die Vorhersagen anhand einer der folgenden Kategorien verifizieren: richtig positiv, falsch positiv, wahr negativ, falsch negativ. Nachdem Sie die Vorhersagen verifiziert haben, können Sie einen Adapter anhand Ihres Feedbacks trainieren.

Trainieren des Adapters

Sobald Sie die Überprüfung der durch die Massenanalyse zurückgegebenen Vorhersagen abgeschlossen haben, können Sie den Trainingsprozess für Ihren Adapter einleiten.

Holen Sie sich das AdapterId

Sobald der Adapter trainiert wurde, können Sie die eindeutige ID für Ihren Adapter abrufen, die er für die Bildanalyse von Rekognition verwenden kann. APIs

Rufen Sie die API-Operation auf

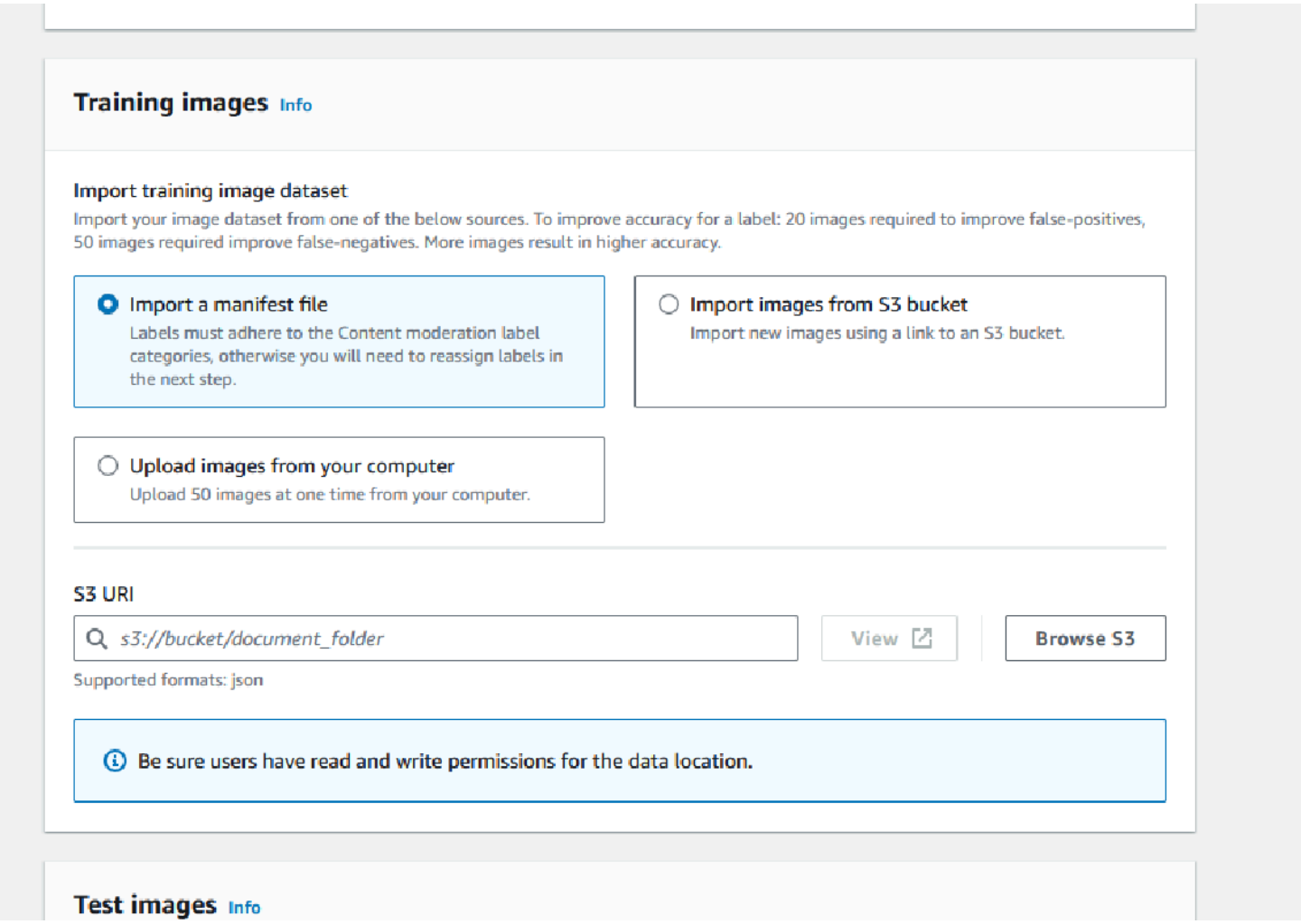
Um Ihren benutzerdefinierten Adapter anzuwenden, geben Sie dessen ID an, wenn Sie einen der Bildanalyseprogramme aufrufen APIs , der Adapter unterstützt. Dies verbessert die Genauigkeit der Vorhersagen für Ihre Bilder.

Manuelle Kommentierung

Bei diesem Ansatz erstellen Sie Ihre Trainingsdaten, indem Sie Bilder hochladen und manuell kommentieren. Sie erstellen Ihre Testdaten, indem Sie entweder Testbilder hochladen und mit Anmerkungen versehen oder sie automatisch aufteilen, sodass Rekognition automatisch einen Teil Ihrer Trainingsdaten als Testbilder verwendet.

Bilder hochladen und kommentieren

Um den Adapter zu trainieren, müssen Sie eine Reihe von Beispielbildern hochladen, die für Ihren Anwendungsfall repräsentativ sind. Um optimale Ergebnisse zu erzielen, sollten Sie für das Training so viele Bilder wie möglich bis zu einer Obergrenze von 10.000 bereitstellen und sicherstellen, dass die Bilder für alle Aspekte Ihres Anwendungsfalls repräsentativ sind.



Wenn Sie die AWS Konsole verwenden, können Sie Bilder direkt von Ihrem Computer hochladen, eine Manifestdatei bereitstellen oder einen Amazon S3 S3-Bucket bereitstellen, in dem Ihre Bilder gespeichert werden.

Wenn Sie Rekognition jedoch APIs mit einem SDK verwenden, müssen Sie eine Manifestdatei bereitstellen, die auf Bilder verweist, die in einem Amazon S3 S3-Bucket gespeichert sind.

Sie können die Annotationsoberfläche der Rekognition-Konsole verwenden, um Ihre Bilder mit Anmerkungen zu versehen. Kommentieren Sie Ihre Bilder, indem Sie sie mit Labels versehen. Auf diese Weise erhalten Sie eine „Grundwahrheit“ für das Training. Sie müssen auch Trainings- und Testsets festlegen oder das auto-split-Feature verwenden, bevor Sie einen Adapter trainieren können. Wenn Sie mit der Benennung Ihrer Datensätze und dem Hinzufügen von Anmerkungen zu Ihren Bildern fertig sind, können Sie einen Adapter erstellen, der auf den kommentierten Bildern in Ihrem Testsatz basiert. Anschließend können Sie die Leistung Ihres Adapters bewerten.

Erstellen eines Testsatzes

Sie müssen einen kommentierten Testsatz bereitstellen oder das auto-split-Feature verwenden. Das Trainingsset wird verwendet, um den Adapter tatsächlich zu trainieren. Der Adapter lernt die Muster, die in diesen kommentierten Bildern enthalten sind. Der Testsatz wird verwendet, um die Leistung des Modells zu bewerten, bevor der Adapter fertiggestellt wird.

Trainieren des Adapters

Sobald Sie mit dem Annotieren der Trainingsdaten fertig sind oder eine Manifestdatei bereitgestellt haben, können Sie den Trainingsprozess für Ihren Adapter starten.

Adapter-ID abrufen

Sobald der Adapter trainiert wurde, können Sie die eindeutige ID für Ihren Adapter abrufen, die er für die Bildanalyse von Rekognition verwenden kann. APIs

Rufen Sie die API-Operation auf

Um Ihren benutzerdefinierten Adapter anzuwenden, geben Sie dessen ID an, wenn Sie einen der Bildanalyseprogramme aufrufen APIs , der Adapter unterstützt. Dies verbessert die Genauigkeit der Vorhersagen für Ihre Bilder.

Vorbereiten Ihrer Datensätze

Um einen Adapter zu erstellen, müssen Sie Rekognition zwei Datensätze zur Verfügung stellen, einen Trainingsdatensatz und einen Testdatensatz. Jeder Datensatz besteht aus zwei Elementen: Bildern und Anmerkungen/Labels. In den folgenden Abschnitten wird erklärt, wofür Labels und Bilder verwendet werden und wie sie zusammengefügt werden, um Datensätze zu erstellen.

Bilder

Sie müssen einen Adapter anhand repräsentativer Stichproben Ihrer Bilder trainieren. Wenn Sie Bilder für das Training auswählen, versuchen Sie, mindestens einige Bilder einzufügen, die die erwartete Reaktion für jedes der Labels zeigen, auf die Sie mit Ihrem Adapter abzielen.

Um einen Trainingsdatensatz zu erstellen, müssen Sie einen der folgenden beiden Bildtypen angeben:

- Bilder mit falsch positiven Vorhersagen. Zum Beispiel, wenn ein Basismodell voraussagt, dass in einem Bild Alkohol enthalten ist, dies aber nicht der Fall ist.
- Bilder mit falsch negativen Vorhersagen. Zum Beispiel, wenn ein Basismodell voraussagt, dass in einem Bild kein Alkohol enthalten ist, der Alkohol aber vorhanden ist.

Um einen ausgewogenen Datensatz zu erstellen, wird empfohlen, einen der beiden folgenden Bildtypen bereitzustellen:

- Bilder mit wahren positiven Vorhersagen. Zum Beispiel, wenn ein Basismodell korrekt vorhersagt, dass ein Bild Alkohol enthält. Es wird empfohlen, diese Bilder bereitzustellen, wenn Sie falsch positive Bilder angeben.
- Bilder mit wahren negativen Vorhersagen. Zum Beispiel, wenn ein Basismodell korrekt voraussagt, dass ein Bild keinen Alkohol enthält. Es wird empfohlen, diese Bilder bereitzustellen, wenn Sie falsch negative Bilder bereitstellen.

Labels

Ein Label bezieht sich auf Folgendes: Objekte, Ereignisse, Konzepte oder Aktivitäten. Bei der Inhaltsmoderation ist ein Label ein Beispiel für Inhalte, die unangemessen, unerwünscht oder anstößig sind.

Im Zusammenhang mit der Erstellung eines Adapters durch Training des Basismodells von Rekognition wird ein Label, das einem Bild zugewiesen wird, als Anmerkung bezeichnet. Wenn Sie einen Adapter mit der Rekognition-Konsole trainieren, verwenden Sie die Konsole, um Anmerkungen zu Ihren Bildern hinzuzufügen, indem Sie ein Label auswählen und dann die Bilder markieren, die dem Label entsprechen. Durch diesen Prozess lernt das Modell, Elemente Ihrer Bilder anhand des zugewiesenen Labels zu identifizieren. Durch diesen Verknüpfungsprozess kann sich das Modell bei der Erstellung eines Adapters auf die relevantesten Inhalte konzentrieren, was zu einer verbesserten Genauigkeit bei der Bildanalyse führt.

Alternativ können Sie eine Manifestdatei bereitstellen, die Informationen zu Bildern und den dazugehörigen Anmerkungen enthält.

Trainieren und Testen von Datensätzen

Der Trainingsdatensatz ist die Grundlage für die Feinabstimmung des Modells und die Erstellung eines benutzerdefinierten Adapters. Sie müssen einen kommentierten Trainingsdatensatz bereitstellen, aus dem das Modell lernen kann. Das Modell lernt aus diesem Datensatz, um seine Leistung bei der Art der von Ihnen bereitgestellten Bilder zu verbessern.

Um die Genauigkeit zu verbessern, müssen Sie Ihren Trainingsdatensatz erstellen, indem Sie Bilder kommentieren/beschriften. Sie können dies auf zwei Arten erreichen:

- Manuelle Labelzuweisung: Sie können die Rekognition-Konsole verwenden, um einen Trainingsdatensatz zu erstellen, indem Sie die Bilder hochladen, die Ihr Datensatz enthalten soll, und diesen Bildern dann manuell Labels zuweisen.
- Manifestdatei: Sie können eine Manifestdatei verwenden, um Ihren Adapter zu trainieren. Die Manifestdatei enthält Informationen zu den Grundwahrheitsanmerkungen für Ihre Trainings- und Testbilder sowie den Speicherort Ihrer Trainingsbilder. Sie können die Manifestdatei bereitstellen, wenn Sie einen Adapter mit der Rekognition trainieren APIs oder wenn Sie die AWS Konsole verwenden.

Der Testdatensatz wird verwendet, um die Leistung des Adapters nach dem Training zu bewerten. Um eine zuverlässige Auswertung zu gewährleisten, wird der Testdatensatz unter Verwendung eines Abschnitts des ursprünglichen Trainingsdatensatzes erstellt, den das Modell noch nicht gesehen hat. Dieser Prozess stellt sicher, dass die Leistung des Adapters anhand neuer Daten bewertet wird, wodurch genaue Messungen und Metriken erstellt werden. Optimale Genauigkeitsverbesserungen finden Sie unter Bewährte Methoden für Trainingsadapter.

Adapter mit der AWS CLI verwalten und SDKs

Mit Rekognition können Sie mehrere Funktionen nutzen, die vortrainierte Computer-Vision-Modelle nutzen. Mit diesen Modellen können Sie Aufgaben wie die Erkennung von Labels und die Moderation von Inhalten ausführen. Sie können diese bestimmten Modelle auch mithilfe eines Adapters anpassen.

Sie können die Projekterstellung und das Projektmanagement von Rekognition APIs (wie und CreateProjectVersion) nutzen, um CreateProjectAdapter zu erstellen und zu trainieren. Auf den

folgenden Seiten wird beschrieben, wie Sie die API-Operationen verwenden, um Ihre Adapter mithilfe der AWS Konsole, des von Ihnen ausgewählten AWS SDK oder der AWS CLI zu erstellen, zu trainieren und zu verwalten.

Nachdem Sie einen Adapter trainiert haben, können Sie ihn verwenden, um Inferenz mit unterstützten Features auszuführen. Derzeit werden Adapter unterstützt, wenn das Feature zur Inhaltsmoderation verwendet wird.

Wenn Sie einen Adapter mit einem AWS SDK trainieren, müssen Sie Ihre Ground-Truth-Beschriftungen (Bildanmerkungen) in Form einer Manifestdatei bereitstellen. Alternativ können Sie die Rekognition-Konsole verwenden, um einen Adapter zu erstellen und zu trainieren.

> **Note**
>
> Adapter können nicht kopiert werden. Nur Projektversionen von Rekognition Custom Labels können kopiert werden.

Themen

- Status der Adapter
- Erstellen eines Projekts
- Projekte beschreiben
- Löschen eines Projekts
- Erstellen einer Projektversion
- Beschreibung einer Projektversion
- Löschen einer Projektversion

Status der Adapter

Der benutzerdefinierte Moderationsadapter (Projektversionen) kann sich in einem der folgenden Status befinden:

- TRAINING_IN_PROGRESS — Der Adapter wird gerade mit den Dateien trainiert, die Sie als Schulungsdokumente bereitgestellt haben.
- TRAINING_COMPLETED — Der Adapter hat die Schulung erfolgreich abgeschlossen und kann nun von Ihnen überprüft werden.

- TRAINING_FAILED — Der Adapter konnte sein Training aus irgendeinem Grund nicht abschließen. Informationen zur Ursache des Fehlers finden Sie in der Ausgabemanifestdatei und in der Zusammenfassung des Ausgabemanifests.
- LÖSCHEN — Der Adapter wird gerade gelöscht.
- VERALTET — Der Adapter wurde auf einer älteren Version des Basismodells der Inhaltsmoderation trainiert. Es befindet sich in einer Übergangszeit und läuft innerhalb von 60 bis 90 Tagen nach der Veröffentlichung der neuen Basismodellversion ab. Während der Übergangszeit können Sie den Adapter weiterhin für Inferenzen DetectModerationLabelsoder StartMediaAnalysisJobAPI-Operationen verwenden. Das Ablaufdatum Ihrer Adapter finden Sie in der Custom Moderation Console.
- ABGELAUFEN — Der Adapter wurde auf einer älteren Version des Basismodells der Inhaltsmoderation trainiert und kann nicht mehr verwendet werden, um benutzerdefinierte Ergebnisse mit den DetectModerationLabels oder StartMediaAnalysisJob API-Operationen zu erzielen. Wenn in einer Inferenzanforderung ein abgelaufener Adapter angegeben ist, wird er ignoriert und die Antwort wird stattdessen von der neuesten Version des Basismodells der benutzerdefinierten Moderation zurückgegeben.

Erstellen eines Projekts

Mit dieser CreateProjectOperation können Sie ein Projekt erstellen, das einen Adapter für die Labelerkennungsoperationen von Rekognition enthält. Ein Projekt ist eine Gruppe von Ressourcen, und im Fall von Etikettenerkennungsoperationen wie einem Projekt können Sie Adapter speichern DetectModerationLabels, mit denen Sie das Rekognition-Basismodell anpassen können. Beim Aufrufen CreateProject geben Sie dem Argument den Namen des Projekts, das Sie erstellen möchten. ProjectName

Um ein Projekt mit der AWS Konsole zu erstellen:

- Melden Sie sich bei der Rekognition-Konsole an
- Klicken Sie auf Benutzerdefinierte Moderation
- Wählen Sie Projekt erstellen aus
- Wählen Sie entweder Neues Projekt erstellen oder Zu einem vorhandenen Projekt hinzufügen
- Fügen Sie einen Projektnamen hinzu
- Fügen Sie einen Adapternamen hinzu
- Fügen Sie bei Bedarf eine Beschreibung hinzu

- Wählen Sie aus, wie Sie Ihre Trainingsbilder importieren möchten: Manifest-Datei, aus dem S3-Bucket oder von Ihrem Computer
- Wählen Sie aus, ob Sie Ihre Trainingsdaten automatisch splitten oder eine Manifestdatei importieren möchten
- Wählen Sie aus, ob das Projekt automatisch aktualisiert werden soll
- Klicken Sie auf Projekt erstellen

Um ein Projekt mit der AWS CLI und dem SDK zu erstellen:

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.
2. Verwenden Sie den folgenden Code, um ein Projekt zu erstellen:

CLI

```
# Request
# Creating Content Moderation Project
aws rekognition create-project \
    --project-name "project-name" \
    --feature CONTENT_MODERATION \
    --auto-update ENABLED
    --profile profile-name
```

Projekte beschreiben

Sie können die DescribeProjectsAPI verwenden, um Informationen zu Ihren Projekten abzurufen, einschließlich Informationen zu allen Adaptern, die einem Projekt zugeordnet sind.

Um Projekte mit der AWS CLI und dem SDK zu beschreiben:

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.
2. Verwenden Sie den folgenden Code, um ein Projekt zu beschreiben:

CLI

```
# Request
# Getting CONTENT_MODERATION project details
aws rekognition describe-projects \
    --features CONTENT_MODERATION
    --profile profile-name
```

Löschen eines Projekts

Sie können ein Projekt löschen, indem Sie die Rekognition-Konsole verwenden oder die DeleteProjectAPI aufrufen. Um ein Projekt zu löschen, müssen Sie zuerst jeden der zugehörigen Adapter löschen. Ein gelöschtes Projekt oder Modell kann nicht rückgängig gemacht werden.

Um ein Projekt mit der AWS Konsole zu löschen:

- Melden Sie sich bei der Rekognition-Konsole an.
- Klicken Sie auf Benutzerdefinierte Moderation.
- Sie müssen jeden Adapter löschen, der Ihrem Projekt zugeordnet ist, bevor das Projekt selbst gelöscht werden kann. Löschen Sie alle mit dem Projekt verknüpften Adapter, indem Sie den Adapter auswählen und dann Löschen wählen.
- Wählen Sie das Projekt aus und klicken Sie dann auf die Schaltfläche Löschen.

Um ein Projekt mit der AWS CLI und dem SDK zu löschen:

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.
2. Verwenden Sie den folgenden Code, um ein Projekt zu löschen:

CLI

```
aws rekognition delete-project
  --project-arn project_arn \
```

```
  --profile profile-name
```

Erstellen einer Projektversion

Sie können einen Adapter für die Bereitstellung trainieren, indem Sie den CreateProjectVersionVorgang verwenden. CreateProjectVersion erstellt zuerst eine neue Version eines Adapters, der einem Projekt zugeordnet ist, und beginnt dann mit dem Training des Adapters. Die Antwort von CreateProjectVersion ist ein Amazon Resource Name (ARN) für die Version des Modells. Es dauert eine Weile, bis das Training abgeschlossen ist. Sie können den aktuellen Status abrufen, indem Sie anrufen DescribeProjectVersions. Beim Training eines Modells verwendet Rekognition die mit dem Projekt verknüpften Trainings- und Testdatensätze. Sie erstellen Datensätze über die Konsole. Weitere Informationen finden Sie im Abschnitt über Datensätze.

Um eine Projektversion mit der Rekognition-Konsole zu erstellen:

- Melden Sie sich bei der AWS-Rekognition-Konsole an
- Klicken Sie auf „Benutzerdefinierte Moderation“
- Wählen Sie ein Projekt aus.
- Wählen Sie auf der Seite „Projektdetails“ die Option Adapter erstellen
- Geben Sie auf der Seite „Projekt erstellen“ die erforderlichen Details für Projektdetails, Trainingsbilder und Testbilder ein und wählen Sie dann Projekt erstellen aus.
- Fügen Sie auf der Seite „Bildern Label zuweisen“ Labels zu Ihren Bildern hinzu und wählen Sie, wenn Sie fertig sind, die Option Training starten

So erstellen Sie eine Projektversion mit der AWS CLI und dem SDK:

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.
2. Verwenden Sie den folgenden Code, um eine Projektversion zu erstellen:

CLI

```
# Request
```

```
aws rekognition create-project-version \
 --project-arn project-arn \
 --training-data '{Assets=[GroundTruthManifest={S3Object="amzn-s3-demo-source-
bucket",Name="manifest.json"}]}' \
 --output-config S3Bucket=amzn-s3-demo-destination-bucket,S3KeyPrefix=my-results \
 --feature-config "ContentModeration={ConfidenceThreshold=70}"
 --profile profile-name
```

Beschreibung einer Projektversion

Mithilfe der DescribeProjectVersionsOperation können Sie Adapter, die einem Projekt zugeordnet sind, auflisten und beschreiben. Sie können bis zu 10 Modellversionen in angeben ProjectVersionArns. Wenn Sie keinen Wert angeben, werden Beschreibungen für alle Modellversionen im Projekt zurückgegeben.

Um eine Projektversion mit der AWS CLI und dem SDK zu beschreiben:

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.
2. Verwenden Sie den folgenden Code, um eine Projektversion zu beschreiben:

CLI

```
aws rekognition describe-project-versions
  --project-arn project_arn \
  --version-names [versions]
```

Löschen einer Projektversion

Mit diesem Vorgang können Sie einen Rekognition-Adapter löschen, der DeleteProjectVersioneinem Projekt zugeordnet ist. Sie können einen Adapter nicht löschen, wenn er ausgeführt oder trainiert wird. Um den Status eines Adapters zu überprüfen, rufen Sie den DescribeProjectVersions Vorgang auf und überprüfen Sie das vom Adapter zurückgegebene Statusfeld. Um einen laufenden Adapter zu beenden, rufen Sie ihn auf StopProjectVersion. Wenn das Modell trainiert wird, warten Sie, bis es das

Training abgeschlossen hat, um es zu löschen. Sie müssen jeden Adapter löschen, der Ihrem Projekt zugeordnet ist, bevor das Projekt selbst gelöscht werden kann.

Um eine Projektversion mit der Rekognition-Konsole zu löschen:

- Melden Sie sich bei der Rekognition-Konsole an
- Klicken Sie auf „Benutzerdefinierte Moderation“
- Auf der Registerkarte „Projekte“ können Sie alle Ihre Projekte und die zugehörigen Adapter sehen. Wählen Sie einen Adapter aus und klicken Sie dann auf Löschen.

Um eine Projektversion mit der AWS CLI und dem SDK zu löschen:

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.
2. Verwenden Sie den folgenden Code, um eine Projektversion zu löschen:

CLI

```
# Request
aws rekognition delete-project-version
  --project-version-arn model_arn \
  --profile profile-name
```

Tutorial zum benutzerdefinierten Moderationsadapter

In diesem Tutorial erfahren Sie, wie Sie Adapter mit der Rekognition-Konsole erstellen, trainieren, evaluieren, verwenden und verwalten. Informationen zum Erstellen, Verwenden und Verwalten von Adaptern mit dem AWS SDK finden Sie unterAdapter mit der AWS CLI verwalten und SDKs.

Mit Adaptern können Sie die Genauigkeit der API-Operationen von Rekognition verbessern und das Verhalten des Modells an Ihre eigenen Bedürfnisse und Anwendungsfälle anpassen. Nachdem Sie mit diesem Tutorial einen Adapter erstellt haben, können Sie ihn verwenden, um Ihre eigenen Bilder mit Operationen wie DetectModerationLabelszu analysieren und den Adapter für weitere future Verbesserungen neu zu trainieren.

In diesem Tutorial lernen Sie Folgendes:

- Erstellen eines Projekts mit der Rekognition-Konsole
- Kommentieren Ihrer Trainingsdaten
- Trainieren Ihres Adapters anhand Ihres Trainingsdatensatzes
- Überprüfen der Leistung Ihres Adapters
- Verwenden Ihres Adapters für die Bildanalyse

Voraussetzungen

Bevor Sie dieses Tutorial abschließen, wird empfohlen, dass Sie Adapter erstellen und verwenden durchlesen.

Um einen Adapter zu erstellen, können Sie das Rekognition-Konsole-Tool verwenden, um ein Projekt zu erstellen, Ihre eigenen Bilder hochzuladen und mit Anmerkungen zu versehen und dann einen Adapter mit diesen Bildern zu trainieren. Informationen zum Einstieg finden Sie unter Ein Projekt erstellen und einen Adapter trainieren.

Alternativ können Sie die Konsole oder API von Rekognition verwenden, um Vorhersagen für Bilder abzurufen und die Vorhersagen dann zu überprüfen, bevor Sie einen Adapter anhand dieser Vorhersagen trainieren. Informationen zum Einstieg finden Sie unter Massenanalyse, Überprüfung der Vorhersagen und Training eines Adapters.

Bildanmerkung

Sie können Bilder selbst mit Anmerkungen versehen, indem Sie sie mit der Rekognition-Konsole beschriften. Oder Sie verwenden die Rekognition-Massenanalyse, um Bilder mit Anmerkungen zu versehen, deren korrekte Labels Sie anschließend überprüfen können. Wählen Sie eines der folgenden Themen aus, um loszulegen.

Themen

- Ein Projekt erstellen und einen Adapter trainieren
- Massenanalyse, Überprüfung der Vorhersagen und Training eines Adapters

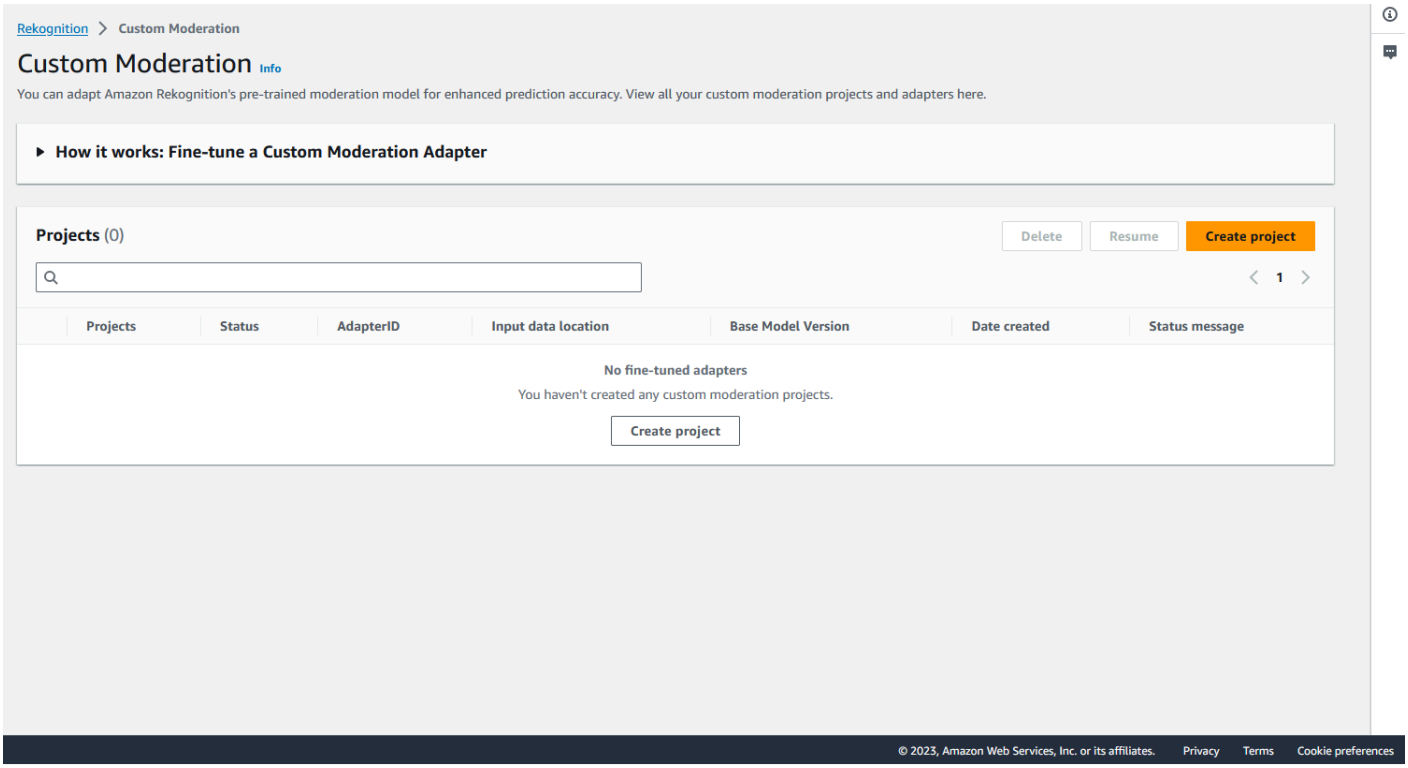
Ein Projekt erstellen und einen Adapter trainieren

Führen Sie die folgenden Schritte aus, um Ihren Adapter zu trainieren, indem Sie Bilder mit der Rekognition-Konsole kommentieren.

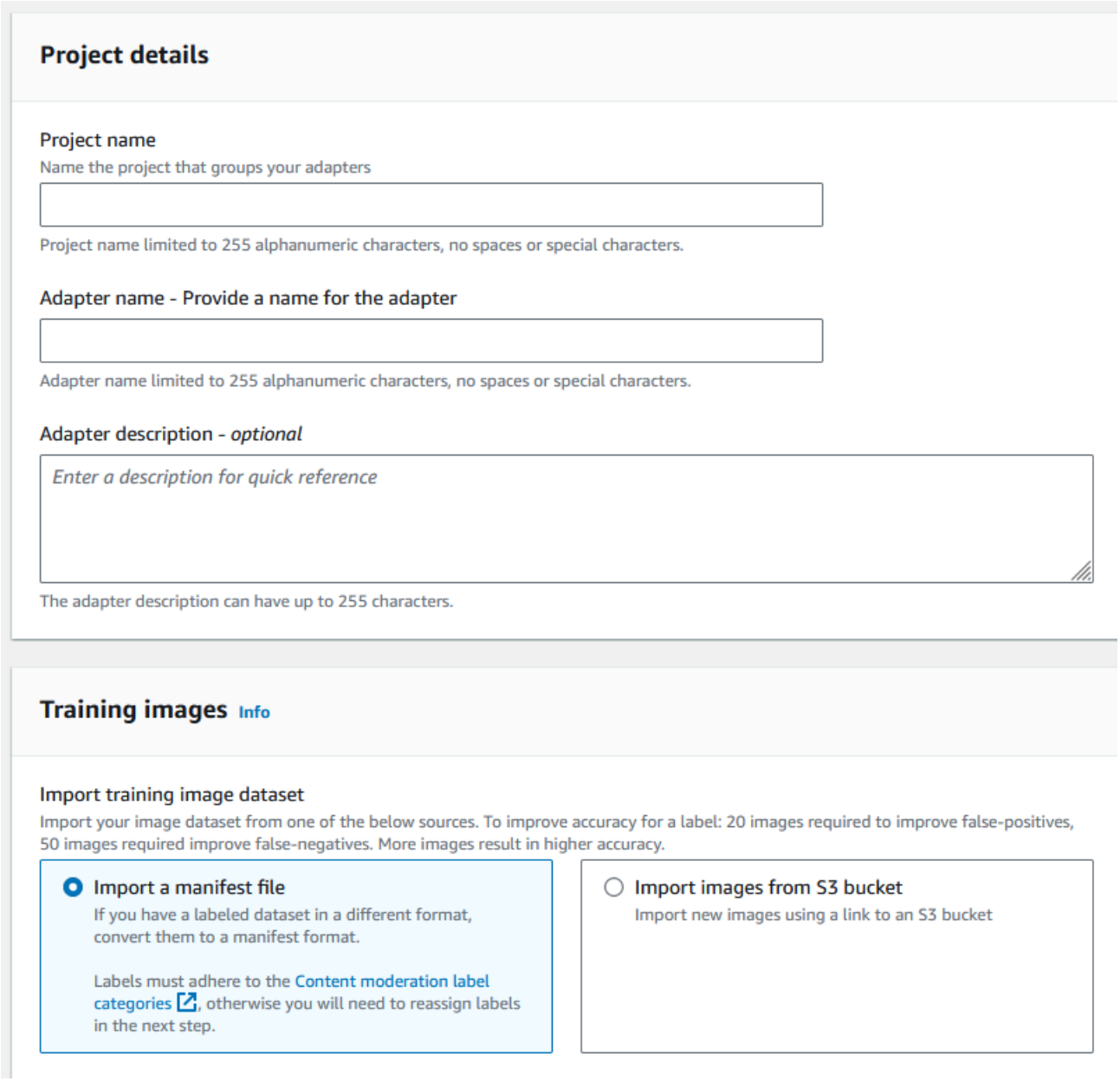
Erstellen eines Projekts

Bevor Sie einen Adapter trainieren oder verwenden können, müssen Sie das Projekt erstellen, das ihn enthalten soll. Sie müssen auch die Bilder bereitstellen, die zum Trainieren Ihres Adapters verwendet wurden. Um ein Projekt, einen Adapter und Ihre Bilddatensätze zu erstellen:

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Rekognition-Konsole unter. https://console.aws.amazon.com/rekognition/
2. Wählen Sie im linken Navigationsbereich Benutzerdefinierte Moderation aus. Die Landingpage der benutzerdefinierten Moderation von Rekognition wird angezeigt.



3. Die Landingpage der benutzerdefinierten Moderation zeigt Ihnen eine Liste all Ihrer Projekte und Adapter. Außerdem gibt es eine Schaltfläche, mit der Sie einen Adapter erstellen können. Wählen Sie Projekt erstellen aus, um ein neues Projekt und einen neuen Adapter zu erstellen.
4. Wenn Sie zum ersten Mal einen Adapter erstellen, werden Sie aufgefordert, einen Amazon-S3-Bucket zu erstellen, in dem Dateien gespeichert werden, die sich auf Ihr Projekt und Ihren Adapter beziehen. Wählen Sie Amazon-S3-Bucket erstellen aus.
5. Geben Sie auf der folgenden Seite den Adapternamen und den Projektnamen ein. Geben Sie bei Bedarf eine Adapterbeschreibung an.



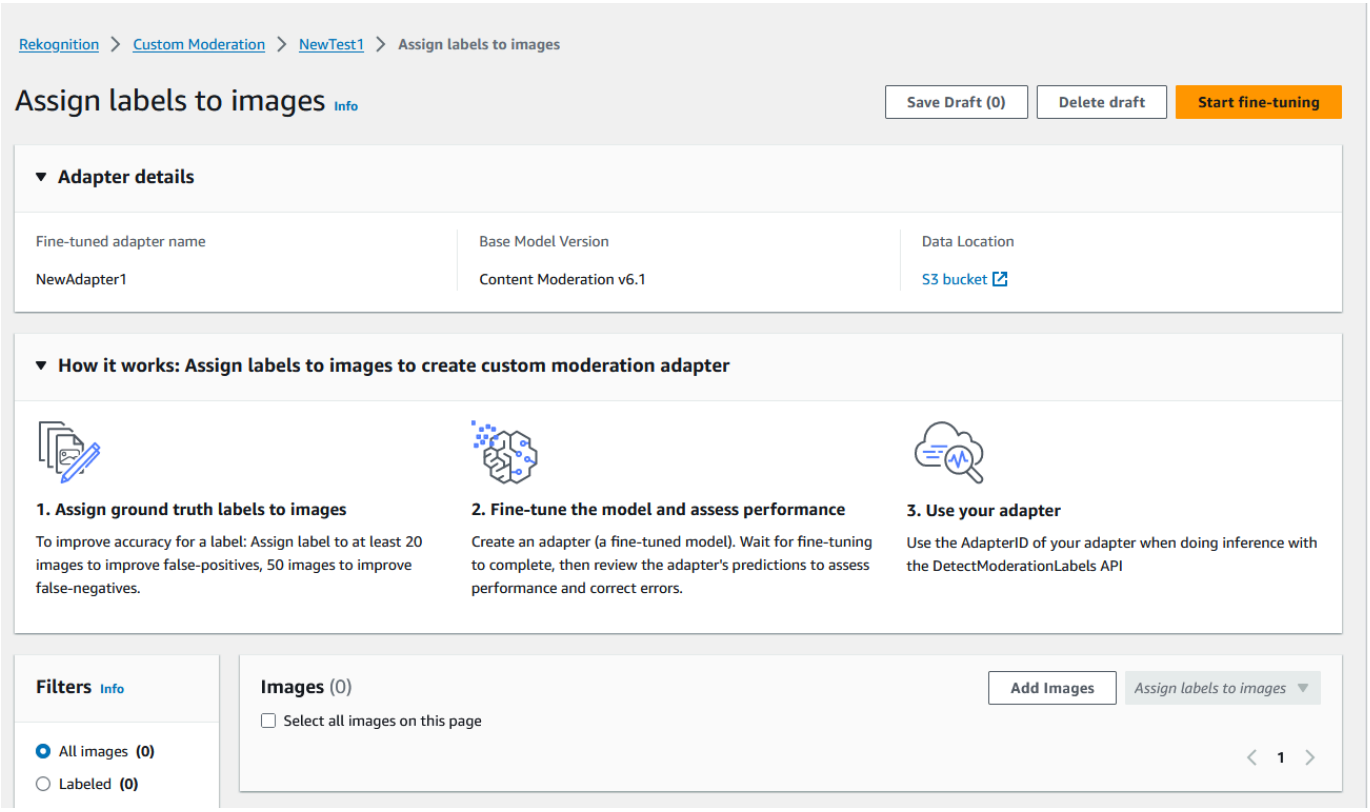
6. In diesem Schritt stellen Sie auch die Bilder für Ihren Adapter bereit. Sie können wählen: Bilder von Ihrem Computerimportieren, Manifestdatei importieren oder Bilder aus Amazon-S3-Bucket importieren. Wenn Sie Ihre Bilder aus einem Amazon-S3-Bucket importieren möchten, geben Sie den Pfad zu dem Bucket und dem Ordner an, der Ihre Trainingsbilder enthält. Wenn Sie Ihre Bilder direkt von Ihrem Computer hochladen, beachten Sie, dass Sie nur bis zu 30 Bilder gleichzeitig hochladen können. Wenn Sie eine Manifestdatei verwenden, die Anmerkungen enthält, können Sie die unten aufgeführten Schritte zur Bildanmerkung überspringen und mit dem Abschnitt über Überprüfen der Adapterleistung fortfahren.

7. Wählen Sie im Abschnitt Testdatensatzdetails die Option Autosplit, damit Rekognition automatisch den entsprechenden Prozentsatz Ihrer Bilder als Testdaten auswählt, oder Sie können Manifestdatei manuell importieren wählen.
8. Nachdem Sie diese Informationen eingegeben haben, wählen Sie Projekt erstellen.

Trainieren eines Adapters

So trainieren Sie einen Adapter mit Ihren eigenen Bildern ohne Anmerkungen:

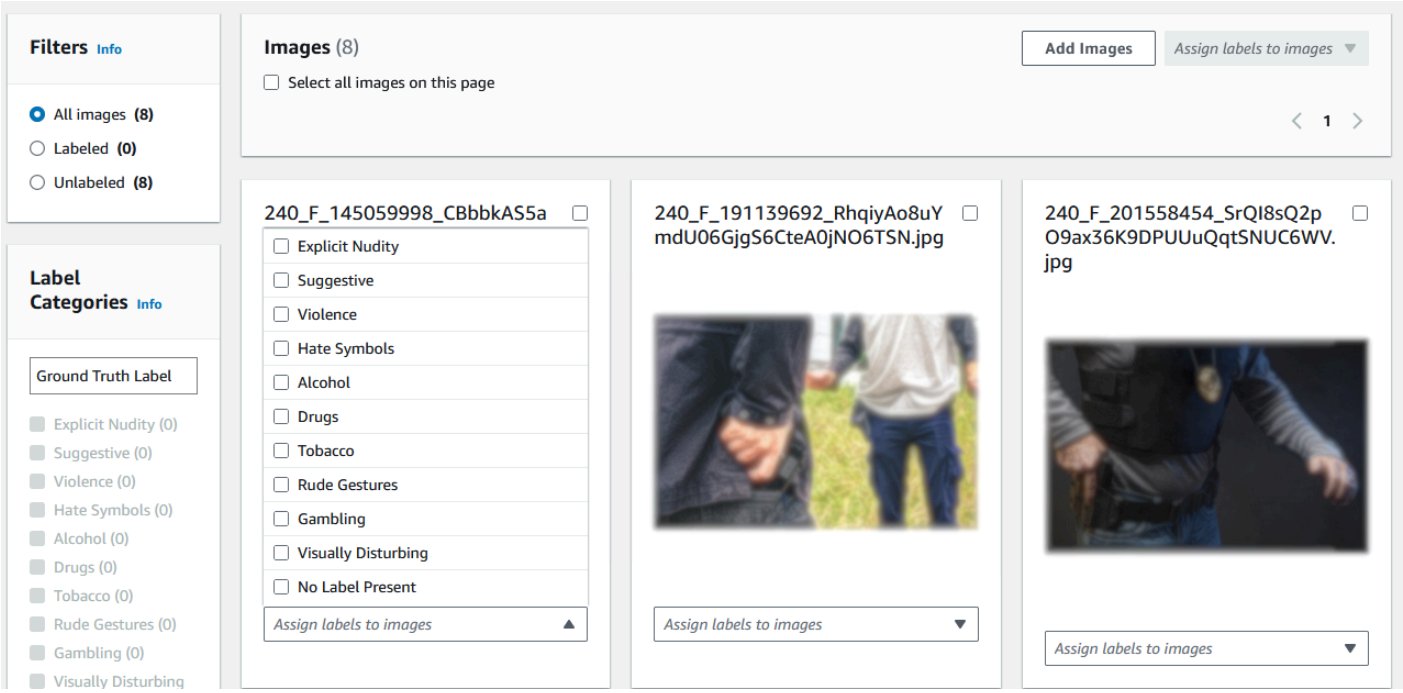
1. Wählen Sie das Projekt aus, das Ihren Adapter enthält, und wählen Sie dann die Option Bildern ein Label zuweisen.
2. Auf der Seite Bildern ein Label zuweisen sehen Sie alle Bilder, die als Trainingsbilder hochgeladen wurden. Mithilfe der beiden Attributauswahlfelder auf der linken Seite können Sie diese Bilder sowohl nach dem Status „Mit/ohne Label“ als auch nach der Labelkategorie filtern. Sie können Ihrem Trainingsdatensatz weitere Bilder hinzufügen, indem Sie auf die Schaltfläche Bilder hinzufügen klicken.



3. Nachdem Sie dem Trainingsdatensatz Bilder hinzugefügt haben, müssen Sie Ihre Bilder mit Labels versehen. Nach dem Hochladen Ihrer Bilder wird die Seite „Bildern Label zuweisen“ aktualisiert und zeigt nun die Bilder an, die Sie hochgeladen haben. Sie werden aufgefordert, das für Ihre

Bilder passende Label aus einer Drop-down-Liste mit Label auszuwählen, die von Rekognition Moderation unterstützt werden. Sie können mehr als ein Label auswählen.

4. Setzen Sie diese Operation fort, bis Sie jedem Bild in Ihren Trainingsdaten Labels hinzugefügt haben.
5. Nachdem Sie alle Ihre Daten beschriftet haben, wählen Sie Training starten aus, um mit dem Training des Modells zu beginnen, das Ihren Adapter erstellt.



6. Bevor Sie mit dem Training beginnen, können Sie dem Adapter beliebige Tags hinzufügen. Sie können dem Adapter auch einen benutzerdefinierten Verschlüsselungsschlüssel zur Verfügung stellen oder einen AWS KMS-Schlüssel verwenden. Wenn Sie alle gewünschten Tags hinzugefügt und die Verschlüsselung nach Ihren Wünschen angepasst haben, wählen Sie Adapter trainieren aus, um den Trainingsvorgang für Ihren Adapter zu starten.
7. Warten Sie, bis Ihr Adapter das Training beendet hat. Sobald das Training abgeschlossen ist, erhalten Sie eine Benachrichtigung, dass die Erstellung Ihres Adapters abgeschlossen ist.

Sobald der Status Ihres Adapters „Training abgeschlossen“ lautet, können Sie die Metriken Ihres Adapters überprüfen

Massenanalyse, Überprüfung der Vorhersagen und Training eines Adapters

Führen Sie die folgenden Schritte durch, um Ihren Adapter zu trainieren, indem Sie die Vorhersagen für die Massenanalyse anhand des Inhaltsmoderationsmodells von Rekognition verifizieren.

Um einen Adapter zu trainieren, indem Sie Vorhersagen aus dem Inhaltsmoderationsmodell von Rekognition verifizieren, müssen Sie:

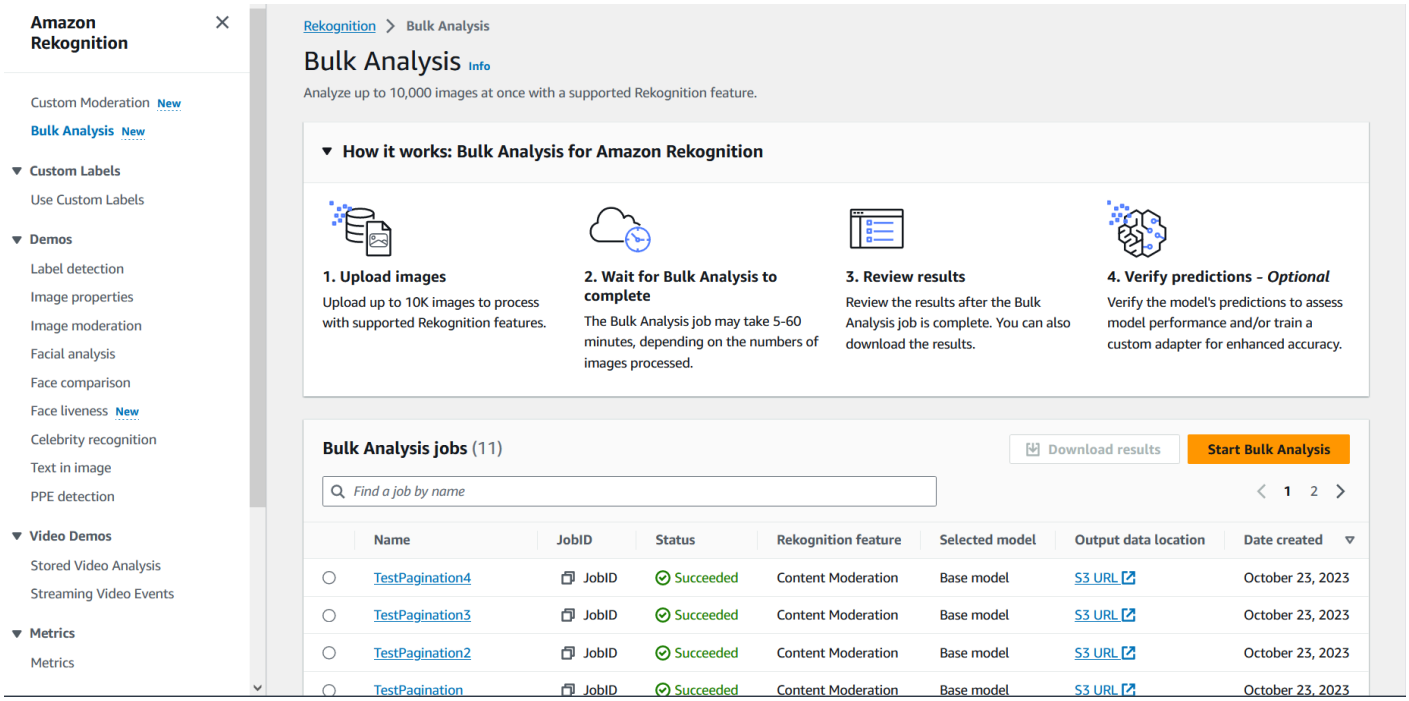
1. Führen Sie eine Massenanalyse Ihrer Bilder durch
2. Überprüfen Sie die für Ihre Bilder zurückgegebenen Vorhersagen

Sie können Vorhersagen für Bilder erhalten, indem Sie eine Massenanalyse mit dem Basismodell von Rekognition oder einem Adapter, den Sie bereits erstellt haben, durchführen.

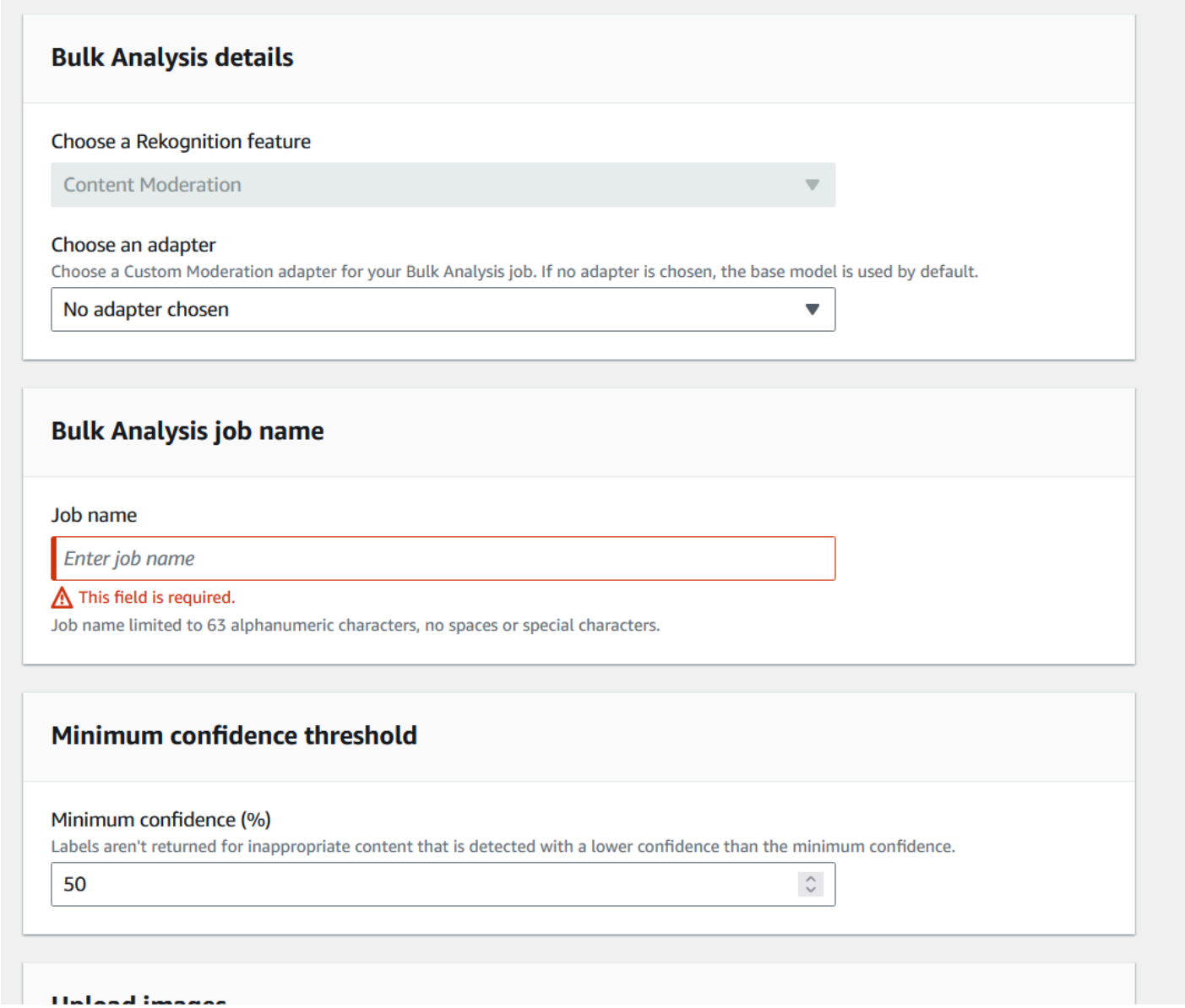
Führen Sie eine Massenanalyse Ihrer Bilder durch

Um einen Adapter anhand von Vorhersagen zu trainieren, die Sie verifiziert haben, müssen Sie zunächst eine Massenanalyse starten, um einen Stapel von Bildern mit dem Basismodell von Rekognition oder einem Adapter Ihrer Wahl zu analysieren. So führen Sie einen Massenanalyseauftrag aus:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Rekognition Rekognition-Konsole unter. https://console.aws.amazon.com/rekognition/
2. Wählen Sie im linken Navigationsbereich Massenanalyse aus. Die Landingpage für die Massenanalyse wird angezeigt. Wählen Sie Massenanalyse starten. Die Übersicht über die Funktion „Massenanalyse“ zeigt die Schritte zum Hochladen von Bildern, zum Warten auf die Analyse, zum Überprüfen der Ergebnisse und optional zum Überprüfen von Modellvorhersagen. Führt die letzten Massenanalyseaufträge für die Inhaltsmoderation unter Verwendung des Basismodells auf.



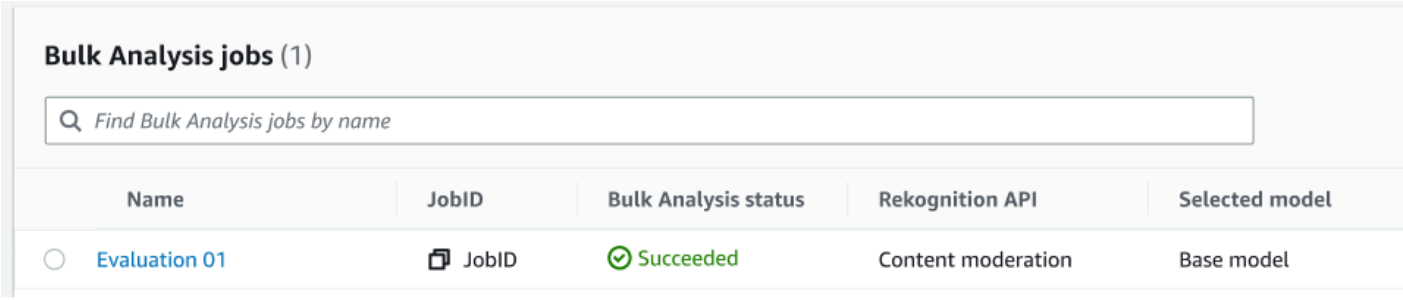
3. Wenn Sie zum ersten Mal einen Adapter erstellen, werden Sie aufgefordert, einen Amazon-Simple-Storage-Service-Bucket zu erstellen, in dem Dateien gespeichert werden, die sich auf Ihr Projekt und Ihren Adapter beziehen. Wählen Sie Amazon-S3-Bucket erstellen aus.
4. Wählen Sie im Drop-down-Menü Adapter auswählen den Adapter aus, den Sie für die Massenanalyse verwenden möchten. Wenn kein Adapter ausgewählt wurde, wird standardmäßig das Basismodell verwendet. Für dieses Tutorial wählen Sie keinen Adapter.



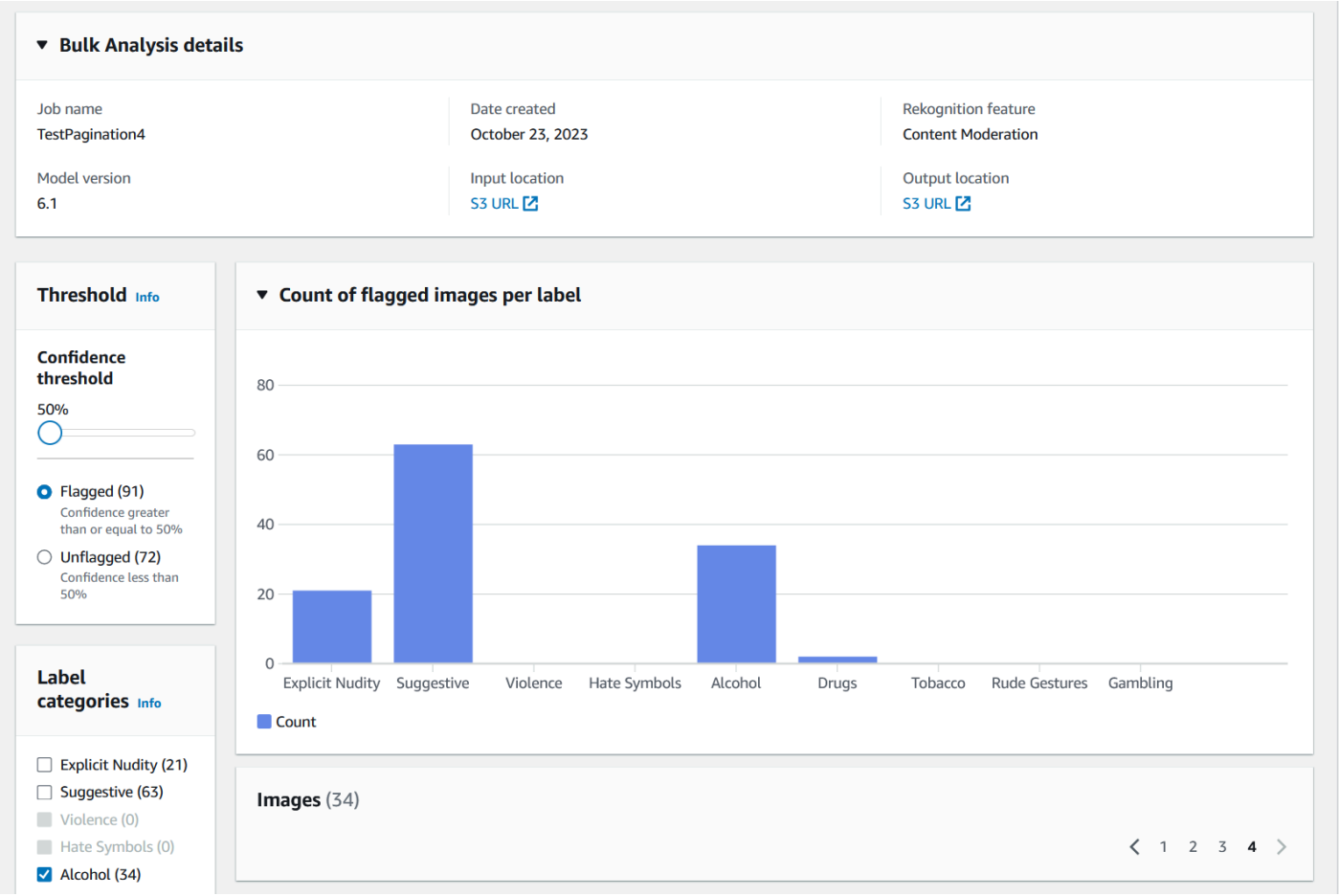
5. Geben Sie im Feld Name des Massenanalyseauftrags den Namen des Massenanalyseauftrags ein.
6. Wählen Sie einen Wert für den Mindestzuverlässigkeitsschwellenwert aus. Labelvorhersagen, deren Zuverlässigkeitsschwelle unter dem von Ihnen gewählten Zuverlässigkeitsschwellenwert liegt, werden nicht zurückgegeben. Beachten Sie, dass Sie bei der späteren Bewertung der Leistung des Modells den Zuverlässigkeitsschwellenwert nicht unter den ausgewählten Mindestzuverlässigkeitsschwellenwert anpassen können.
7. In diesem Schritt stellen Sie auch die Bilder bereit, die Sie mit der Massenanalyse analysieren wollen. Diese Bilder können auch zum Trainieren Ihres Adapters verwendet werden. Sie können Bilder von Ihrem Computer hochladen oder Bilder aus Amazon-S3-Bucket importieren auswählen. Wenn Sie Ihre Dokumente aus einem Amazon-S3-Bucket importieren möchten,

geben Sie den Pfad zu dem Bucket und dem Ordner an, der Ihre Trainingsbilder enthält. Wenn Sie Ihre Dokumente direkt von Ihrem Computer hochladen, beachten Sie, dass Sie nur 50 Bilder gleichzeitig hochladen können.

8. Nachdem Sie diese Informationen eingegeben haben, wählen Sie Analyse starten. Dadurch wird der Analyseprozess mit dem Basismodell von Rekognition gestartet.
9. Sie können den Status Ihres Massenanalyseauftrags überprüfen, indem Sie den Status der Massenanalyse des Jobs auf der Hauptseite der Massenanalyse überprüfen. Wenn der Status der Massenanalyse „Erfolgreich“ lautet, können die Ergebnisse der Analyse überprüft werden.



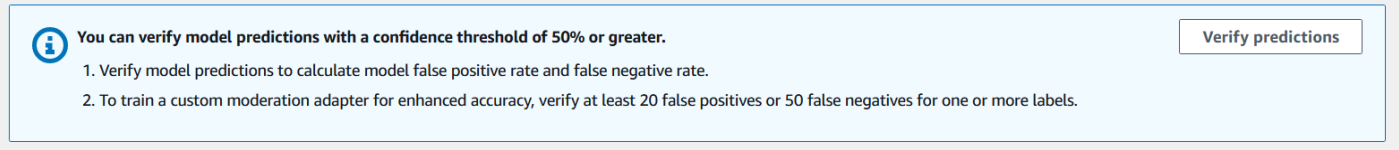
10. Wählen Sie die Analyse, die Sie erstellt haben, aus der Liste der Massenanalyseaustrag aus.
11. Auf der Detailseite zur Massenanalyse können Sie die Vorhersagen sehen, die das Basismodell von Rekognition für die von Ihnen hochgeladenen Bilder gemacht hat.
12. Überprüfen Sie die Leistung des Basismodells. Sie können den Zuverlässigkeitsschwellenwert ändern, über den Ihr Adapter verfügen muss, um einem Bild ein Label zuzuweisen, indem Sie den Schieberegler Zuverlässigkeitsschwellenwert verwenden. Die Anzahl der markierten und nicht markierten Instances ändert sich, wenn Sie den Zuverlässigkeitsschwellenwert anpassen. Im Bereich „Labelkategorien“ werden die Kategorien der obersten Ebene angezeigt, die Rekognition erkennt. Sie können in dieser Liste eine Kategorie auswählen, um alle Bilder anzuzeigen, denen dieses Label zugewiesen wurde.



Bestätigen der Vorhersagen

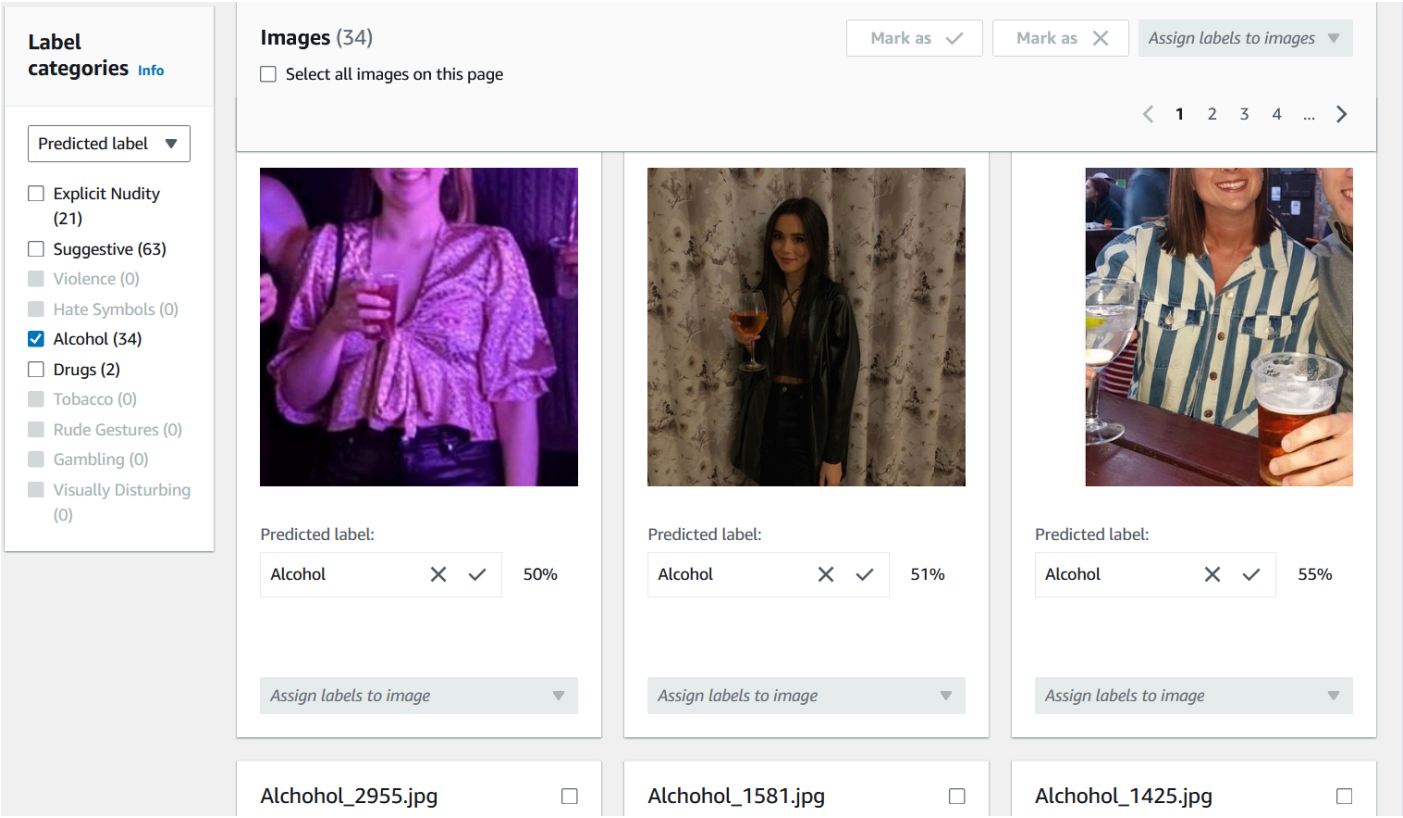
Wenn Sie die Genauigkeit des Basismodells von Rekognition oder eines ausgewählten Adapters überprüft haben und diese Genauigkeit verbessern möchten, können Sie den Verifizierungs-Workflow verwenden:

1. Wenn Sie mit der Überprüfung der Leistung des Basismodells fertig sind, sollten Sie die Vorhersagen überprüfen. Wenn Sie die Vorhersagen korrigieren, können Sie einen Adapter trainieren. Wählen Sie oben auf der Seite für die Massenanalyse die Option Vorhersagen verifizieren aus.

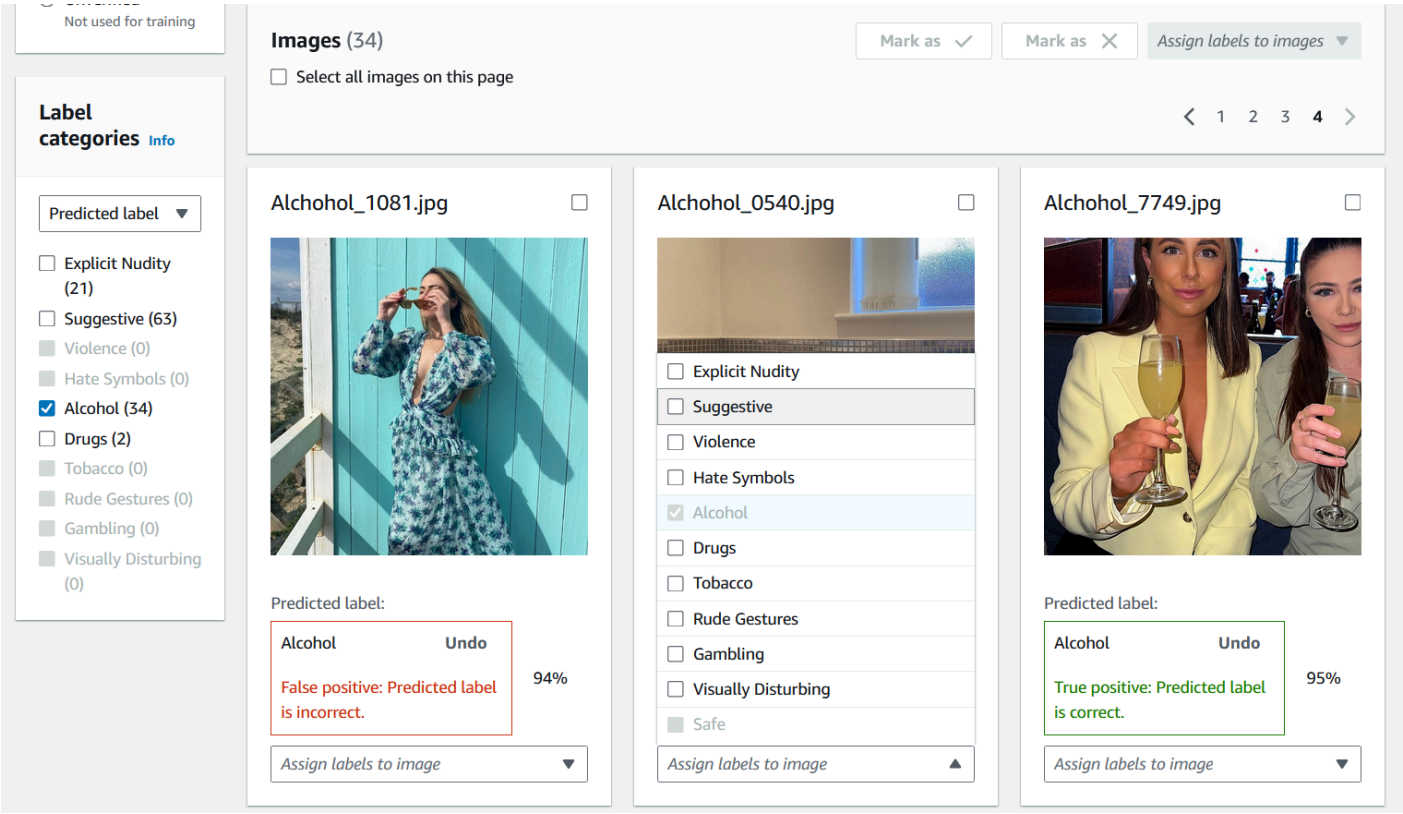


2. Auf der Seite „Vorhersagen verifizieren“ können Sie alle Bilder sehen, die Sie dem Basismodell von Rekognition oder einem ausgewählten Adapter zur Verfügung gestellt haben, zusammen mit

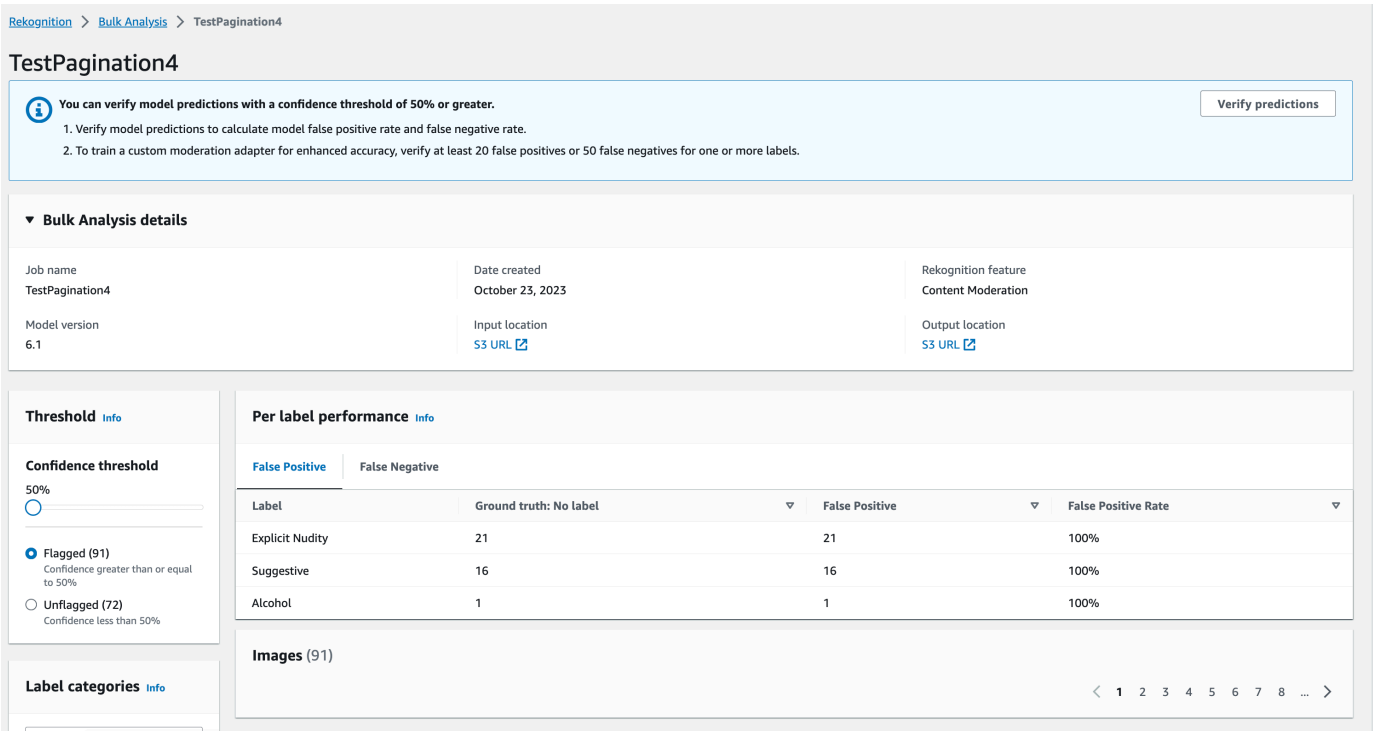
dem vorhergesagten Label für jedes Bild. Sie müssen jede Vorhersage mithilfe der Schaltflächen unter dem Bild als richtig oder falsch verifizieren. Verwenden Sie die Schaltfläche „X“, um eine Vorhersage als falsch zu markieren, und die Schaltfläche mit dem Häkchen, um eine Vorhersage als richtig zu markieren. Um einen Adapter zu trainieren, müssen Sie mindestens 20 falsch positive Vorhersagen und 50 falsch negative Vorhersagen für ein bestimmtes Label verifizieren. Je mehr Vorhersagen Sie überprüfen, desto besser ist die Leistung des Adapters.



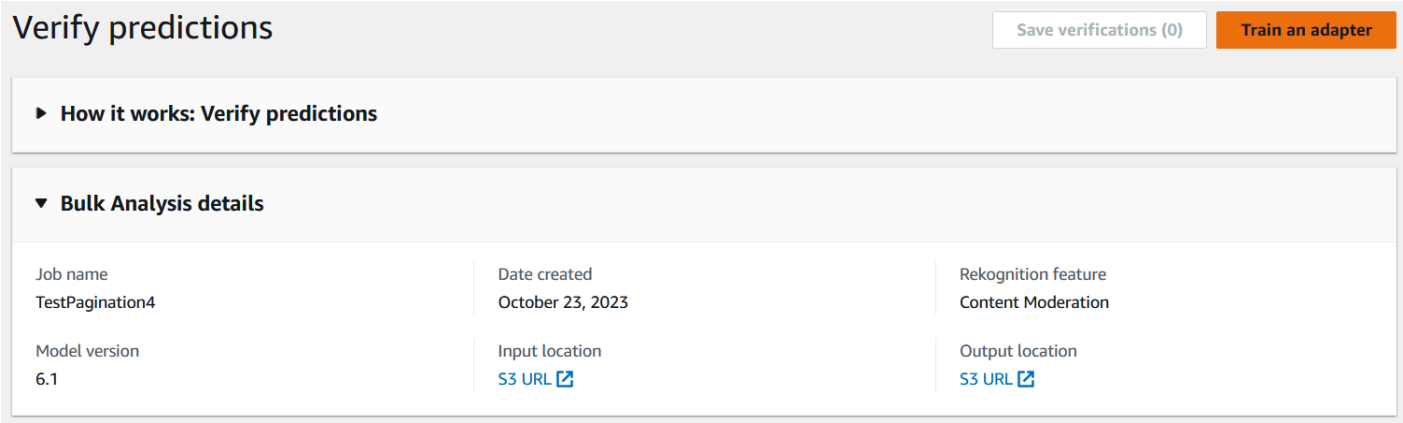
Nachdem Sie eine Vorhersage verifiziert haben, ändert sich der Text unter dem Bild und zeigt Ihnen, welche Art von Vorhersage Sie verifiziert haben. Sobald Sie ein Bild verifiziert haben, können Sie dem Bild auch weitere Labels hinzufügen, indem Sie das Menü Labels dem Bild zuweisen verwenden. Sie können sehen, welche Bilder vom Modell für den von Ihnen gewählten Zuverlässigkeitsschwellenwert gekennzeichnet oder nicht markiert wurden, oder Sie können Bilder nach Kategorie filtern.



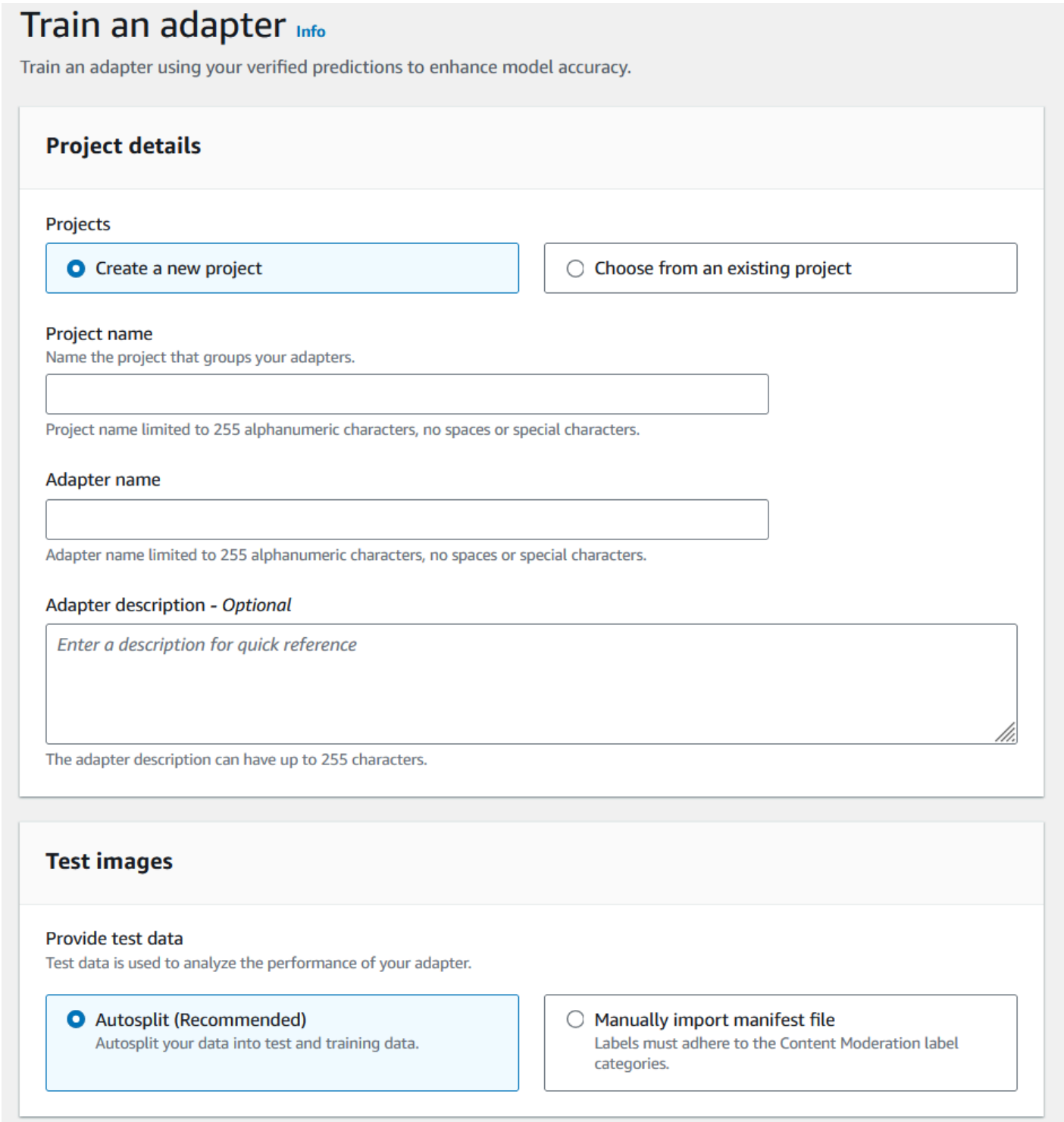
3. Sobald Sie die Überprüfung aller Vorhersagen abgeschlossen haben, die Sie verifizieren möchten, können Sie auf der Seite „Überprüfung“ im Abschnitt Leistung pro Label Statistiken zu Ihren verifizierten Vorhersagen einsehen. Sie können auch zur Detailseite für die Massenanalyse zurückkehren, um sich diese Statistiken anzusehen.



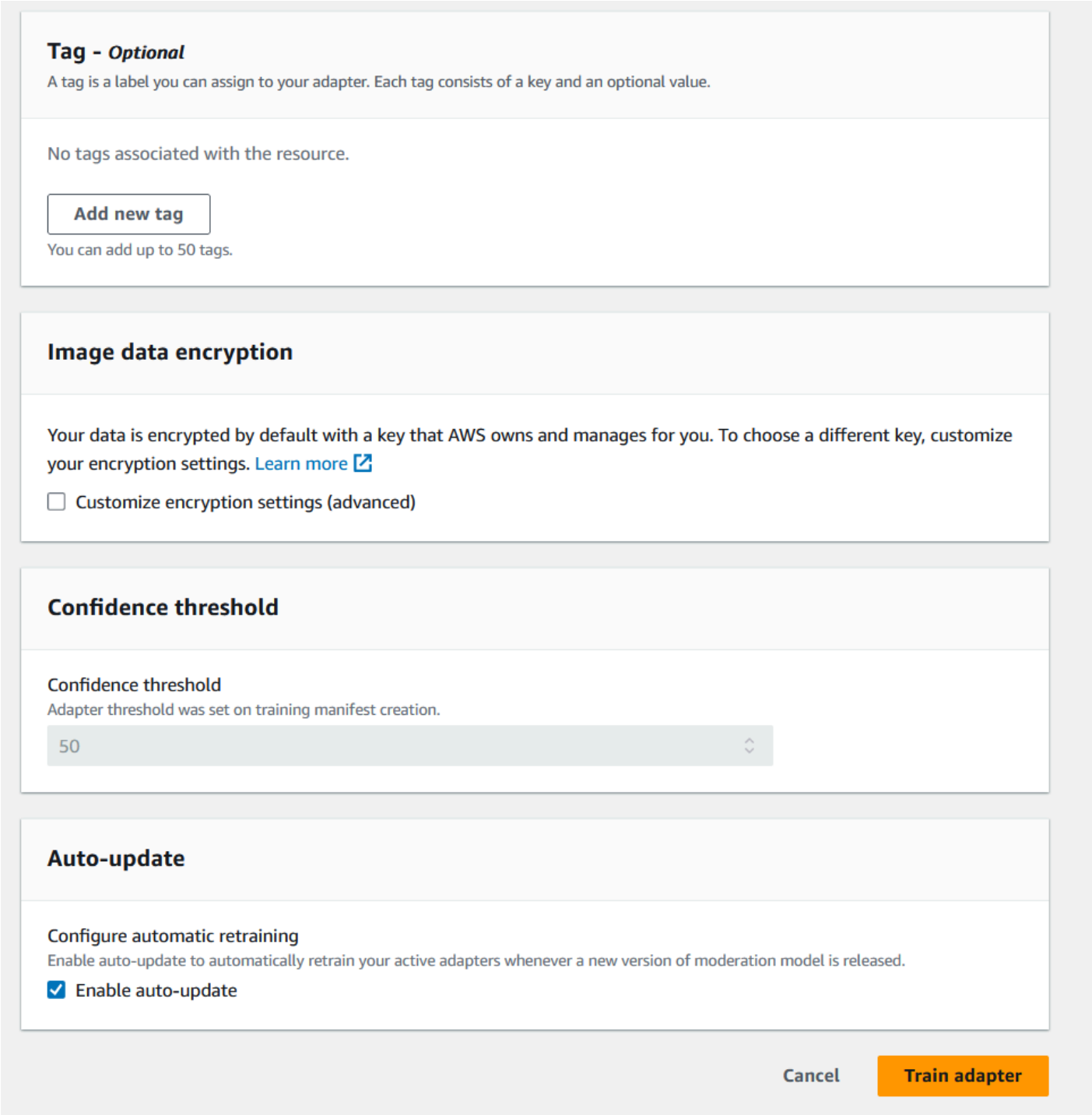
4. Wenn Sie mit den Statistiken zur Leistung pro Label zufrieden sind, rufen Sie erneut die Seite Vorhersagen verifizieren auf und klicken Sie dann auf die Schaltfläche Adapter trainieren, um mit dem Training Ihres Adapters zu beginnen.



5. Auf der Seite „Adapter trainieren“ werden Sie aufgefordert, ein Projekt zu erstellen oder ein vorhandenes Projekt auszuwählen. Benennen Sie das Projekt und den Adapter, die im Projekt enthalten sein werden. Sie müssen auch die Quelle Ihrer Testbilder angeben. Bei der Angabe der Bilder können Sie Autosplit wählen, damit Rekognition automatisch einen Teil Ihrer Trainingsdaten als Testbilder verwendet, oder Sie können manuell eine Manifestdatei angeben. Es wird empfohlen, Autosplit zu wählen.



6. Geben Sie alle gewünschten Tags sowie einen AWS KMS Schlüssel an, wenn Sie den AWS Standardschlüssel nicht verwenden möchten. Es wird empfohlen, die automatische Aktualisierung aktiviert zu lassen.
7. Wählen Sie Adapter trainieren.

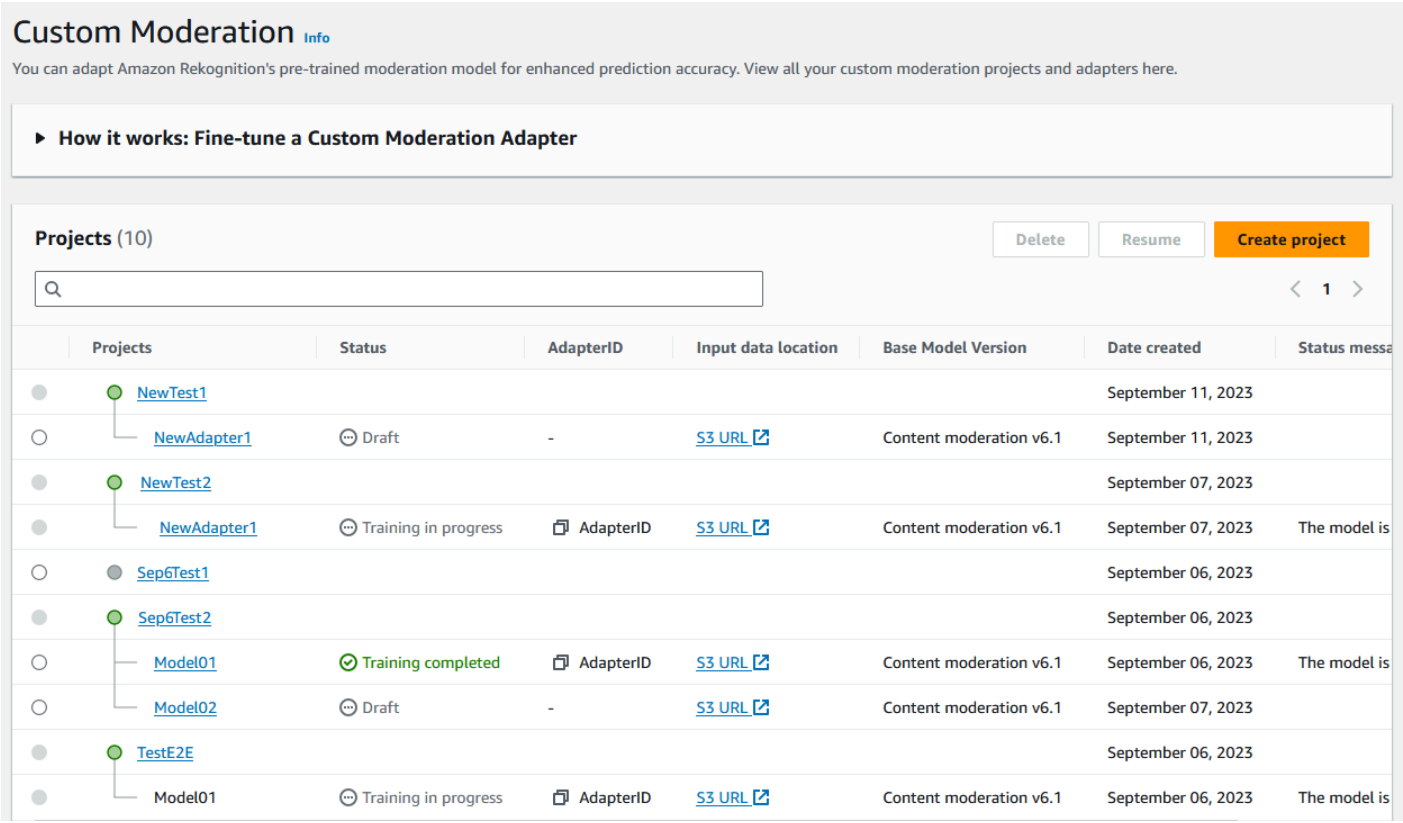


8. Sobald der Status Ihres Adapters auf der Startseite „Benutzerdefinierte Moderation“ den Status „Training abgeschlossen“ hat, können Sie die Leistung Ihres Adapters überprüfen. Weitere Informationen finden Sie unter Überprüfen der Adapterleistung.

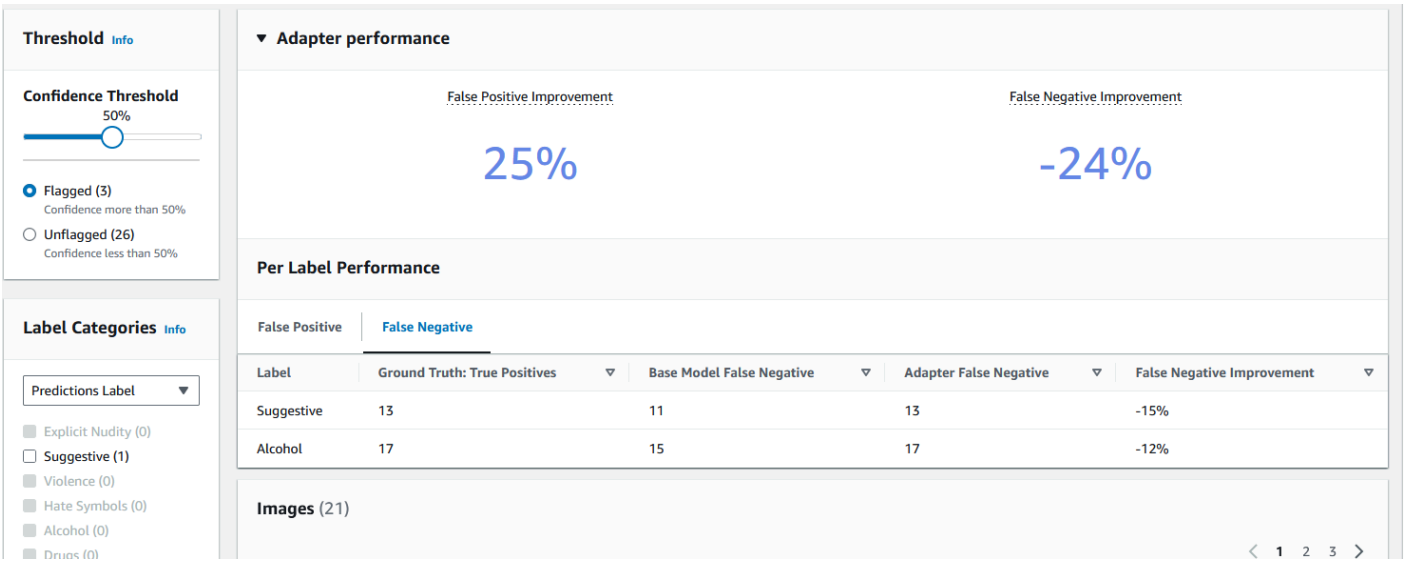
Überprüfen der Adapterleistung

So überprüfen Sie die Leistung Ihres Adapters:

1. Wenn Sie die Konsole verwenden, können Sie den Status aller Adapter, die einem Projekt zugeordnet sind, auf der Landingpage „Benutzerdefinierte Moderation“ auf der Registerkarte „Projekte“ einsehen. Navigieren Sie zur Landingpage „Benutzerdefinierte Moderation“.



2. Wählen Sie den Adapter, den Sie überprüfen möchten, aus dieser Liste aus. Auf der folgenden Seite mit den Adapterdetails können Sie eine Vielzahl von Metriken für den Adapter sehen.

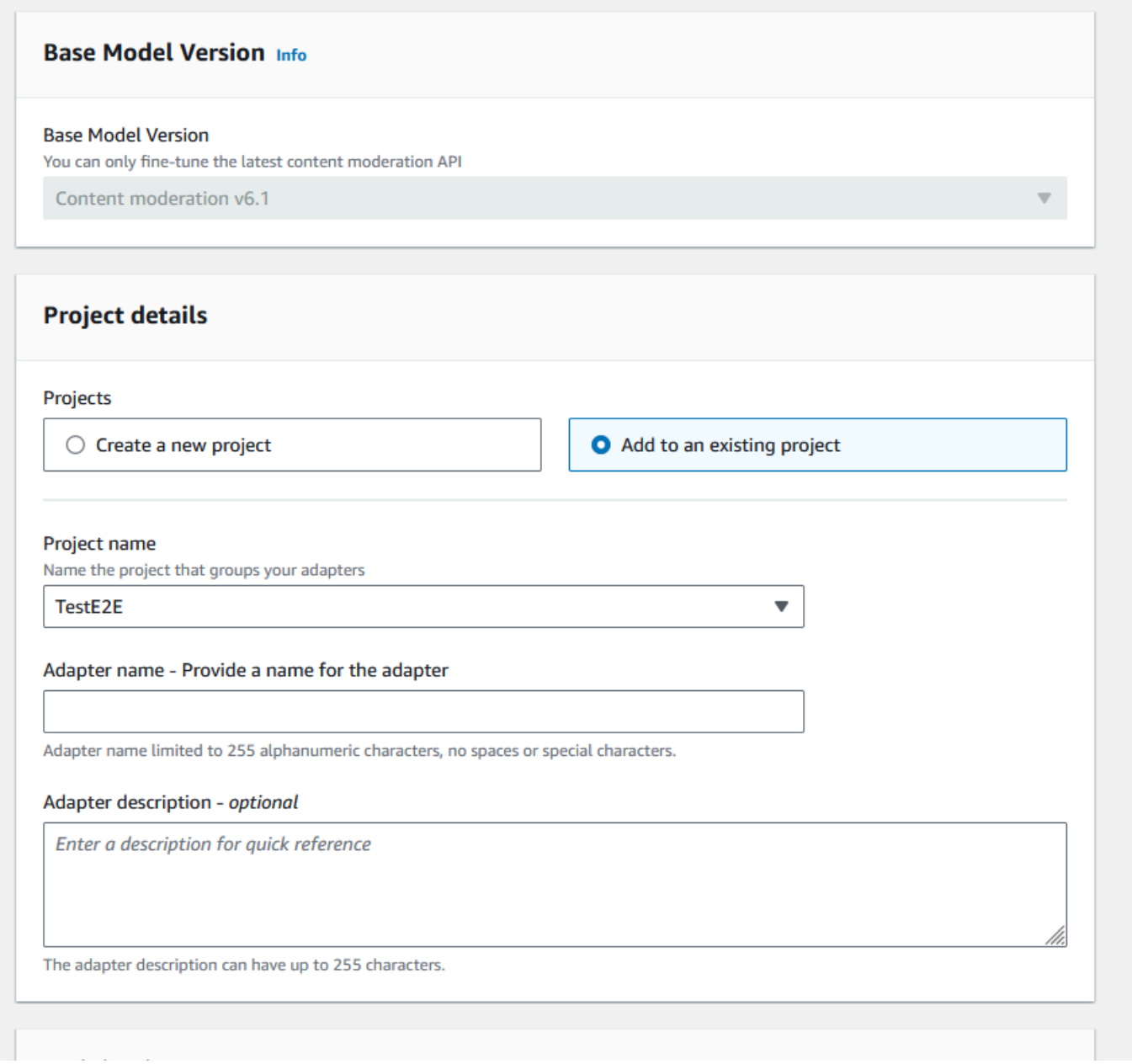


3. Im Bereich Schwellenwert können Sie den Mindestzuverlässigkeitsschwellenwert ändern, über den Ihr Adapter verfügen muss, um einem Bild ein Label zuzuweisen. Die Anzahl der markierten und nicht markierten Instances ändert sich, wenn Sie den Zuverlässigkeitsschwellenwert anpassen. Sie können auch nach Labelkategorie filtern, um Metriken für die von Ihnen ausgewählten Kategorien zu sehen. Legen Sie den von Ihnen gewählten Schwellenwert fest.
4. Sie können die Leistung Ihres Adapters anhand Ihrer Testdaten bewerten, indem Sie die Metriken im Bereich „Adapterleistung“ untersuchen. Diese Metriken werden berechnet, indem die Extraktionen des Adapters mit den „Grundwahrheits“-Annotationen auf dem Testdatensatz verglichen werden.

Im Leistungsbereich des Adapters werden die Verbesserungsraten falsch positiv und falsch negativ für den Adapter angezeigt, den Sie erstellt haben. Auf der Registerkarte „Leistung pro Label“ können Sie die Leistung von Adapter und Basismodell für jede Labelkategorie vergleichen. Sie zeigt die Anzahl falsch positiver und falsch negativer Vorhersagen sowohl durch das Basismodell als auch durch den Adapter, stratifiziert nach Labelkategorien. Anhand dieser Kennzahlen können Sie feststellen, wo der Adapter verbessert werden muss. Weitere Informationen zu diesen Metriken, finden Sie unter Evaluierung und Verbesserung Ihres Adapters.

Um die Leistung zu verbessern, können Sie mehr Trainingsbilder sammeln und dann innerhalb des Projekts einen neuen Adapter erstellen. Kehren Sie einfach zur Landingpage der benutzerdefinierten Moderation zurück und erstellen Sie innerhalb Ihres Projekts einen neuen Adapter, der mehr Trainingsbilder für den zu trainierenden Adapter bereitstellt. Wählen Sie diesmal die Option Zu einem vorhandenen Projekt hinzufügen statt Neues Projekt erstellen und wählen Sie im Drop-down-Menü

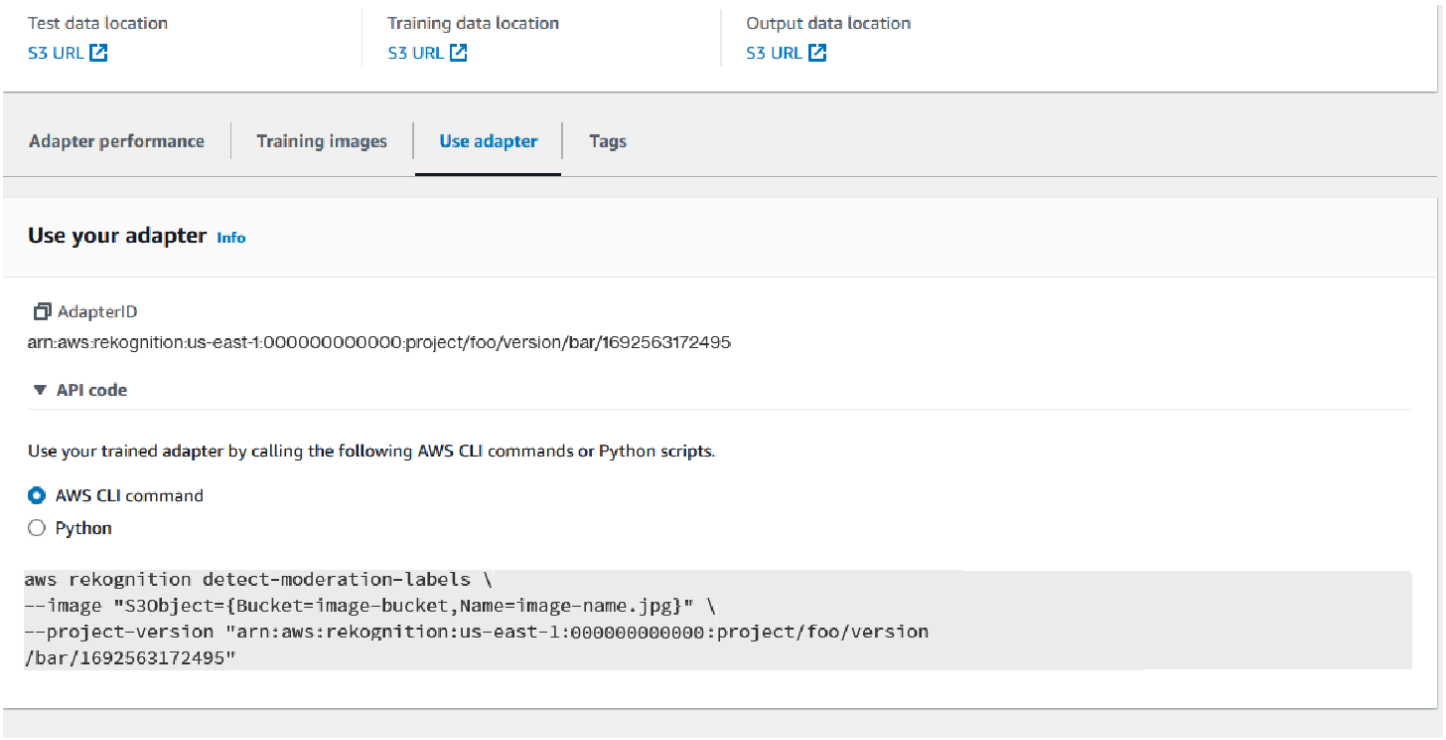
Projektname das Projekt aus, für das Sie den neuen Adapter erstellen möchten. Kommentieren Sie wie zuvor Ihre Bilder oder stellen Sie eine Manifestdatei mit Anmerkungen bereit.



Verwendung Ihres Adapters

Nachdem Sie Ihren Adapter erstellt haben, können Sie ihn für eine unterstützte Rekognition-Operation wie verwenden. DetectModerationLabels Um Codebeispiele zu sehen, die Sie verwenden können, um Inferenzen mit Ihrem Adapter durchzuführen, wählen Sie die Registerkarte „Adapter verwenden“, auf der Sie Codebeispiele sowohl für die AWS CLI als auch für Python sehen können. Weitere Codebeispiele, Setup-Anweisungen und ein JSON-Beispiel finden Sie auch im

entsprechenden Abschnitt der Dokumentation für die Operation, für die Sie einen Adapter erstellt haben.



Ihren Adapter und Ihr Projekt löschen

Sie können einzelne Adapter oder Ihr Projekt löschen. Sie müssen jeden Adapter löschen, der Ihrem Projekt zugeordnet ist, bevor das Projekt selbst gelöscht werden kann.

1. Um einen mit dem Projekt verknüpften Adapter zu löschen, wählen Sie den Adapter aus und klicken Sie dann auf Löschen.
2. Um ein Projekt zu löschen, wählen Sie das Projekt aus, das Sie löschen möchten, und wählen Sie dann Löschen.

Evaluierung und Verbesserung Ihres Adapters

Nach jeder Adaptertrainingsrunde sollten Sie die Leistungskennzahlen im Rekognition-Konsole-Tool überprüfen, um festzustellen, wie nahe der Adapter an Ihrem gewünschten Leistungsniveau ist. Anschließend können Sie die Genauigkeit Ihres Adapters für Ihre Bilder weiter verbessern, indem Sie einen neuen Stapel von Trainingsbildern hochladen und einen neuen Adapter in Ihrem Projekt trainieren. Sobald Sie eine verbesserte Version des Adapters erstellt haben, können Sie die Konsole verwenden, um alle älteren Versionen des Adapters zu löschen, die Sie nicht mehr benötigen.

Sie können Metriken auch mithilfe der DescribeProjectVersionsAPI-Operation abrufen.

Leistungsmetriken

Nachdem Sie den Trainingsprozess abgeschlossen und Ihren Adapter erstellt haben, müssen Sie unbedingt überprüfen, wie gut der Adapter Informationen aus Ihren Bildern extrahiert.

In der Rekognition-Konsole stehen Ihnen zwei Metriken zur Verfügung, die Sie bei der Analyse der Leistung Ihres Adapters unterstützen: falsch positive Verbesserung und falsch negative Verbesserung.

Sie können sich diese Metriken für jeden Adapter ansehen, indem Sie im Adapterbereich der Konsole den Tab „Adapterleistung“ auswählen. Im Leistungsbereich des Adapters werden die Verbesserungsraten falsch positiv und falsch negativ für den Adapter angezeigt, den Sie erstellt haben.

Die Verbesserung falsch positiver Ergebnisse gibt an, um wie viel sich die Erkennung falsch-positiver Ergebnisse durch den Adapter im Vergleich zum Basismodell verbessert hat. Wenn der falsch positive Verbesserungswert 25 % beträgt, bedeutet dies, dass der Adapter seine Erkennung falsch positiver Ergebnisse im Testdatensatz um 25 % verbessert hat.

Bei der Verbesserung falsch negativer Ergebnisse wird gemessen, um wie viel sich die Erkennung falsch-negativer Werte durch den Adapter im Vergleich zum Basismodell verbessert hat. Wenn der falsch negative Verbesserungswert 25 % beträgt, bedeutet dies, dass der Adapter seine Erkennung falsch negativer Ergebnisse im Testdatensatz um 25 % verbessert hat.

Auf der Registerkarte „Leistung pro Label“ können Sie die Leistung von Adapter und Basismodell für jede Labelkategorie vergleichen. Sie zeigt die Anzahl falsch positiver und falsch negativer Vorhersagen sowohl durch das Basismodell als auch durch den Adapter, stratifiziert nach Labelkategorien. Anhand dieser Kennzahlen können Sie feststellen, wo der Adapter verbessert werden muss.

Wenn beispielsweise die Falsch-Negativ-Rate des Basismodells für die Kategorie Alkohol 15 beträgt, während die Falsch-Negativ-Rate des Adapters 15 oder mehr beträgt, wissen Sie, dass Sie sich bei der Erstellung eines neuen Adapters darauf konzentrieren sollten, mehr Bilder hinzuzufügen, die das Label Alkohol enthalten.

Bei Verwendung der Rekognition-API-Operationen wird beim Aufrufen der Operation die F1-Score-Metrik zurückgegeben. DescribeProjectVersions

Verbessern Ihres Modells

Die Bereitstellung von Adaptern ist ein iterativer Prozess, da Sie einen Adapter wahrscheinlich mehrmals trainieren müssen, um die angestrebte Genauigkeit zu erreichen. Nachdem Sie Ihren Adapter erstellt und trainiert haben, sollten Sie die Leistung Ihres Adapters auf verschiedenen Labeltypen testen und bewerten.

Wenn die Genauigkeit Ihres Adapters in irgendeinem Bereich unzureichend ist, fügen Sie neue Beispiele für diese Bilder hinzu, um die Leistung des Adapters für diese Labels zu erhöhen. Versuchen Sie, dem Adapter zusätzliche, vielfältige Beispiele zur Verfügung zu stellen, die die Fälle widerspiegeln, in denen er Probleme hat. Wenn Sie Ihrem Adapter repräsentative, abwechslungsreiche Bilder zur Verfügung stellen, kann er verschiedene Beispiele aus der Praxis verarbeiten.

Nachdem Sie Ihrem Trainingsset neue Bilder hinzugefügt haben, trainieren Sie den Adapter erneut und bewerten Sie ihn anschließend anhand Ihres Testsets und Labels erneut. Wiederholen Sie diese Operation, bis der Adapter das gewünschte Leistungsniveau erreicht hat. Wenn Sie aussagekräftigere Bilder und Anmerkungen bereitstellen, verbessern sich falsch positive und falsch negative Punkte im Laufe aufeinanderfolgender Trainingsiterationen allmählich.

Manifest-Dateiformate

Die folgenden Abschnitte enthalten Beispiele für die Manifestdateiformate für Eingabe-, Ausgabe- und Evaluierungsdateien.

Eingabemanifest

Eine Manifestdatei ist eine durch Json-Zeilen begrenzte Datei, wobei jede Zeile ein JSON enthält, das Informationen über ein einzelnes Bild enthält.

Jeder Eintrag im Eingabemanifest muss das `source-ref`-Feld mit einem Pfad zum Bild im Amazon-S3-Bucket und für benutzerdefinierte Moderation das `content-moderation-groundtruth`-Feld mit Grundanmerkungen enthalten. Es wird erwartet, dass sich alle Bilder aus einem Datensatz in demselben Bucket befinden. Die Struktur ist sowohl für Trainings- als auch für Testmanifestdateien gleich.

Die `CreateProjectVersion`-Operation für die benutzerdefinierte Moderation verwendet die im Eingabemanifest bereitgestellten Informationen, um einen Adapter zu trainieren.

Das folgende Beispiel zeigt eine Zeile einer Manifestdatei für ein einzelnes Bild, das eine einzelne unsichere Klasse enthält:

```
{
   "source-ref": "s3://foo/bar/1.jpg",
   "content-moderation-groundtruth": {
        "ModerationLabels": [
            {
                "Name": "Rude Gesture"
            }
        ]
   }
}
```

Das folgende Beispiel ist eine Zeile einer Manifestdatei für ein einzelnes, unsicheres Bild, das mehrere unsichere Klassen enthält, insbesondere Nacktheit und unhöfliche Geste.

```
{
   "source-ref": "s3://foo/bar/1.jpg",
   "content-moderation-groundtruth": {
        "ModerationLabels": [
            {
                "Name": "Rude Gesture"
            },
            {
                "Name": "Nudity"
            }
        ]
   }
}
```

Das folgende Beispiel ist eine Zeile einer Manifestdatei für ein einzelnes Bild, das keine unsicheren Klassen enthält:

```
{
   "source-ref": "s3://foo/bar/1.jpg",
   "content-moderation-groundtruth": {
        "ModerationLabels": []
   }
```

```
}
```

Die vollständige Liste der unterstützten Labels finden Sie unter Inhalte moderieren.

Ausgabemanifest

Nach Abschluss eines Trainingsjobs wird eine Ausgabemanifestdatei zurückgegeben. Die Ausgabe-Manifestdatei ist eine durch Zeilen getrennte JSON-Datei, wobei jede Zeile eine JSON-Datei enthält, die Informationen für ein einzelnes Bild enthält. Der Amazon S3 S3-Pfad zu der OutputManifest kann über die folgende `DescribeProjectVersion` Antwort abgerufen werden:

- `TrainingDataResult.Output.Assets[0].GroundTruthManifest.S3Object` für Trainingsdatensatz
- `TestingDataResult.Output.Assets[0].GroundTruthManifest.S3Object` für Testdatensatz

Die folgenden Informationen werden für jeden Eintrag im Ausgabemanifest zurückgegeben:

Schlüsselname	Beschreibung
`source-ref`	Verweis auf ein Bild in S3, das im Eingabema nifest bereitgestellt wurde
`content-moderation-groundtruth`	Ground-Truth-Anmerkungen, die im Eingabema nifest bereitgestellt wurden
`detect-moderation-labels`	Adaptervorhersagen, nur Teil des Testdaten satzes
`detect-moderation-labels-base-model`	Vorhersagen des Basismodells, nur Teil des Testdatensatzes

Adapter- und Basismodellvorhersagen werden mit ConfidenceTrehsold 5,0 in dem Format zurückgegeben, das der DetectModerationLabelsAntwort ähnelt.

Das folgende Beispiel zeigt die Struktur der Adapter- und Basismodellvorhersagen:

```
{
```

```
   "ModerationLabels": [
      {
         "Confidence": number,
         "Name": "string",
         "ParentName": "string"
      }
   ],
   "ModerationModelVersion": "string",
   "ProjectVersion": "string"
}
```

Die vollständige Liste der zurückgegebenen Label finden Sie unter Inhalte moderieren.

Manifest für Evaluierungsergebnisse

Nach Abschluss einer Trainingsaufgabe wird eine Manifestdatei mit den Evaluierungsergebnissen zurückgegeben. Das Manifest der Evaluierungsergebnisse ist eine JSON-Datei, die vom Trainingsauftrag ausgegeben wird und Informationen darüber enthält, wie gut der Adapter bei den Testdaten abgeschnitten hat.

Der Amazon S3 S3-Pfad zum Manifest der Evaluierungsergebnisse kann dem `EvaluationResult.Summary.S3Object` Feld in der DescribeProejctVersion Antwort entnommen werden.

Das folgende Beispiel zeigt die Struktur des Manifests für Evaluierungsergebnisse:

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

Die Evaluierungsmanifestdatei enthält:

- Aggregierte Ergebnisse, wie definiert durch `F1Score`
- Einzelheiten zum Evaluierungsjob ProjectVersionArn, einschließlich der Anzahl der Trainings-Images, der Anzahl der Test-Images und der Labels, auf denen der Adapter trainiert wurde.
- Aggregierte TruePositive, TrueNegative FalsePositive, und FalseNegative Ergebnisse sowohl für das Basismodell als auch für die Adapterleistung.
- Pro Etikett TruePositive, TrueNegative FalsePositive, und FalseNegative Ergebnisse sowohl für das Basismodell als auch für die Adapterleistung, berechnet am Schwellenwert für die Eingabesicherheit.

- Aggregiert und pro Etikett TruePositive, TrueNegative FalsePositive, und FalseNegative Ergebnisse sowohl für das Basismodell als auch für die Adapterleistung bei unterschiedlichen Konfidenzschwellen. Der Zuverlässigkeitsschwellenwert reicht von 5 bis 100 in Schritten von 5 bis 100.

Bewährte Methoden für Trainingsadapter

Es wird empfohlen, dass Sie sich bei Erstellung, Training und Verwendung Ihrer Adapter an die folgenden bewährten Methoden halten:

1. Die Beispielbilddaten sollten die repräsentativen Fehler erfassen, die die Kunden zu unterdrücken beabsichtigen. Wenn das Modell auf visuell ähnlichen Bildern wiederholt Fehler macht, stellen Sie sicher, dass Sie viele dieser Bilder zum Training mitbringen.
2. Anstatt nur Bilder einzufügen, die zeigen, dass das Model bei einem bestimmten Moderationslabel Fehler macht, sollten Sie auch Bilder mit einbeziehen, die zeigen, dass das Model bei diesem Moderationslabel keine Fehler macht.
3. Stellen Sie mindestens 50 falsch negative Proben ODER 20 falsch positive Proben für das Training und mindestens 20 Proben für Tests bereit. Stellen Sie jedoch so viele kommentierte Bilder wie möglich bereit, um die Adapterleistung zu verbessern.
4. Kommentieren Sie alle Labels, die für Sie von Bedeutung sind, für alle Bilder – wenn Sie entscheiden, dass Sie das Vorkommen eines Labels auf einem Bild kommentieren müssen, stellen Sie sicher, dass Sie das Vorkommen dieses Labels auch auf allen anderen Bildern kommentieren.
5. Die Beispielbilddaten sollten so viele Variationen wie möglich auf dem Label enthalten, wobei der Schwerpunkt auf Exemplaren liegt, die repräsentativ für die Bilder sind, die in einer Produktionsumgebung analysiert werden.

AutoUpdateBerechtigungen einrichten

Rekognition unterstützt die AutoUpdate Funktion für benutzerdefinierte Adapter. Das bedeutet, dass automatische Umschulungen nach bestem Wissen und Gewissen durchgeführt werden, wenn die AutoUpdate Markierung für ein Projekt AKTIVIERT ist. Für diese automatischen Updates ist die Erlaubnis zum Zugriff auf Ihre Trainings-/Testdatensätze und den AWS KMS Schlüssel erforderlich, mit dem Sie Ihren Kundenadapter schulen. Sie können diese Berechtigungen erteilen, indem Sie die folgenden Schritte ausführen.

Amazon S3 Bucket-Berechtigungen

Standardmäßig werden alle Amazon-S3-Buckets und -Objekte als privat eingestuft. Nur der Besitzer der Ressource, also das AWS Konto, das den Bucket erstellt hat, kann auf den Bucket und alle darin enthaltenen Objekte zugreifen. Der Ressourcenbesitzer kann jedoch anderen Ressourcen und Benutzern Zugriffsberechtigungen gewähren, indem er eine Bucket-Richtlinie schreibt.

Wenn Sie einen Amazon-S3-Bucket erstellen oder ändern möchten, der als Quelle für Eingabedatensätze und als Ziel für Trainingsergebnisse in einem benutzerdefinierten Adaptertraining verwendet werden soll, müssen Sie die Bucket-Richtlinie weiter ändern. Um aus einem Amazon-S3-Bucket lesen oder in einen Bucket schreiben zu können, muss Rekognition über die folgenden Berechtigungen verfügen.

Amazon-S3-Richtlinie für Rekognition erforderlich

Rekognition erfordert eine Berechtigungsrichtlinie mit den folgenden Attributen:

- Die Anweisungs-ID (SID)
- Den Bucket-Namen
- Den Prinzipal-Namen des Services für Rekognition.
- Die Ressourcen, die für Rekognition, den Bucket und seinen gesamten Inhalt benötigt werden
- Die erforderlichen Maßnahmen, die Rekognition ergreifen muss.

Die folgende Richtlinie erlaubt Rekognition den Zugriff auf einen Amazon-S3-Bucket während eines automatisierten Neutrainings.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "AllowRekognitionAutoUpdateActions",
            "Principal": {
                "Service": "rekognition.amazonaws.com"
            },
            "Action": [
                "s3:ListBucket",
                "s3:GetObject",
                "s3:PutObject",
                "s3:HeadObject",
                "s3:HeadBucket"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

Sie können dieser Anleitung folgen, um die oben genannte Bucket-Richtlinie zu Ihrem S3-Bucket hinzuzufügen.

Weitere Informationen zu Bucket-Richtlinien finden Sie hier.

AWS KMS Wichtige Berechtigungen

Rekognition ermöglicht es Ihnen, KmsKeyId während des Trainings einen optionalen Adapter bereitzustellen. Sofern bereitgestellt, verwendet Rekognition diesen Schlüssel, um Trainings- und Testbilder zu verschlüsseln, die für das Modelltraining in den Service kopiert wurden. Der Schlüssel wird auch verwendet, um Trainingsergebnisse und Manifestdateien zu verschlüsseln, die in den Amazon S3 S3-Ausgabe-Bucket (OutputConfig) geschrieben werden.

Wenn Sie sich dafür entscheiden, einen KMS-Schlüssel als Eingabe für Ihr benutzerdefiniertes Adaptertraining (d. h. `Rekognition:CreateProjectVersion`) bereitzustellen, müssen Sie die KMS-Schlüsselrichtlinie weiter ändern, damit der Rekognition-Service-Prinzipal diesen Schlüssel in Zukunft für automatisiertes Neutraining verwenden kann. Rekognition muss über die folgenden Berechtigungen verfügen.

Schlüsselrichtlinie „Rekognition erforderlich AWS KMS “

Amazon Rekognition erfordert eine Berechtigungsrichtlinie mit den folgenden Attributen:

- Die Anweisungs-ID (SID)
- Den Prinzipal-Namen des Services für Amazon Rekognition.
- Die erforderlichen Maßnahmen, die Amazon Rekognition ergreifen muss.

Die folgende wichtige Richtlinie ermöglicht Amazon Rekognition den Zugriff auf einen Amazon-KMS-Schlüssel während eines automatisierten Neutrainings:

```
{
```

```
  "Version": "2023-10-06",
  "Statement": [
    {
      "Sid": "KeyPermissions",
      "Effect": "Allow",
      "Principal": {
            "Service": "rekognition.amazonaws.com"
        },
      "Action": [
        "kms:DescribeKey",
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ]
      "Resource": "*"
    }
  ]
}
```

Sie können dieser Anleitung folgen, um die oben genannte AWS KMS Richtlinie zu Ihrem AWS KMS Schlüssel hinzuzufügen.

Weitere Informationen zu den AWS KMS Richtlinien finden Sie hier.

AWS Health Dashboard-Benachrichtigung für Rekognition

Ihr AWS Health Dashboard bietet Unterstützung für Benachrichtigungen, die von Rekognition stammen. Diese Benachrichtigungen informieren Sie über geplante Änderungen an Rekognition-Modellen, die sich auf Ihre Anwendungen auswirken können, und geben Hinweise zur Behebung dieser Probleme. Derzeit sind nur Ereignisse verfügbar, die für das Rekognition-Inhaltsmoderations-Feature spezifisch sind.

Das AWS Health Dashboard ist Teil des AWS Gesundheitsdienstes. Sie benötigt keine Einrichtung und kann von jedem Benutzer angezeigt werden, der in Ihrem Konto authentifiziert ist. Weitere Informationen finden Sie unter Erste Schritte mit dem AWS-Servicestatus-Dashboard.

Wenn Sie eine Nachricht ähnlich den folgenden erhalten, sollte sie als Alarm behandelt werden, um Maßnahmen zu ergreifen.

Beispielbenachrichtigung: Für Rekognition Content Moderation ist eine neue Modellversion verfügbar.

Rekognition veröffentlicht das `AWS_MODERATION_MODEL_VERSION_UPDATE_NOTIFICATION` Ereignis im AWS Health Dashboard, um darauf hinzuweisen, dass eine neue Version des

Moderationsmodells veröffentlicht wurde. Dieses Ereignis ist wichtig, wenn Sie die API und Adapter mit dieser DetectModerationLabels API verwenden. Neue Modelle können sich je nach Anwendungsfall auf die Qualität auswirken und werden irgendwann frühere Modellversionen ersetzen. Es wird empfohlen, die Qualität Ihres Modells zu überprüfen und die Zeitpläne für Modellaktualisierungen zu beachten, wenn Sie diese Warnung erhalten.

Wenn Sie eine Benachrichtigung über ein Update der Modellversion erhalten, sollten Sie dies als Warnung betrachten, um Maßnahmen zu ergreifen. Wenn Sie keine Adapter verwenden, sollten Sie die Qualität des aktualisierten Modells anhand Ihres bestehenden Anwendungsfalls bewerten. Wenn Sie Adapter verwenden, sollten Sie neue Adapter mit dem aktualisierten Modell trainieren und deren Qualität bewerten. Wenn Sie auto-train eingestellt haben, werden neue Adapter automatisch trainiert, und dann können Sie ihre Qualität bewerten.


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

```
    }
}
```

Informationen zur Erkennung und Reaktion auf Integritätsereignisse mithilfe von Amazon finden EventBridge Sie unter Überwachung von AWS AWS-Gesundheitsereignissen mit Amazon EventBridge.

Überprüfen unangemessener Inhalte mit Amazon Augmented AI

Mit Amazon Augmented AI (Amazon A2I) können Sie Workflows erstellen, die für die menschliche Überprüfung von Machine-Learning-Vorhersagen erforderlich sind.

Amazon Rekognition ist direkt in Amazon A2I integriert, so dass Sie die menschliche Überprüfung für den Anwendungsfall der Erkennung unsicherer Bilder einfach implementieren können. Amazon A2I bietet einen Workflow für die menschliche Überprüfung zur Bildmoderation. Auf diese Weise können Sie die Vorhersagen von Amazon Rekognition ganz einfach überprüfen. Sie können Zuverlässigkeitsschwellenwerte für Ihren Anwendungsfall definieren und diese im Laufe der Zeit anpassen. Mit Amazon A2I können Sie einen Pool von Prüfern innerhalb Ihrer eigenen Organisation oder Amazon Mechanical Turk verwenden. Sie können außerdem Personaldienstleister einsetzen, die von AWS auf Qualität und Einhaltung der Sicherheitsverfahren vorab geprüft werden.

In den folgenden Schritten erfahren Sie, wie Sie Amazon A2I mit Amazon Rekognition einrichten. Zuerst erstellen Sie eine Flow-Definition mit Amazon A2I, die die Bedingungen aufweist, die menschliche Überprüfungen auslösen. Anschließend übergeben Sie den Amazon-Ressourcennamen (ARN) der Flow-Definition an die Amazon-Rekognition-`DetectModerationLabel`-Operation. In der `DetectModerationLabel`-Antwort können Sie sehen, ob menschliche Überprüfung erforderlich ist. Die Ergebnisse der menschlichen Überprüfung sind in einem Amazon-S3-Bucket verfügbar, der durch die Flow-Definition festgelegt wird.

Eine end-to-end Demonstration der Verwendung von Amazon A2I mit Amazon Rekognition finden Sie in einem der folgenden Tutorials im Amazon SageMaker AI Developer Guide.

- Demo: Erste Schritte in der Amazon-A2I-Konsole
- Demo: Erste Schritte mit der Amazon-A2I-API

  Um mit der Verwendung der API zu beginnen, können Sie auch ein Beispiel-Jupyter-Notebook ausführen. Siehe Verwenden einer SageMaker Notebook-Instance mit Amazon A2I Jupyter

Notebook, um die Amazon Augmented AI (Amazon A2I) -Integration mit Amazon Rekognition [Beispiel] in einer KI-Notebook-Instance zu verwenden. SageMaker

Betrieb DetectModerationLabels mit Amazon A2I

> **Note**
>
> Erstellen Sie alle Ressourcen für Amazon A2I und Amazon Rekognition in derselben AWS-Region.

1. Erfüllen Sie die Voraussetzungen, die unter Erste Schritte mit Amazon Augmented AI in der SageMaker KI-Dokumentation aufgeführt sind.

   Denken Sie außerdem daran, Ihre IAM-Berechtigungen wie auf der Seite Berechtigungen und Sicherheit in Amazon Augmented AI in der SageMaker KI-Dokumentation einzurichten.
2. Folgen Sie den Anweisungen zur Erstellung eines Workflows zur Überprüfung durch Mitarbeiter in der SageMaker KI-Dokumentation.

   Ein Workflow für die menschliche Überprüfung verwaltet die Verarbeitung eines Bildes. Er enthält die Bedingungen, die eine menschliche Überprüfung auslösen, das Arbeitsteam, an das das Bild gesendet wird, die Benutzeroberflächenvorlage, die das Arbeitsteam verwendet, und den Amazon-S3-Bucket, an den die Ergebnisse des Arbeitsteams gesendet werden.

   Innerhalb Ihres `CreateFlowDefinition` Anrufs müssen Sie den Wert `HumanLoopRequestSource` auf "AWS/Rekognition/DetectModerationLabels/Image/V3" setzen. Danach müssen Sie entscheiden, wie Sie Ihre Bedingungen einrichten möchten, die eine menschliche Überprüfung auslösen.

   Mit Amazon Rekognition haben Sie zwei Optionen für `ConditionType`: `ModerationLabelConfidenceCheck` und `Sampling`.

   `ModerationLabelConfidenceCheck` erstellt eine Schleife zur Prüfung durch Menschen (Human Loop), wenn die Zuverlässigkeit eines Moderations-Labels innerhalb eines bestimmten Bereichs liegt. Schließlich sendet `Sampling` einen willkürlichen Prozentsatz der verarbeiteten Dokumente an die Prüfung durch Menschen. Jeder `ConditionType` legt mittels eines anderen Satzes von `ConditionParameters` fest, welche Ergebnisse zu einer Prüfung durch Menschen führen.

Für `ModerationLabelConfidenceCheck` gilt `ConditionParameters` `ModerationLableName`, wodurch der Schlüssel festlegt wird, bei dem eine Prüfung durch Menschen erforderlich ist. Darüber hinaus gibt es Konfidenzwerte, die den prozentualen Bereich für das Senden an eine menschliche Überprüfung mit LessThan GreaterThan, und Entspricht festlegen. `SamplinghatRandomSamplingPercentage`, wodurch ein Prozentsatz der Dokumente festgelegt wird, die zur Überprüfung durch einen Mitarbeiter gesendet werden.

Das folgende Codebeispiel ist ein Teilaufruf von `CreateFlowDefinition`. Es sendet ein Bild zur menschlichen Überprüfung, wenn es weniger als 98 % auf dem Label „Suggestive“ und mehr als 95 % auf dem Label „Female Swimwear or Underwear“ angibt. Dies bedeutet, dass das Bild nicht als unbedingt suggestiv betrachtet wird, aber eine Frau in Unterwäsche oder Bademode zeigt; Sie können das Bild überprüfen, indem Sie menschliche Überprüfung verwenden.

```
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` gibt eine `FlowDefinitionArn` zurück, die Sie im nächsten Schritt beim Aufruf von `DetectModerationLabels` verwenden.

Weitere Informationen finden Sie CreateFlowDefinitionin der SageMaker AI-API-Referenz.

3. Legen Sie beim Aufruf von `DetectModerationLabels` den Parameter `HumanLoopConfig` wie in Erkennen unangemessener Bilder gezeigt fest. In Schritt 4 finden Sie Beispiele für einen `DetectModerationLabels`-Aufruf mit `HumanLoopConfig`-Set.

   a. Legen Sie innerhalb des `HumanLoopConfig`-Parameters den `FlowDefinitionArn` auf den ARN der Flow-Definition fest, die Sie in Schritt 2 erstellt haben.

   b. Einstellen Ihrer `HumanLoopName`. Dies sollte innerhalb einer Region eindeutig sein und muss aus Kleinbuchstaben bestehen.

   c. (Optional) Sie können mit `DataAttributes` festlegen, ob das Bild, das Sie an Amazon Rekognition weitergegeben haben, frei von persönlich identifizierbaren Informationen (PII) ist. Sie müssen diesen Parameter festlegen, um Informationen an Amazon Mechanical Turk zu senden.

4. Führen Sie `DetectModerationLabels`.

Die folgenden Beispiele zeigen, wie Sie das AWS CLI und AWS SDK für Python (Boto3) zum Ausführen `DetectModerationLabels` mit `HumanLoopConfig` Set verwenden.

AWS SDK für Python (Boto3)

Die folgenden Beispiele verwenden den SDK for Python (Boto3). Weitere Informationen finden Sie unter detect_moderation_Label im AWS -SDK für die API-Referenz 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

Das folgende Anforderungsbeispiel verwendet die AWS CLI. Weitere Informationen finden Sie unter detect-moderation-labels in der Referenz zum AWS CLI -Befehl.

```
$ 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"]}}'
```

Wenn Sie `DetectModerationLabels` mit `HumanLoopConfig` aktivierter Option ausführen, ruft Amazon Rekognition den SageMaker KI-API-Vorgang auf. `StartHumanLoop` Dieser Befehl nimmt die Antwort von `DetectModerationLabels` und überprüft sie anhand der Bedingungen der Ablaufdefinition im Beispiel. Wenn es die Bedingungen für die Überprüfung erfüllt, wird ein `HumanLoopArn` zurückgegeben. Dies bedeutet, dass die Mitglieder des Arbeitsteams, die Sie in der Flow-Definition festgelegt haben, das Bild jetzt überprüfen können. Der Aufruf der Amazon Augmented AI-Laufzeitoperation `DescribeHumanLoop` stellt Informationen zum Ergebnis der Schleife bereit. Weitere Informationen finden Sie DescribeHumanLoopin der Referenzdokumentation zur Amazon Augmented AI API.

Nachdem das Bild überprüft wurde, können Sie die Ergebnisse in dem Bucket sehen, der im Ausgabepfad der Flow-Definition angegeben ist. Amazon A2I benachrichtigt Sie auch über Amazon CloudWatch Events, wenn die Überprüfung abgeschlossen ist. Informationen dazu, nach welchen Ereignissen Sie suchen müssen, finden Sie in der SageMaker KI-Dokumentation unter CloudWatch Ereignisse.

Weitere Informationen finden Sie unter Erste Schritte mit Amazon Augmented AI in der SageMaker KI-Dokumentation.

Erkennen von Text

Amazon Rekognition kann Text in Bildern und Videos erkennen. Sie kann anschließend den erkannten Text in maschinenlesbaren Text umwandeln. Sie können die maschinenlesbare Texterkennung in Bildern verwenden, um Lösungen zu implementieren wie:

- Visuelle Suche. Beispielsweise können Sie Bilder abrufen und anzeigen, die denselben Text enthalten.
- Content-Einblicke. Ein Beispiel ist die Bereitstellung von Einblicken in Themen, die in Text vorkommen, der in extrahierten Videobildern erkannt wird. Ihre Anwendung kann erkannten Text nach relevanten Inhalten wie Nachrichten, Sportergebnissen, Athletennummern und Bildunterschriften durchsuchen.
- Navigation. Sie könnten zum Beispiel eine sprachgesteuerte, mobile App für sehbehinderte Menschen entwickeln, die Namen von Restaurants, Geschäften oder Straßenschildern erkennt.
- Unterstützung der öffentlichen Sicherheit und des Verkehrs. Zum Beispiel die Erkennung von Kfz-Kennzeichen aus Verkehrskamerabildern.
- Filtern. Zum Beispiel, Herausfiltern von persönlich identifizierbaren Informationen (PII) aus Bildern.

Für die Texterkennung in Videos können Sie Lösungen implementieren wie:

- Suchen von Videos nach Clips, in denen bestimmte Textschlüsselwörter, z. B. der Name des Gastes auf einer Grafik in einer Nachrichtensendung, vorkommen
- Moderation von Inhalten zur Einhaltung organisatorischer Standards durch Erkennung von versehentlichem Text, Schimpfwörtern oder Spam
- Suchen aller Texteinblendungen auf der Videozeitleiste zur weiteren Verarbeitung, z. B. Ersetzen von Text durch Text in einer anderen Sprache für die Internationalisierung von Inhalten
- Finden von Textstellen, damit andere Grafiken entsprechend angepasst werden können

Verwenden Sie den DetectTextVorgang, um Text in Bildern im JPEG- oder PNG-Format zu erkennen. Verwenden Sie die GetTextDetectionOperationen StartTextDetectionund, um Text in Videos asynchron zu erkennen. Sowohl Bild- als auch Videotexterkennungs-Operationen unterstützen die meisten Schriftarten, dazu zählen auch hochstilisierte Schriftarten. Wenn Text erkannt wird, erzeugt Amazon Rekognition eine Darstellung der erkannten Wörter und Textzeilen,

zeigt die Beziehung zwischen ihnen an und zeigt Ihnen an, wo sich der Text auf einem Bild oder einem Video-Frame befindet.

Die `DetectText`- und `GetTextDetection`-Operationen erkennen Wörter und Zeilen. Ein Wort besteht aus einem oder mehreren Skriptzeichen, die nicht durch Leerzeichen getrennt sind. `DetectText` kann bis zu 100 Wörter in einem Bild erkennen. `GetTextDetection` kann auch bis zu 100 Wörter pro Video-Frame erkennen.

Ein Wort besteht aus einem oder mehreren Textzeichen, die nicht durch Leerzeichen getrennt sind. Amazon Rekognition wurde entwickelt, um Text in den Sprachen Englisch, Arabisch, Russisch, Deutsch, Französisch, Italienisch, Portugiesisch und Spanisch zu erkennen.

Eine Zeile besteht aus einer Folge von Wörtern mit gleichem Abstand. Eine Zeile ist nicht unbedingt ein vollständiger Satz (Punkte bedeuten nicht das Ende einer Zeile). Amazon Rekognition erkennt zum Beispiel die Nummer eines Führerscheins als Zeile. Eine Zeile endet, wenn hinter ihr kein ausgerichteter Text steht oder wenn zwischen den Wörtern ein großer Abstand besteht, der im Verhältnis zur Länge der Wörter besteht. Je nach Abstand zwischen den Wörtern erkennt Amazon Rekognition möglicherweise mehrere Textzeilen, die in dieselbe Richtung ausgerichtet sind. Wenn ein Satz über mehrere Zeilen geht, gibt die Operation mehrere Zeilen zurück.

Betrachten Sie das folgende Bild:

Die blauen Felder enthalten Informationen über den erkannten Text und die Position des Textes, der durch die `DetectText`-Operation zurückgegeben wird. In diesem Beispiel erkennt Amazon Rekognition „IT's“, „MONDAY“, „but“, „keep“, und „Smiling“ als Wörter. Amazon Rekognition erkennt „IT'S“, „MONDAY“, „but keep“ und „Smiling“ als Zeilen. Um erkannt zu werden, muss sich der Text innerhalb von +/- 90 Grad in der horizontalen Achse befinden.

Ein Beispiel finden Sie unter Erkennen von Text in einem Bild.

Themen

- Erkennen von Text in einem Bild
- Erkennen von Text in einem gespeicherten Video

Erkennen von Text in einem Bild

Sie können ein Eingabebild als Bild-Byte-Array (base64-verschlüsselte Bild-Bytes) oder als Amazon-S3-Objekt zur Verfügung stellen. Bei dieser Vorgehensweise laden Sie ein JPEG- oder PNG-Bild in Ihren S3-Bucket hoch und geben den Dateinamen an.

Um Text in einem Bild zu erkennen (API)

1. Falls Sie dies noch nicht getan haben, müssen Sie die folgenden Voraussetzungen erfüllen.

   a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`- und `AmazonS3ReadOnlyAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

   b. Installieren und konfigurieren Sie das AWS Command Line Interface und das. AWS SDKs Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Laden Sie das Bild, das Text enthält, in Ihren S3-Bucket hoch.

   Weitere Anleitungen finden Sie unter Upload eines Objekts in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service.

3. Verwenden Sie die folgenden Beispiele zum Aufrufen der `DetectText`-Operation.

   Java

   Der folgende Beispielcode zeigt Zeilen und Wörter an, die in einem Bild erkannt wurden.

Ersetzen Sie die Werte von `amzn-s3-demo-bucket` und `photo` durch die Namen des S3-Buckets und des Bildes, das Sie in Schritt 2 verwendet haben.

```
//Copyright 2018 Amazon.com, Inc. or its 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

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
/**
*  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

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `detect-text` CLI-Vorgang an.

Ersetzen Sie die Werte von `amzn-s3-demo-bucket` und `Name` durch die Namen des S3-Buckets und des Bildes, das Sie in Schritt 2 verwendet haben.

Ersetzen Sie den Wert von `profile_name` mit dem Namen Ihres Entwicklerprofils.

```
aws rekognition detect-text  --image "{"S3Object":{"Bucket":"amzn-s3-demo-
bucket","Name":"image-name"}}" --profile default
```

Wenn Sie auf einem Windows-Gerät auf die CLI zugreifen, verwenden Sie doppelte Anführungszeichen anstelle von einfachen Anführungszeichen und maskieren Sie die inneren doppelten Anführungszeichen durch einen Backslash (d. h. \), um eventuell auftretende Parserfehler zu beheben. Sehen Sie sich zum Beispiel Folgendes an:

```
aws rekognition detect-text  --image "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --profile default
```

Python

Der folgende Beispielcode zeigt erkannte Zeilen und Wörter an, die in einem Bild erkannt wurden.

Ersetzen Sie die Werte von `amzn-s3-demo-bucket` und `photo` durch die Namen des S3-Buckets und des Bildes, das Sie in Schritt 2 verwendet haben. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

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

import boto3

def detect_text(photo, bucket):

    session = boto3.Session(profile_name='default')
    client = session.client('rekognition')

    response = client.detect_text(Image={'S3Object': {'Bucket': bucket, 'Name':
 photo}})
```

```
    textDetections = response['TextDetections']
    print('Detected text\n----------')
    for text in textDetections:
        print('Detected text:' + text['DetectedText'])
        print('Confidence: ' + "{:.2f}".format(text['Confidence']) + "%")
        print('Id: {}'.format(text['Id']))
        if 'ParentId' in text:
            print('Parent Id: {}'.format(text['ParentId']))
        print('Type:' + text['Type'])
        print()
    return len(textDetections)

def main():
    bucket = 'amzn-s3-demo-bucket'
    photo = 'photo-name'
    text_count = detect_text(photo, bucket)
    print("Text detected: " + str(text_count))

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

.NET

Der folgende Beispielcode zeigt erkannte Zeilen und Wörter an, die in einem Bild erkannt wurden.

Ersetzen Sie die Werte von `amzn-s3-demo-bucket` und `photo` durch die Namen des S3-Buckets und des Bildes, das Sie in Schritt 2 verwendet haben.

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

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

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

```
        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        DetectTextRequest detectTextRequest = new DetectTextRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket
                }
            }
        };

        try
        {
            DetectTextResponse detectTextResponse =
 rekognitionClient.DetectText(detectTextRequest);
            Console.WriteLine("Detected lines and words for " + photo);
            foreach (TextDetection text in detectTextResponse.TextDetections)
            {
                Console.WriteLine("Detected: " + text.DetectedText);
                Console.WriteLine("Confidence: " + text.Confidence);
                Console.WriteLine("Id : " + text.Id);
                Console.WriteLine("Parent Id: " + text.ParentId);
                Console.WriteLine("Type: " + text.Type);
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```

Node.JS

Der folgende Beispielcode zeigt erkannte Zeilen und Wörter an, die in einem Bild erkannt wurden.

Ersetzen Sie die Werte von `amzn-s3-demo-bucket` und `photo` durch die Namen des S3-Buckets und des Bildes, das Sie in Schritt 2 verwendet haben. Ersetzen Sie den Wert von `region` durch die Region, die Sie in Ihren AWS-Anmeldeinformationen finden. Ersetzen Sie den Wert von `profile_name` in der Zeile, die die Rekognition-Sitzung erstellt, durch den Namen Ihres Entwicklerprofils.

```
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 Operationsanforderung

In der `DetectText`-Operation stellen Sie ein Eingabebild bereit, entweder als base64-codiertes Byte-Array oder als Bild, das in einem Amazon-S3-Bucket gespeichert ist. Das folgende Beispiel einer JSON-Anforderung zeigt das aus einem Amazon-S3-Bucket geladene Bild.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "inputtext.jpg"
        }
    }
}
```

Filter

Die Filterung nach Textbereich, Größe und Zuverlässigkeitswert bietet Ihnen zusätzliche Flexibilität bei der Steuerung Ihrer Texterkennungsausgabe. Durch die Verwendung von „Bereiche von Interesse“ können Sie die Texterkennung bequem auf die für Sie relevanten Bereiche beschränken, z. B. auf die obere rechte Ecke eines Profilfotos oder auf eine feste Position in Bezug auf einen Referenzpunkt beim Lesen von Teilenummern aus einem Bild einer Maschine. Mithilfe eines Filters für die Größe von Wortbegrenzungsrahmen kann kleiner Hintergrundtext vermieden werden, der u. U. stört oder irrelevant ist. Der Wortzuverlässigkeitsfilter ermöglicht Ihnen das Entfernen von Ergebnissen, die aufgrund von Unschärfe möglicherweise unzuverlässig sind.

Hinweise zu Filterwerten finden Sie unter `DetectTextFilters`.

Sie können die folgenden Filter verwenden:

- MinConfidence— Legt das Konfidenzniveau der Worterkennung fest. Wörter mit einer Erkennungszuverlässigkeit unterhalb dieses Niveaus werden vom Ergebnis ausgeschlossen. Die Werte sollten zwischen 0 und 100 liegen.
- MinBoundingBoxWidth— Legt die Mindestbreite des Begrenzungsrahmens für Wörter fest. Wörter mit Begrenzungsrahmen, die kleiner als dieser Wert sind, werden aus dem Ergebnis ausgeschlossen. Der Wert ist relativ zur Bildbreite.

- MinBoundingBoxHeight— Legt die Mindesthöhe des Begrenzungsrahmens für Wörter fest. Wörter mit einer Begrenzungsrahmenhöhe unter diesem Wert werden aus dem Ergebnis ausgeschlossen. Der Wert ist relativ zur Bildhöhe.
- RegionsOfInterest— Beschränkt die Erkennung auf einen bestimmten Bereich des Bildrahmens. Die Werte sind relativ zur Bildgröße. Bei Text, der nur teilweise innerhalb eines Bereichs liegt, ist die Antwort undefiniert.

DetectText Reaktion auf den Betrieb

Die `DetectText` Operation analysiert das Bild und gibt ein Array zurück TextDetections, wobei jedes Element (`TextDetection`) für eine Zeile oder ein Wort steht, das im Bild erkannt wurde. Für jedes Element gibt `DetectText` die folgenden Informationen zurück:

- Den erkannten Text (`DetectedText`)
- Die Beziehungen zwischen Wörtern und Zeilen (`Id` und `ParentId`)
- Die Position des Textes auf dem Bild (`Geometry`)
- Das Vertrauen von Amazon Rekognition in die Genauigkeit des erkannten Textes und des Begrenzungsrahmens (`Confidence`)
- Die Art des erkannten Textes (`Type`)

Erkannter Text

Jedes `TextDetection`-Element enthält erkannten Text (Wörter oder Zeilen) im Feld `DetectedText`. Ein Wort besteht aus einem oder mehreren Schriftzeichen, die nicht durch Leerzeichen getrennt sind. `DetectText` kann bis zu 100 Wörter in einem Bild erkennen. Der zurückgegebene Text kann Zeichen enthalten, die ein Wort unkenntlich machen. Beispiel: C@t anstelle von Cat. Um festzustellen, ob ein Element `TextDetection` eine Textzeile oder ein Wort repräsentiert, verwenden Sie das Feld `Type`.

Jedes `TextDetection`-Element enthält einen Prozentwert, der den Grad des Vertrauens repräsentiert, den Amazon Rekognition in die Genauigkeit des erkannten Texts und des den Text umgebenden Begrenzungsrahmens hat.

Wort- und Zeilen-Beziehungen

Jedes `TextDetection`-Element verfügt über ein ID-Feld, `Id`. Die `Id` zeigt die Position des Wortes in einer Zeile an. Wenn es sich einem Element um ein Wort handelt, identifiziert das Feld der übergeordneten ID, `ParentId`, die Zeile, in der das Wort erkannt wurde. Die `ParentId` für eine Zeile ist Null. Zum Beispiel hat die Zeile „but keep“ im Beispielbild folgende `Id`- und `ParentId`-Werte:

Text	ID	Parent ID
but keep	3	
but	8	3
keep	9	3

Textposition auf einem Bild

Um zu ermitteln, wo sich der erkannte Text auf einem Bild befindet, verwenden Sie die Informationen des Begrenzungsrahmens (Geometrie), den `DetectText` zurückgibt. Das `Geometry`-Objekt enthält zwei Arten von Informationen über Begrenzungsrahmen für erkannte Zeilen und Wörter:

- Ein achsenausgerichteter grober rechteckiger Umriss in einem BoundingBox-Objekt
- Ein feingliedrigeres Polygon, bestehend aus mehreren X- und Y-Koordinaten in einem Punkt-Array

Der Begrenzungsrahmen und die Koordinaten des Polygons zeigen die Position des Textes auf dem Quellbild an. Die Koordinatenwerte sind ein Verhältnis der gesamten Bildgröße. Weitere Informationen finden Sie unter BoundingBox.

Die folgende JSON-Antwort der `DetectText`-Operation zeigt die erkannten Wörter und Zeilen im nachfolgenden Bild.

```
{
 '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'}
```

Erkennen von Text in einem gespeicherten Video

Die Texterkennung von Amazon Rekognition Video in gespeicherten Videos ist eine asynchrone Operation. Rufen StartTextDetectionSie an, um mit der Texterkennung zu beginnen. Amazon

Rekognition Video veröffentlicht den Fertigstellungsfortschritt der Videoanalyse in einem Amazon-SNS-Thema. Wenn die Videoanalyse erfolgreich ist, rufen Sie GetTextDetection auf, um die Analyseergebnisse abzurufen. Weitere Informationen zum Starten der Videoanalyse und zum Abrufen der Ergebnisse finden Sie unter Amazon-Rekognition-Video-Operationen aufrufen.

Durch dieses Verfahren wird der Code in Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) erweitert. Bei dem Verfahren wird eine Amazon-SQS-Warteschlange verwendet, um den Abschlussstatus einer Videoanalyseanforderung zu erhalten.

So erkennen Sie Text in einem Video, das in einem Amazon-S3-Bucket gespeichert ist (SDK)

1. Führen Sie die Schritte unter Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) aus.
2. Fügen Sie in Schritt 1 den folgenden Code zur Klasse `VideoDetect` hinzu.

   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);


}
```

Ersetzen Sie in der Funktion `main` die folgenden Zeilen:

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

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

mit:

```
        StartTextDetection(amzn-s3-demo-bucket, video);

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

Java V2

Dieser Code stammt aus dem AWS Documentation SDK Examples GitHub Repository. Das vollständige Beispiel finden Sie hier.

```
//snippet-start:[rekognition.java2.recognize_video_text.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
```

```
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;
//snippet-end:[rekognition.java2.recognize_video_text.import]

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

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

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

     if (args.length != 4) {
         System.out.println(usage);
         System.exit(1);
     }

     String bucket = args[0];
```

```
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];

    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
        .build();

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

    startTextLabels(rekClient, channel, bucket, video);
    GetTextResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

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

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

        StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidOb)
            .build();
```

```
        StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

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

public static void GetTextResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetTextDetectionResponse textDetectionResponse=null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (textDetectionResponse !=null)
                paginationToken = textDetectionResponse.nextToken();

            GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                status = textDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }
```

```
            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData=textDetectionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());
            System.out.println("Job");

            List<TextDetectionResult> labels=
 textDetectionResponse.textDetections();
            for (TextDetectionResult detectedText: labels) {
                System.out.println("Confidence: " +
 detectedText.textDetection().confidence().toString());
                System.out.println("Id : " +
 detectedText.textDetection().id());
                System.out.println("Parent Id: " +
 detectedText.textDetection().parentId());
                System.out.println("Type: " +
 detectedText.textDetection().type());
                System.out.println("Text: " +
 detectedText.textDetection().detectedText());
                System.out.println();
            }

        } while (textDetectionResponse !=null &&
 textDetectionResponse.nextToken() != null);

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

Python

```
#Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)
```

```
    def StartTextDetection(self):
        response=self.rek.start_text_detection(Video={'S3Object': {'Bucket':
 self.bucket, 'Name': self.video}},
            NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
 self.snsTopicArn})

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

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

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

            print('Codec: ' + response['VideoMetadata']['Codec'])

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

            for textDetection in response['TextDetections']:
                text=textDetection['TextDetection']

                print("Timestamp: " + str(textDetection['Timestamp']))
                print("   Text Detected: " + text['DetectedText'])
                print("   Confidence: " +  str(text['Confidence']))
                print ("      Bounding box")
                print ("        Top: " + str(text['Geometry']['BoundingBox']
['Top']))
                print ("        Left: " + str(text['Geometry']['BoundingBox']
['Left']))
                print ("        Width: " +  str(text['Geometry']['BoundingBox']
['Width']))
                print ("        Height: " +  str(text['Geometry']['BoundingBox']
['Height']))
                print ("   Type: " + str(text['Type']) )
```

```
                print()

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

Ersetzen Sie in der Funktion `main` die folgenden Zeilen:

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

mit:

```
    analyzer.StartTextDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetTextDetectionResults()
```

CLI

Führen Sie den folgenden AWS CLI Befehl aus, um mit der Texterkennung in einem Video zu beginnen.

```
 aws rekognition start-text-detection --video "{"S3Object":{"Bucket":"amzn-s3-
demo-bucket","Name":"video-name"}}"\
 --notification-channel "{"SNSTopicArn":"topic-arn","RoleArn":"role-arn"}" \
 --region region-name --profile profile-name
```

Aktualisieren Sie die folgenden Werte:

- Ändern Sie `amzn-s3-demo-bucket` und `video-name` in den Amazon-S3-Bucket-Namen und den Dateinamen, die Sie in Schritt 2 angegeben haben.
- Ändern Sie `region-name` in die von Ihnen verwendete AWS-Region.
- Ersetzen Sie den Wert von `profile-name` mit dem Namen Ihres Entwicklerprofils.
- Ändern Sie `topic-ARN` in den ARN des Amazon-SNS-Themas, das Sie in Schritt 3 von Amazon Rekognition Video konfigurieren erstellt haben.
- Ändern Sie `role-ARN` in den ARN der IAM-Servicerolle, die Sie in Schritt 7 von Amazon Rekognition Video konfigurieren erstellt haben.

Wenn Sie auf einem Windows-Gerät auf die CLI zugreifen, verwenden Sie doppelte Anführungszeichen anstelle von einfachen Anführungszeichen und maskieren Sie die inneren doppelten Anführungszeichen durch einen Backslash (d. h. \), um eventuell auftretende Parserfehler zu beheben. Ein Beispiel finden Sie unten:

```
aws rekognition start-text-detection --video \
 "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-bucket\",\"Name\":\"video-name\"}}" \
 --notification-channel "{\"SNSTopicArn\":\"topic-arn\",\"RoleArn\":\"role-arn
\"}" \
 --region region-name --profile profile-name
```

Nachdem Sie das vorangegangene Codebeispiel ausgeführt haben, kopieren Sie die zurückgegebene `jobID` und geben Sie sie an den folgenden `GetTextDetection`-Befehl weiter, um Ihre Ergebnisse zu erhalten, und ersetzen Sie `job-id-number` durch `jobID`, die Sie zuvor erhalten haben:

```
aws rekognition get-text-detection --job-id job-id-number --profile profile-name
```

> **Note**
>
> Wenn Sie zusätzlich zu Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) bereits ein anderes Videobeispiel ausgeführt haben, ist der zu ersetzende Code möglicherweise anders.

3. Führen Sie den Code aus. Text, der im Video erkannt wurde, wird in einer Liste angezeigt.

Filter

Filter sind optionale Anforderungsparameter, die beim Aufruf von `StartTextDetection` verwendet werden können. Die Filterung nach Textbereich, Größe und Zuverlässigkeitswert bietet Ihnen zusätzliche Flexibilität bei der Steuerung Ihrer Texterkennungsausgabe. Durch die Verwendung von „Regions of Interest“ (Bereiche von Interesse) können Sie die Texterkennung bequem auf die Bereiche beschränken, die relevant sind, z. B. auf einen Bereich im unteren Drittel für Grafiken oder eine Ecke links oben zum Lesen von Anzeigetafeln in einem Fußballspiel. Mithilfe eines Filters für die

Größe von Wortbegrenzungsrahmen kann kleiner Hintergrundtext vermieden werden, der u. U. stört oder irrelevant ist. Und schließlich ermöglicht Ihnen der Wortzuverlässigkeitsfilter das Entfernen von Ergebnissen, die aufgrund von Unschärfe möglicherweise unzuverlässig sind.

Hinweise zu Filterwerten finden Sie unter `DetectTextFilters`.

Sie können die folgenden Filter verwenden:

- MinConfidence— Legt das Konfidenzniveau der Worterkennung fest. Wörter mit einer Erkennungszuverlässigkeit unterhalb dieses Niveaus werden vom Ergebnis ausgeschlossen. Die Werte sollten zwischen 0 und 100 liegen.
- MinBoundingBoxWidth— Legt die Mindestbreite des Begrenzungsrahmens für Wörter fest. Wörter mit Begrenzungsrahmen, die kleiner als dieser Wert sind, werden aus dem Ergebnis ausgeschlossen. Der Wert ist relativ zur Videobildbreite.
- MinBoundingBoxHeight— Legt die Mindesthöhe des Begrenzungsrahmens für Wörter fest. Wörter mit einer Begrenzungsrahmenhöhe unter diesem Wert werden aus dem Ergebnis ausgeschlossen. Der Wert ist relativ zur Videobildhöhe.
- RegionsOfInterest— Beschränkt die Erkennung auf einen bestimmten Bereich des Frames. Die Werte sind relativ zur Bildgröße. Bei Objekten, die nur teilweise innerhalb der Bereiche liegen, ist die Antwort undefiniert.

GetTextDetection — Antwort

`GetTextDetection` gibt ein Array (`TextDetectionResults`) zurück, das Informationen über den erkannten Text im Video enthält. Das Array -Element TextDetection wird jedes Mal erzeugt, wenn ein Wort oder eine Zeile im Video erkannt wird. Die Array-Elemente werden nach Zeit (in Millisekunden) ab dem Start des Videos sortiert.

Folgendes ist ein Teil einer JSON-Antwort von `GetTextDetection`. In der Antwort ist Folgendes zu beachten:

- Textinformationen — Das `TextDetectionResult` Array-Element enthält Informationen über den erkannten Text (TextDetection) und die Uhrzeit, zu der der Text im Video erkannt wurde (`Timestamp`).
- Seiteninformationen – Das Beispiel zeigt eine Seite mit Informationen der Texterkennung. Sie können im `MaxResults`-Eingabeparameter für `GetTextDetection` angeben, wie viele Textelemente zurückgegeben werden sollen. Wenn mehr Ergebnisse als `MaxResults` vorhanden

sind oder es mehr Ergebnisse als das Standardmaximum gibt, gibt `GetTextDetection` ein Token (`NextToken`) zurück, das verwendet wird, um die nächste Seite der Ergebnisse abzurufen. Weitere Informationen finden Sie unter Analyseergebnisse von Amazon Rekognition Video abrufen.

- Video-Informationen – Die Antwort enthält Informationen über das Videoformat (`VideoMetadata`) auf jeder Seite mit Informationen, die von `GetTextDetection` zurückgegeben werden.


```
{
    "JobStatus": "SUCCEEDED",
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 174441,
        "Format": "QuickTime / MOV",
        "FrameRate": 29.970029830932617,
        "FrameHeight": 480,
        "FrameWidth": 854
    },
    "TextDetections": [
        {
            "Timestamp": 967,
            "TextDetection": {
                "DetectedText": "Twinkle Twinkle Little Star",
                "Type": "LINE",
                "Id": 0,
                "Confidence": 99.91780090332031,
                "Geometry": {
                    "BoundingBox": {
                        "Width": 0.8337579369544983,
                        "Height": 0.08365312218666077,
                        "Left": 0.08313830941915512,
                        "Top": 0.4663468301296234
                    },
                    "Polygon": [
                        {
                            "X": 0.08313830941915512,
                            "Y": 0.4663468301296234
                        },
                        {
                            "X": 0.9168962240219116,
                            "Y": 0.4674469828605652
                        },
```

```
                    {
                        "X": 0.916861355304718,
                        "Y": 0.5511001348495483
                    },
                    {
                        "X": 0.08310343325138092,
                        "Y": 0.5499999523162842
                    }
                ]
            }
        }
    },
    {
        "Timestamp": 967,
        "TextDetection": {
            "DetectedText": "Twinkle",
            "Type": "WORD",
            "Id": 1,
            "ParentId": 0,
            "Confidence": 99.98338317871094,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.2423887550830841,
                    "Height": 0.0833333358168602,
                    "Left": 0.08313817530870438,
                    "Top": 0.46666666865348816
                },
                "Polygon": [
                    {
                        "X": 0.08313817530870438,
                        "Y": 0.46666666865348816
                    },
                    {
                        "X": 0.3255269229412079,
                        "Y": 0.46666666865348816
                    },
                    {
                        "X": 0.3255269229412079,
                        "Y": 0.550000011920929
                    },
                    {
                        "X": 0.08313817530870438,
                        "Y": 0.550000011920929
                    }
```

```
                ]
            }
        }
    },
    {
        "Timestamp": 967,
        "TextDetection": {
            "DetectedText": "Twinkle",
            "Type": "WORD",
            "Id": 2,
            "ParentId": 0,
            "Confidence": 99.982666015625,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.2423887550830841,
                    "Height": 0.08124999701976776,
                    "Left": 0.3454332649707794,
                    "Top": 0.46875
                },
                "Polygon": [
                    {
                        "X": 0.3454332649707794,
                        "Y": 0.46875
                    },
                    {
                        "X": 0.5878220200538635,
                        "Y": 0.46875
                    },
                    {
                        "X": 0.5878220200538635,
                        "Y": 0.550000011920929
                    },
                    {
                        "X": 0.3454332649707794,
                        "Y": 0.550000011920929
                    }
                ]
            }
        }
    },
    {
        "Timestamp": 967,
        "TextDetection": {
            "DetectedText": "Little",
```

```
                "Type": "WORD",
                "Id": 3,
                "ParentId": 0,
                "Confidence": 99.8787612915039,
                "Geometry": {
                    "BoundingBox": {
                        "Width": 0.16627635061740875,
                        "Height": 0.08124999701976776,
                        "Left": 0.6053864359855652,
                        "Top": 0.46875
                    },
                    "Polygon": [
                        {
                            "X": 0.6053864359855652,
                            "Y": 0.46875
                        },
                        {
                            "X": 0.7716627717018127,
                            "Y": 0.46875
                        },
                        {
                            "X": 0.7716627717018127,
                            "Y": 0.550000011920929
                        },
                        {
                            "X": 0.6053864359855652,
                            "Y": 0.550000011920929
                        }
                    ]
                }
            }
        },
        {
            "Timestamp": 967,
            "TextDetection": {
                "DetectedText": "Star",
                "Type": "WORD",
                "Id": 4,
                "ParentId": 0,
                "Confidence": 99.82640075683594,
                "Geometry": {
                    "BoundingBox": {
                        "Width": 0.12997658550739288,
                        "Height": 0.08124999701976776,
```

```
                        "Left": 0.7868852615356445,
                        "Top": 0.46875
                    },
                    "Polygon": [
                        {
                            "X": 0.7868852615356445,
                            "Y": 0.46875
                        },
                        {
                            "X": 0.9168618321418762,
                            "Y": 0.46875
                        },
                        {
                            "X": 0.9168618321418762,
                            "Y": 0.550000011920929
                        },
                        {
                            "X": 0.7868852615356445,
                            "Y": 0.550000011920929
                        }
                    ]
                }
            }
        }
    ],
    "NextToken": "NiHpGbZFnkM/S8kLcukMni15wb05iKtquu/Mwc+Qg1LVlMjjKNOD0Z0GusSPg7TONLe
+OZ3P",
    "TextModelVersion": "3.0"
}
```

Erkennen von Videosegmenten in gespeicherten Videos

Amazon Rekognition Video bietet eine API, die nützliche Videosegmente wie schwarze Frames und Abspann identifiziert.

Heute werden Menschen mehr Inhalte als jemals zuvor präsentiert. Insbesondere die Plattformen Over-The-Top (OTT) und Video-On-Demand (VOD) bieten jederzeit, überall und auf jedem Bildschirm eine große Auswahl an Inhalten. Angesichts der wachsenden Inhaltsvolumina stehen Medienunternehmen vor Herausforderungen bei der Vorbereitung und Verwaltung ihrer Inhalte. Dies ist sehr wichtig, um hochwertigen Benutzerkomfort bereitzustellen und Inhalte besser monetarisieren zu können. Heute verwenden Unternehmen große Teams speziell geschulter Arbeitskräfte, um Aufgaben wie die folgenden zu erfüllen.

- Herausfinden, wo sich der Vor- und Abspann in einem Inhalt befindet
- Auswahl der richtigen Stellen zum Einfügen von Werbung, z. B. in stillen schwarzen Frame-Sequenzen
- Aufteilen von Videos in kleinere Clips für eine einfachere Indizierung.

Diese manuellen Prozesse sind teuer, langsam und können nicht skaliert werden, um mit der Menge an Inhalten, die täglich produziert, lizenziert und aus Archiven abgerufen werden, Schritt zu halten.

Sie können Amazon Rekognition Video verwenden, um betriebliche Medienanalyseaufgaben mithilfe einer vollständig verwalteten, speziell entwickelten Videosegmenterkennung auf APIs Basis von maschinellem Lernen (ML) zu automatisieren. Mithilfe des Amazon Rekognition Video-Segments APIs können Sie problemlos große Videomengen analysieren und Markierungen wie schwarze Bilder oder Veränderungen bei der Aufnahme erkennen. Sie erhalten Zeitcodes, Zeitstempel und Bildnummern von SMPTE (Society of Motion Picture and Television Engineers) zu jeder Erkennung. ML-Erfahrung ist nicht erforderlich.

Amazon Rekognition Video analysiert Videos, die in einem Amazon-Simple-Storage-Service-Bucket (Amazon S3) gespeichert sind. Die zurückgegebenen SMPTE-Zeitcodes sind bildgenau – Amazon Rekognition Video liefert die genaue Bildnummer eines erkannten Videosegments und verarbeitet automatisch verschiedene Video-Bildratenformate. Sie können die Frame-genauen Metadaten aus Amazon Rekognition Video verwenden, um bestimmte Aufgaben vollständig zu automatisieren oder den Prüfungsaufwand für geschulte menschliche Bediener erheblich zu reduzieren, damit diese sich auf kreativere Arbeiten konzentrieren können. Sie können Aufgaben wie die Vorbereitung von

Inhalten, das Einfügen von Werbung und das Hinzufügen von „Binge-Markern“ zu Inhalten in großem Umfang in der Cloud durchführen.

Informationen zu Preisen finden Sie unter Amazon Rekognition – Preise.

Die Segmenterkennung von Amazon Rekognition Video unterstützt zwei Arten von Segmentierungsaufgaben: Technische Signale-Erkennung und Einstellungserkennung.

Themen

- Technische Signale
- Einstellungserkennung
- Über die Amazon-Rekognition-Video-Segmenterkennungs-API
- Verwenden der Amazon-Rekognition-Segment-API
- Beispiel: Erkennen von Segmenten in einem gespeicherten Video

Technische Signale

Ein technisches Signal kennzeichnet schwarze Frames, Farbbalken, Vor- und Abspänne, Studiologos und primäre Programminhalte in einem Video.

Schwarze Frames

Videos enthalten oft leere schwarze Frames ohne Ton, die dazu dienen, Werbung einzufügen oder das Ende eines Programmabschnitts zu markieren, z. B. einer Szene oder eines Vorspanns. Mit Amazon Rekognition Video können Sie Sequenzen mit schwarzen Frames erkennen, um die Schaltung von Werbung zu automatisieren, Inhalte für VOD zu verpacken oder verschiedene Programmteile oder Szenen abzugrenzen. Schwarze Frames mit Ton (wie z. B. Ausblendungen oder Voiceover-Passagen) werden als Inhalt betrachtet und nicht zurückgegeben.

Guthaben

Amazon Rekognition Video kann automatisch die genauen Frames identifizieren, in denen der Vor- und Abspann eines Films oder einer Fernsehsendung beginnt und endet. Mit diesen Informationen können Sie in Video-on-Demand-Anwendungen (VOD) „Binge-Marker“ oder interaktive Zuschaueraufforderungen wie „Nächste Folge“ oder „Vorspann überspringen“ generieren. Sie können auch den ersten und den letzten Frame des Programminhalts in einem Video erkennen. Amazon Rekognition Video ist darauf trainiert, mit einer Vielzahl von Eröffnungs- und Abschlussarten

umzugehen, von einfachen rollierenden Krediten bis hin zu anspruchsvolleren Abrechnungen neben Inhalten.

Farbbalken

Mit Amazon Rekognition Video können Sie Videoabschnitte erkennen, die SMPTE-Farbbalken zeigen. Dabei handelt es sich um eine Reihe von Farben, die in bestimmten Mustern angezeigt werden, damit die Farben auf Monitoren, in Programmen und in Kameras richtig eingestellt sind. Weitere Informationen zu SMPTE-Farbbalken finden Sie unter SMPTE-Farbbalken. Diese Metadaten sind nützlich, um Inhalte für VOD-Anwendungen vorzubereiten, indem Abschnitte mit Farbbalken daraus entfernt werden. Zudem lassen sich damit Probleme wie etwa der Verlust von Übertragungssignalen in einer Aufzeichnung erkennen, wenn anstelle des Inhalts Farbbalken kontinuierlich als Standardsignal angezeigt werden.

Slates

Slates sind Abschnitte des Videos, die in der Regel am Anfang stehen und Textmetadaten zur Folge, zum Studio, zum Videoformat, zu den Audiokanälen und mehr enthalten. Amazon Rekognition Video kann den Anfang und das Ende von Slates erkennen, sodass es einfach ist, die Textmetadaten zu verwenden oder die Slate zu entfernen, wenn Inhalte für die endgültige Ansicht vorbereitet werden.

Studiologos

Studiologos sind Sequenzen, die die Logos oder Embleme des Produktionsstudios zeigen, das an der Produktion der Show beteiligt war. Amazon Rekognition Video kann diese Sequenzen erkennen, sodass Benutzer sie überprüfen können, um Studios zu identifizieren.

Inhalt

Inhalt sind die Teile der Fernsehsendung oder des Films, die das Programm oder verwandte Elemente enthalten. Schwarze Frames, Quellenangaben, Farbbalken, Slates und Studiologos gelten nicht als Inhalt. Amazon Rekognition Video kann den Anfang und das Ende jedes Inhaltssegments im Video erkennen, sodass Sie die Laufzeit des Programms oder bestimmte Segmente ermitteln können.

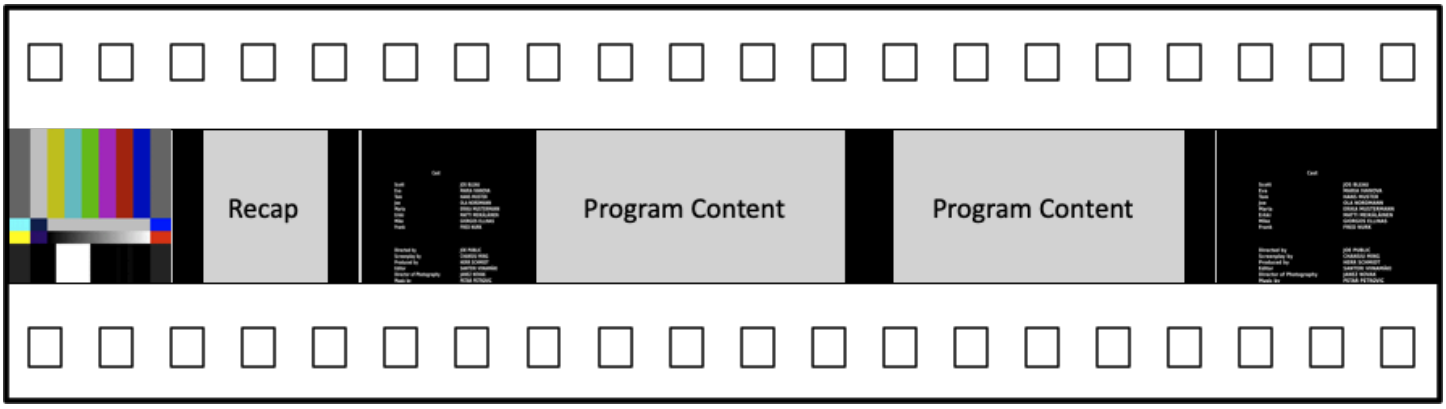
Zu den Inhaltssegmenten gehören unter anderem:

- Programmszenen zwischen zwei Werbepausen
- Eine kurze Zusammenfassung der vorherigen Folge am Anfang des Videos

- Bonus-Inhalt nach dem Abspann
- „Textloser“ Inhalt, z. B. eine Zusammenstellung aller Programmszenen, die ursprünglich überlagerten Text enthielten, bei denen der Text jedoch entfernt wurde, um die Übersetzung in andere Sprachen zu ermöglichen.

Nachdem Amazon Rekognition Video die Erkennung aller Inhaltssegmente abgeschlossen hat, können Sie Domänenwissen anwenden oder die Segmente einer menschlichen Überprüfung unterziehen, um sie weiter zu kategorisieren. Wenn Sie beispielsweise Videos verwenden, die immer mit einer Zusammenfassung beginnen, könnten Sie das erste Inhaltssegment als Zusammenfassung kategorisieren.

Das folgende Diagramm zeigt technische Signalsegmente auf der Zeitachse einer Show oder eines Films. Beachten Sie die Farbbalken und den Vorspann, Inhaltssegmente wie Zusammenfassung und Hauptprogramm, schwarze Frames im gesamten Video und den Abspann.



Einstellungserkennung

Eine Einstellung besteht aus einer Reihe von aufeinander folgenden, zusammenhängenden Bildern, die von einer einzigen Kamera aufgenommen werden und eine kontinuierliche Aktion in Zeit und Raum darstellen. Mit Amazon Rekognition Video können Sie den Anfang, das Ende und die Dauer jeder Aufnahme sowie die Anzahl aller Aufnahmen in einem Inhaltsstück erkennen. Sie können Einstellungsmetadaten für Aufgaben wie die folgenden verwenden.

- Erstellen von Werbevideos mit ausgewählten Einstellungen.
- Einfügen von Werbung an Stellen, an denen der Zuschauer nicht gestört wird, z. B. nicht in der Mitte einer Einstellung, wenn jemand spricht.
- Generieren einer Reihe von Vorschau-Miniaturansichten, die Übergangsinhalte zwischen Einstellungen vermeiden.

Die Einstellungserkennung markiert die exakte Stelle, an der ein harter Schnitt auf eine andere Kamera erfolgt. Bei einem weichen Übergang von einer Kamera zu einer anderen lässt Amazon Rekognition Video den Übergang aus. Dadurch wird sichergestellt, dass Start- und Endzeiten keine Abschnitte ohne eigentlichen Inhalt enthalten.

Das folgende Diagramm zeigt Einstellungserkennungssegmente in einem Film. Beachten Sie, dass jede Einstellung durch einen Schnitt von einem Kamerawinkel oder einer Position zum/zur nächsten identifiziert wird.

Über die Amazon-Rekognition-Video-Segmenterkennungs-API

Um ein gespeichertes Video zu segmentieren, verwenden Sie die asynchronen Operationen StartSegmentDetectionund GetSegmentDetectionAPI-Operationen, um einen Segmentierungsjob zu starten und die Ergebnisse abzurufen. Die Segmenterkennung akzeptiert Videos, die in einem Amazon-S3-Bucket gespeichert sind, und gibt eine JSON-Ausgabe zurück. Sie können wählen, ob nur technische Signale, nur Einstellungsänderungen oder beides zusammen erkannt werden sollen, indem Sie die Amazon-Rekognition-Video-API--`StartSegmentdetection`Anforderung konfigurieren. Sie können erkannte Segmente auch filtern, indem Sie Schwellenwerte für eine minimale Prognosegenauigkeit festlegen. Weitere Informationen finden Sie unter Verwenden der Amazon-Rekognition-Segment-API . Beispielcode finden Sie unter Beispiel: Erkennen von Segmenten in einem gespeicherten Video.

Verwenden der Amazon-Rekognition-Segment-API

Die Segmenterkennung von Amazon Rekognition Video in gespeicherten Videos ist eine asynchrone Operation von Amazon Rekognition Video. Die Amazon-Rekognition-Segment-API ist eine zusammengesetzte API, bei der Sie die Art der Analyse (technische Signale oder

Einstellungserkennung) über einen einzigen API-Aufruf auswählen. Informationen zum Aufrufen asynchroner Operationen finden Sie unter Amazon-Rekognition-Video-Operationen aufrufen.

Themen

- Starten der Segmentanalyse
- Abrufen der Ergebnisse der Segmentanalyse

Starten der Segmentanalyse

So starten Sie die Erkennung von Segmenten in einem gespeicherten Video StartSegmentDetection: Die Eingabeparameter sind mit denen anderer Amazon-Rekognition-Video-Operationen identisch, wobei die Segmenttypauswahl und die Ergebnisfilterung hinzugefügt werden. Weitere Informationen finden Sie unter Starten der Videoanalyse.

Nachfolgend sehen Sie ein von `StartSegmentDetection` übergebenes JSON-Beispiel. Die Anforderung gibt an, dass sowohl technische Signale, als auch Einstellungserkennungssegmente erkannt werden. Für technische Signal-Segmente (90 %) und Einstellungserkennungssegmente (80 %) werden verschiedene Filter für die minimale Prognosegenauigkeit angefordert.

```
{
  "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
      }
  }
}
```

Auswahl eines Segmenttyps

Verwenden Sie den `SegmentTypes`-Array-Eingabeparameter, um technische Signal- und/oder Einstellungserkennungssegmente im Eingabevideo zu erkennen.

- TECHNICAL_CUE – identifiziert framegenaue Zeitstempel für den Beginn, das Ende und die Dauer von technischen Hinweisen (schwarze Frames, Farbbalken, Vorspann, Abspann, Studiologos und primäre Programminhalte), die in einem Video erkannt werden. Beispielsweise können Sie technische Signale verwenden, um den Anfang des Abspanns zu finden. Weitere Informationen finden Sie unter Technische Signale.
- SHOT – identifiziert den Anfang, das Ende und die Dauer einer Einstellung. Sie können beispielsweise das Alleinstellungsmerkmal verwenden, um Kandidateneinstellungen für die abschließende Bearbeitung eines Videos zu identifizieren. Weitere Informationen finden Sie unter Einstellungserkennung.

Filtern der Analyseergebnisse

Mit dem Eingabeparameter `Filters` (StartSegmentDetectionFilters) können Sie die minimale Prognosegenauigkeit für die Antwort angeben. Verwenden Sie in `Filters` `ShotFilter` (StartShotDetectionFilter) zum Filtern der erkannten Einstellungen. Verwenden Sie `TechnicalCueFilter` (StartTechnicalCueDetectionFilter), um technische Hinweise zu filtern.

Beispielcode finden Sie unter Beispiel: Erkennen von Segmenten in einem gespeicherten Video.

Abrufen der Ergebnisse der Segmentanalyse

Das Amazon-Simple-Notification-Service-Thema, zu dem Amazon Rekognition Video die Ergebnisse der Objekterkennung und den Abschlussstatus einer Videoanalyse-Operation veröffentlicht. Wenn die Videoanalyse erfolgreich ist, rufen Sie an, GetSegmentDetectionum die Ergebnisse der Videoanalyse zu erhalten.

Es folgt ein Beispiel für eine `GetSegmentDetection`- Anforderung. Die `JobId` ist die Aufgaben-ID, die vom Aufruf an `StartSegmentDetection` zurückgegeben wird. Informationen zu den weiteren Eingabeparametern finden Sie unter Analyseergebnisse von Amazon Rekognition Video abrufen.

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

```
    "NextToken": "XfXnZKiyMOGDhzBzYUhS5puM+g1IgezqFeYpv/H/+5noP/LmM57FitUAwSQ5D6G4AB/
PNwolrw=="
}
```

`GetSegmentDetection` bietet Ergebnisse für die angeforderte Analyse und allgemeine Informationen über das gespeicherte Video.

Allgemeine Informationen

`GetSegmentDection` gibt die folgenden allgemeinen Informationen zurück.

- Audioinformationen — Die Antwort umfasst Audio-Metadaten in einer Reihe von AudioMetadataObjekten. `AudioMetadata` Es können mehrere Audio-Streams vorhanden sein. Jedes `AudioMetadata`-Objekt enthält Metadaten für einen einzelnen Audio-Stream. Zu den Audioinformationen in `AudioMetadata`-Objekten gehören der Audio-Codec, die Anzahl der Audiokanäle, die Dauer des Audio-Streams und die Abtastrate. Audiometadaten werden auf jeder Seite mit Informationen zurückgegeben, die von `GetSegmentDetection` zurückgegeben werden.
- Videoinformationen — Derzeit gibt Amazon Rekognition Video ein einzelnes VideoMetadataObjekt im `VideoMetadata` Array zurück. Das Objekt enthält Informationen über den Videostrom in der Eingabedatei, die Amazon Rekognition Video für die Analyse ausgewählt hat. Das `VideoMetadata`-Objekt enthält den Video-Codec, das Videoformat und weitere Informationen. Videometadaten werden auf jeder Seite mit Informationen zurückgegeben, die von `GetSegmentDetection` zurückgegeben wird.
- Paging-Informationen – Das Beispiel zeigt eine Seite mit Segmentinformationen. Sie können im `MaxResults`-Eingabeparameter für `GetSegmentDetection` angeben, wie viele Elemente zurückgegeben werden sollen. Wenn mehr Ergebnisse als `MaxResults` vorhanden sind, gibt `GetSegmentDetection` einen Token zurück (`NextToken`), der dazu verwendet wird, die nächste Seite mit Ergebnissen zu erhalten. Weitere Informationen finden Sie unter Analyseergebnisse von Amazon Rekognition Video abrufen.
- Anforderungsinformationen – Die Art der Analyse, die im Aufruf von `StartSegmentDetection` angefordert wird, wird im `SelectedSegmentTypes`-Feld zurückgegeben.

Segmente

Technische Signale und Einstellungsinformationen, die in einem Video erkannt werden, werden in einem Array, `Segments`, von SegmentDetection-Objekten zurückgegeben. Das Array wird nach den Segmenttypen (TECHNICAL_CUE oder SHOT) sortiert, die im `SegmentTypes`-Eingabeparameter

von `StartSegmentDetection` angegeben sind. Innerhalb jedes Segmenttyps wird das Array nach Zeitstempelwerten sortiert. Jedes `SegmentDetection`-Objekt enthält Informationen über den Typ des erkannten Segments (Technisches Signal oder Einstellungserkennung) sowie allgemeine Informationen, wie Startzeit, Endzeit und Dauer des Segments.

Zeitinformationen werden in drei Formaten zurückgegeben.

- Millisekunden

  Die Anzahl der Millisekunden seit dem Start des Videos. Die Felder `DurationMillis`, `StartTimestampMillis` und `EndTimestampMillis` verwenden das Millisekundenformat.

- Timecode

  Amazon-Rekognition-Video-Zeitcodes werden im SMPTE-Format angegeben, wobei jeder Frame des Videos einen eindeutigen Zeitcode-Wert aufweist. Das Format ist hh:mm:ss:frame. Beispielsweise ist ein Timecode-Wert von 01:05:40:07 als „eine Stunde, fünf Minuten, vierzig Sekunden und sieben Frames“ zu lesen. Anwendungsfälle mit Drop-Frame-Rate werden von Amazon Rekognition Video unterstützt. Das Zeitcode-Format für die Drop-Rate ist hh:mm:ss; frame. Die Felder `DurationSMPTE`, `StartTimecodeSMPTE` und `EndTimecodeSMPTE` verwenden das Timecode-Format.

- Frame-Zähler

  Die Dauer jedes Videosegments wird auch durch die Anzahl der Frames ausgedrückt. Das Feld `StartFrameNumber` gibt die Frame-Nummer am Anfang eines Videosegments an und `EndFrameNumber` gibt die Frame-Nummer am Ende eines Videosegments an. `DurationFrames` gibt die Gesamtzahl der Frames in einem Videosegment an. Diese Werte werden anhand eines Frame-Index berechnet, der mit 0 beginnt.

Sie können das `SegmentType`-Feld verwenden, um den Typ eines Segments festzulegen, das von Amazon Rekognition Video zurückgegeben wird.

- Technische Hinweise — Das `TechnicalCueSegment` Feld ist ein TechnicalCueSegmentObjekt, das die Erkennungssicherheit und den Typ eines technischen Hinweises enthält. Die Typen von technischen Hinweisen sind `ColorBars`, `EndCredits`, `BlackFrames`, `OpeningCredits`, `StudioLogo`, `Slate` und `Content`.
- Aufnahme — Das `ShotSegment` Feld ist ein ShotSegmentObjekt, das die Erkennungssicherheit und eine Kennung für das Aufnahmesegment innerhalb des Videos enthält.

Nachfolgend finden Sie ein Beispiel einer JSON-Antwort von `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
        }
    ]
}
```

Beispielcode finden Sie unter Beispiel: Erkennen von Segmenten in einem gespeicherten Video.

Beispiel: Erkennen von Segmenten in einem gespeicherten Video

Das folgende Verfahren zeigt, wie technische Signal-Segmente und Einstellungserkennungssegmente in einem Video, das in einem Amazon-S3-Bucket gespeichert ist, erkannt werden. Das Verfahren zeigt auch, wie erkannte Segmente basierend auf der Erkennungssicherheit von Amazon Rekognition Video gefiltert werden.

Das Beispiel erweitert den Code in Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK), der eine Amazon-Simple-Queue-Service-Warteschlange verwendet, um den Abschlussstatus einer Videoanalyseanforderung zu erhalten.

So erkennen Sie Segmente in einem Video, das in einem Amazon-S3-Bucket gespeichert ist (SDK)

1. Führen Sie Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) aus.
2. Fügen Sie dem Code, den Sie in Schritt 1 verwendet haben, Folgendes hinzu.

   Java

   1. Fügen Sie die folgenden Importe hinzu.

      ```
      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. Fügen Sie den folgenden Code zur Klasse `VideoDetect` hinzu.

```
    //Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights
 Reserved.
    //PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)


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

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

        float minTechnicalCueConfidence = 80F;
        float minShotConfidence = 80F;

        StartSegmentDetectionRequest req = new
 StartSegmentDetectionRequest()
                .withVideo(new Video()
                        .withS3Object(new S3Object()
                                .withBucket(bucket)
                                .withName(video)))
                .withSegmentTypes("TECHNICAL_CUE" , "SHOT")
                .withFilters(new StartSegmentDetectionFilters()
                        .withTechnicalCueFilter(new
 StartTechnicalCueDetectionFilter()

 .withMinSegmentConfidence(minTechnicalCueConfidence))
                        .withShotFilter(new StartShotDetectionFilter()
```

```
 .withMinSegmentConfidence(minShotConfidence)))
                .withJobTag("DetectingVideoSegments")
                .withNotificationChannel(channel);

        StartSegmentDetectionResult startLabelDetectionResult =
 rek.startSegmentDetection(req);
        startJobId=startLabelDetectionResult.getJobId();

    }

    private static void GetSegmentDetectionResults() throws Exception{

        int maxResults=10;
        String paginationToken=null;
        GetSegmentDetectionResult segmentDetectionResult=null;
        Boolean firstTime=true;


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

            GetSegmentDetectionRequest segmentDetectionRequest= new
 GetSegmentDetectionRequest()
                    .withJobId(startJobId)
                    .withMaxResults(maxResults)
                    .withNextToken(paginationToken);

            segmentDetectionResult =
 rek.getSegmentDetection(segmentDetectionRequest);

            if(firstTime) {
                System.out.println("\nStatus\n------");
                System.out.println(segmentDetectionResult.getJobStatus());
                System.out.println("\nRequested features
\n------------------");
                 for (SegmentTypeInfo requestedFeatures :
 segmentDetectionResult.getSelectedSegmentTypes()) {
                    System.out.println(requestedFeatures.getType());
                }
                 int count=1;
```

```
                List<VideoMetadata> videoMetaDataList =
segmentDetectionResult.getVideoMetadata();
                System.out.println("\nVideo Streams\n-------------");
                for (VideoMetadata videoMetaData: videoMetaDataList) {
                    System.out.println("Stream: " + count++);
                    System.out.println("\tFormat: " +
videoMetaData.getFormat());
                    System.out.println("\tCodec: " +
videoMetaData.getCodec());
                    System.out.println("\tDuration: " +
videoMetaData.getDurationMillis());
                    System.out.println("\tFrameRate: " +
videoMetaData.getFrameRate());
                }


                List<AudioMetadata> audioMetaDataList =
segmentDetectionResult.getAudioMetadata();
                System.out.println("\nAudio streams\n-------------");

                count=1;
                for (AudioMetadata audioMetaData: audioMetaDataList) {
                    System.out.println("Stream: " + count++);
                    System.out.println("\tSample Rate: " +
audioMetaData.getSampleRate());
                    System.out.println("\tCodec: " +
audioMetaData.getCodec());
                    System.out.println("\tDuration: " +
audioMetaData.getDurationMillis());
                    System.out.println("\tNumber of Channels: " +
audioMetaData.getNumberOfChannels());
                }
                System.out.println("\nSegments\n--------");

                firstTime=false;
            }


            //Show segment information

            List<SegmentDetection> detectedSegments=
segmentDetectionResult.getSegments();

            for (SegmentDetection detectedSegment: detectedSegments) {
```

```
            if
(detectedSegment.getType().contains(SegmentType.TECHNICAL_CUE.toString()))
{
                System.out.println("Technical Cue");
                TechnicalCueSegment
segmentCue=detectedSegment.getTechnicalCueSegment();
                System.out.println("\tType: " + segmentCue.getType());
                System.out.println("\tConfidence: " +
segmentCue.getConfidence().toString());
             }
            if
(detectedSegment.getType().contains(SegmentType.SHOT.toString())) {
                System.out.println("Shot");
                ShotSegment
segmentShot=detectedSegment.getShotSegment();
                System.out.println("\tIndex " +
segmentShot.getIndex());
                System.out.println("\tConfidence: " +
segmentShot.getConfidence().toString());
             }
             long seconds=detectedSegment.getDurationMillis();
             System.out.println("\tDuration : " + Long.toString(seconds)
+ " milliseconds");
             System.out.println("\tStart time code: " +
detectedSegment.getStartTimecodeSMPTE());
             System.out.println("\tEnd time code: " +
detectedSegment.getEndTimecodeSMPTE());
             System.out.println("\tDuration time code: " +
detectedSegment.getDurationSMPTE());
             System.out.println();

          }

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

   }
```

3. Ersetzen Sie in der Funktion `main` die folgenden Zeilen:

```
      StartLabelDetection(amzn-s3-demo-bucket, video);

      if (GetSQSMessageSuccess()==true)
```

```
         GetLabelDetectionResults();
```

mit:

```
        StartSegmentDetection(amzn-s3-demo-bucket, video);

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

Java V2

```
//snippet-start:[rekognition.java2.recognize_video_text.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;
//snippet-end:[rekognition.java2.recognize_video_text.import]

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

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

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

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];

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

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

    startTextLabels(rekClient, channel, bucket, video);
    GetTextResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

// snippet-start:[rekognition.java2.recognize_video_text.main]
```

```
public static void startTextLabels(RekognitionClient rekClient,
                                NotificationChannel channel,
                                String bucket,
                                String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

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

        StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidOb)
            .build();

        StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

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

public static void GetTextResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetTextDetectionResponse textDetectionResponse=null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (textDetectionResponse !=null)
                paginationToken = textDetectionResponse.nextToken();
```

```
            GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                status = textDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData=textDetectionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());
            System.out.println("Job");

            List<TextDetectionResult> labels=
textDetectionResponse.textDetections();
            for (TextDetectionResult detectedText: labels) {
                System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
                System.out.println("Id : " +
detectedText.textDetection().id());
                System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
                System.out.println("Type: " +
detectedText.textDetection().type());
```

```
                System.out.println("Text: " +
 detectedText.textDetection().detectedText());
                System.out.println();
            }

        } while (textDetectionResponse !=null &&
 textDetectionResponse.nextToken() != null);

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

Python

1. Fügen Sie den folgenden Code in der Klasse `VideoDetect` ein, die Sie in Schritt 1 erstellt haben.

```
# 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. Ersetzen Sie in der Funktion `main` die folgenden Zeilen:

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

mit:

```
    analyzer.StartSegmentDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetSegmentDetectionResults()
```

> **Note**
>
> Wenn Sie zusätzlich zu Analysieren eines in einem Amazon S3-Bucket gespeicherten Videos mit Java oder Python (SDK) bereits ein anderes Videobeispiel ausgeführt haben, ist der zu ersetzende Code möglicherweise anders.

3. Führen Sie den Code aus. Es werden Informationen über die im Eingabevideo erkannten Segmente angezeigt.

Echtheit von Gesichtern erkennen

Mit Amazon Rekognition Face Liveness können Sie überprüfen, ob ein Benutzer, der eine Gesichtsüberprüfung durchläuft, physisch vor einer Kamera anwesend ist. Es erkennt auch gefälschte Angriffe, die auf eine Kamera gerichtet sind oder versuchen, eine Kamera zu umgehen. Benutzer können eine Face-Liveness-Kontrolle durchführen, indem sie ein kurzes Video-Selfie machen, bei dem sie einer Reihe von Anweisungen folgen, um ihre Anwesenheit zu überprüfen.

Die Echtheit von Gesichtern wird anhand einer probabilistischen Berechnung bestimmt, und nach der Überprüfung wird ein Zuverlässigkeitswert (zwischen 0 und 100) zurückgegeben. Je höher der Wert, desto größer ist die Zuverlässigkeit, dass die Person, die den Scheck entgegennimmt, live ist. Face Liveness gibt auch einen Rahmen zurück, ein sogenanntes Referenzbild, das für Gesichtsvergleiche und -suchen verwendet werden kann. Wie bei jedem wahrscheinlichkeitsbasierten System kann Face Liveness keine perfekten Ergebnisse garantieren. Verwenden Sie es zusammen mit anderen Faktoren, um eine risikobasierte Entscheidung über die persönliche Identität der Benutzer zu treffen.

Face Liveness verwendet mehrere Komponenten:

- AWS Amplify SDK (React, Swift (iOS) und Android) mit Komponente FaceLivenessDetector
- AWS SDKs
- AWS Cloud APIs

Wenn Sie Ihre Anwendung für die Integration mit dem Face-Liveness-Feature konfigurieren, verwendet sie die folgenden API-Operationen:

- CreateFaceLivenessSession- Startet eine Face Liveness-Sitzung, sodass das Face Liveness-Erkennungsmodell in Ihrer Anwendung verwendet werden kann. Gibt a SessionId für die erstellte Sitzung zurück. Ermöglicht es Ihnen auch ChallengePrefrence, Ihre festzulegen, sodass Sie die FaceMovementChallenge Option verwenden können.
- StartFaceLivenessSession- Von AWS Amplify FaceLivenessDetector angerufen. Startet einen Ereignisstrom mit Informationen zu relevanten Ereignissen und Attributen in der aktuellen Sitzung.
- GetFaceLivenessSessionResults- Ruft die Ergebnisse einer bestimmten Face Liveness-Sitzung ab, einschließlich eines Face Liveness-Konfidenzwerts, eines Referenzbilds und von Auditbildern.

Sie werden das AWS Amplify SDK verwenden, um die Face Liveness-Funktion in Ihre Workflows zur gesichtsbasierten Überprüfung für Webanwendungen zu integrieren. Wenn sich Benutzer über Ihre

Anwendung anmelden oder authentifizieren, senden Sie sie an den Workflow zur Face-Liveness-Kontrolle im Amplify SDK. Das Amplify SDK kümmert sich um die Benutzeroberfläche und das Feedback der Benutzer in Echtzeit, während sie ihr Video-Selfie aufnehmen.

Wenn sich FaceMovementAndLightChallenge das Gesicht des Benutzers bei Verwendung in das auf seinem Gerät angezeigte Oval bewegt, zeigt das Amplify SDK eine Abfolge von farbigen Lichtern auf dem Bildschirm an. Anschließend wird das Selfie-Video sicher in die Cloud gestreamt. APIs Alternativ bewegt sich das Gesicht des Benutzers bei der Verwendung FaceMovementChallenge in das auf seinem Gerät angezeigte Oval, es gibt jedoch keine Abfolge von farbigen Lichtern. 'FaceMovementAndLightChallenge' ist zwar nach wie vor die beste Einstellung, um die Genauigkeit zu maximieren, ermöglicht es Kunden jedoch, einer schnelleren Überprüfung der Aktualität Vorrang vor der Genauigkeit einzuräumen. FaceMovementChallenge Bei der Auswahl zwischen diesen Einstellungen sollten Kunden ihre jeweiligen Anwendungsfälle berücksichtigen, einschließlich erwarteter Angriffsarten, gewünschter Falschakzeptanz- und Falschablehnungsraten sowie zusätzlicher Prüfungen wie Geolokalisierung (z. B. basierend auf IP), One-Time-Pass-Codes (OTPs) usw. Kunden sollten diese Entscheidung treffen, nachdem sie die Leistung von Liveness anhand verschiedener Schwellenwerte für den Vertrauenswert für ihre spezifischen Inhalte getestet haben. Darüber hinaus sollten die Anwendungseigentümer bei beiden Typen von Verfügbarkeitsprüfungen Kontrollen implementieren, um das Gerät, von dem der Videostream gesendet wird, zu schützen. Nach Abschluss der Analyse erhalten Sie im Backend Folgendes:

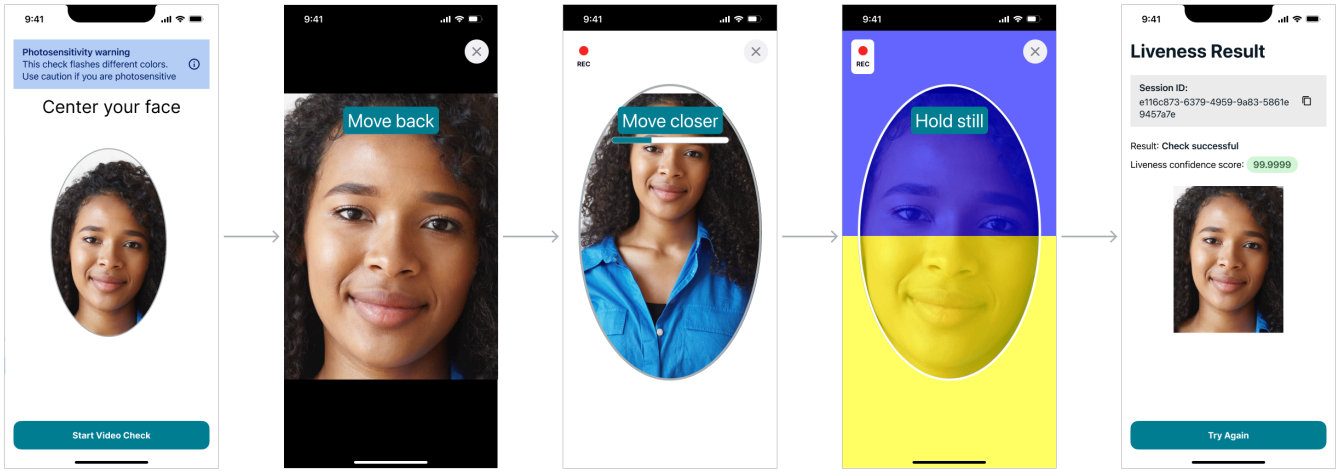
- Ein Face-Liveness-Zuverlässigkeitswert (zwischen 0 und 100)
- Ein qualitativ hochwertiges Bild, das als Referenzbild bezeichnet wird und für den Gesichtsabgleich oder die Gesichtssuche verwendet werden kann
- Ein Satz von bis zu vier Bildern, sogenannten Auditbildern, die aus dem Selfie-Video ausgewählt wurden

Face Liveness kann für eine Vielzahl von Anwendungsfällen genutzt werden. Face Liveness kann beispielsweise zusammen mit dem Gesichtsabgleich (mit CompareFacesund SearchFacesByImage) zur Identitätsprüfung, zur Altersschätzung auf Plattformen mit altersabhängiger Zugangsbeschränkung und zur Erkennung echter menschlicher Benutzer und zur Abschreckung von Bots verwendet werden.

In der Rekognition-Face-Liveness-KI-Servicekarte erfahren Sie mehr über die Anwendungsfälle, für die der Service vorgesehen ist, wie Machine Learning (ML) vom Service verwendet wird und welche wichtigen Überlegungen zur verantwortungsvollen Gestaltung und Nutzung des Dienstes zu beachten sind.

Sie können Schwellenwerte für Face-Liveness- und Gesichtsabgleich-Zuverlässigkeit festlegen. Ihre ausgewählten Schwellenwerte sollten Ihren Anwendungsfall widerspiegeln. Anschließend senden Sie dem Benutzer eine Identitätsprüfung approval/denial , die darauf basiert, dass die Punktzahl über oder unter den Schwellenwerten liegt. Falls dies verweigert wird, bitten Sie den Benutzer, es erneut zu versuchen oder ihn an eine andere Methode weiterzuleiten.

Die folgende Grafik zeigt den Benutzerablauf von den Anweisungen über die Überprüfung der Aktualität bis hin zum zurückgegebenen Ergebnis:



Benutzerseitige Anforderungen an Face Liveness

Für Amazon Rekognition Face Liveness sind die folgenden Mindestanforderungen erforderlich:

Geräte:

- Das Gerät muss über eine nach vorne gerichtete Kamera verfügen
- Minimale Bildwiederholfrequenz der Geräteanzeige: 60 Hz
- Minimale Anzeige- oder Bildschirmgröße: 4 Zoll
- Das Gerät sollte nicht gejailbreakt oder gerootet sein

Technische Daten der Kamera:

- Farbkamera: Die nach vorne gerichtete Kamera sollte Farben aufnehmen können.
- Keine virtuelle Kamera oder Kamerasoftware.

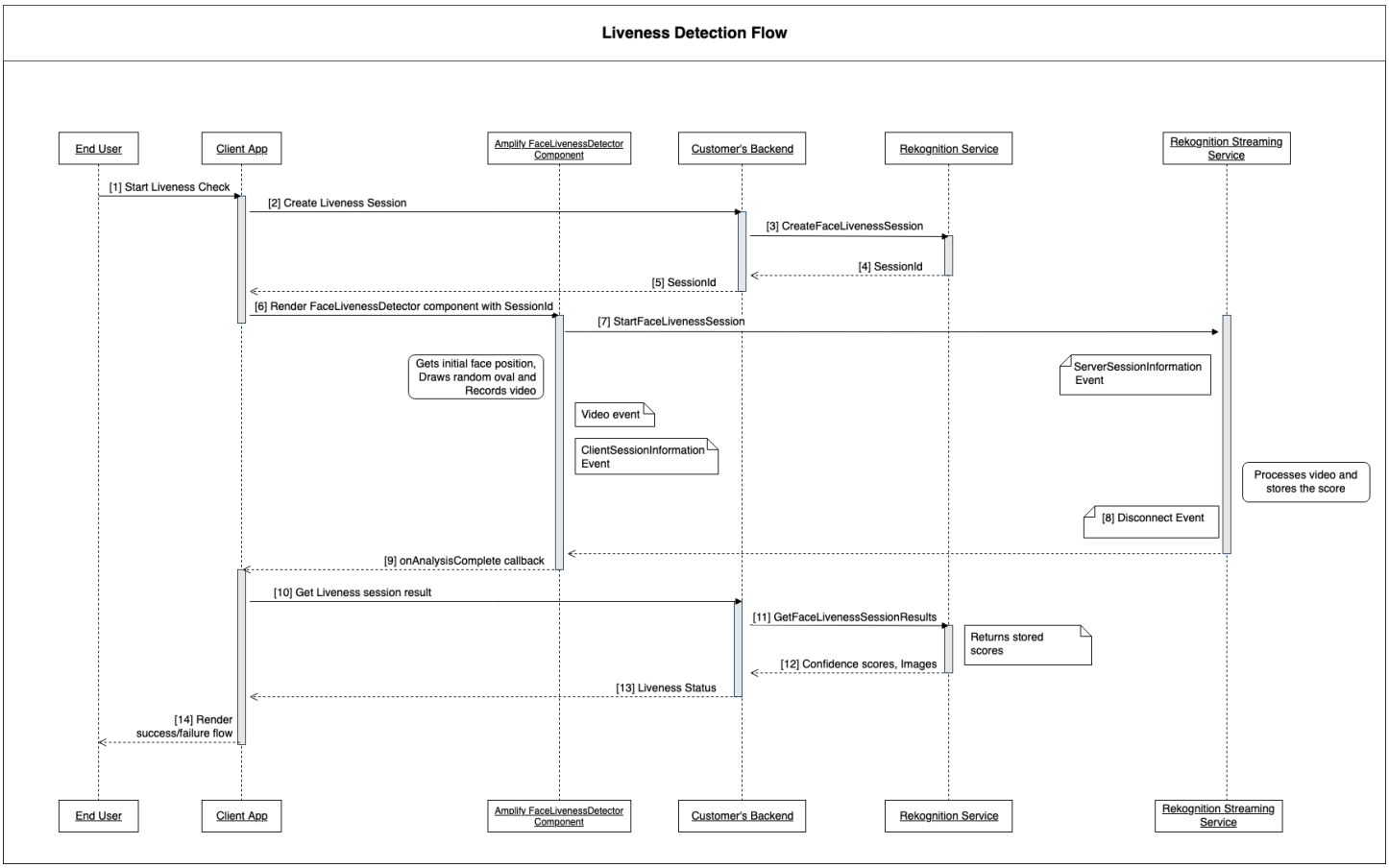
- Minimale Aufnahmekapazität: 15 Bilder pro Sekunde.
- Mindestauflösung für Videoaufnahmen: 480x640px.
- Wenn Benutzer eine Webcam mit einem Desktop für eine Kontrolle der Face Liveness verwenden, ist es wichtig, die Webcam oben auf demselben Bildschirm zu montieren, auf dem die Kontrolle der Face Liveness beginnt.

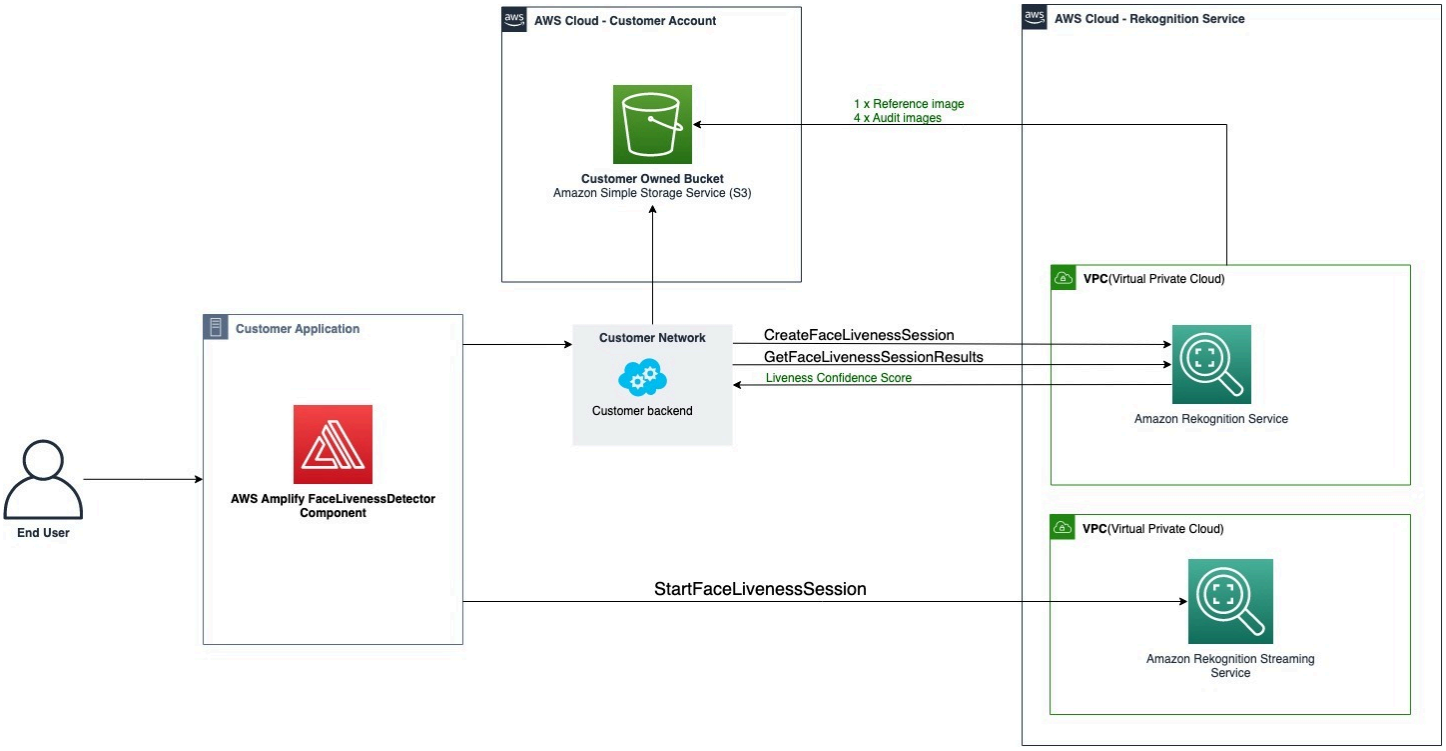
Minimale Bandbreitenanforderung: 100 kBit/s

Unterstützte Browser: Die letzten drei Versionen der wichtigsten Browser wie Google Chrome, Mozilla Firefox, Apple Safari und Microsoft Edge. Weitere Informationen zur Browserunterstützung finden Sie unter Welche Browser werden für die Verwendung mit der AWS-Managementkonsole unterstützt?

Architektur- und Reihenfolgediagramme

In den folgenden Diagrammen wird detailliert beschrieben, wie Amazon Rekognition Face Liveness in Bezug auf die Architektur und die Reihenfolge der Operationen der Funktion funktioniert:





Der Prozess zur Kontrolle von Face Liveness umfasst mehrere Schritte, die im Folgenden beschrieben werden:

1. Der Benutzer initiiert in der Client-App eine Kontrolle der Face Liveness.
2. Die Client-App ruft das Backend des Kunden auf, das wiederum den Amazon-Rekognition-Service aufruft. Der Service erstellt eine Face Liveness-Sitzung und gibt eine eindeutige Sitzung zurück. SessionId Hinweis: Nachdem eine SessionId gesendet wurde, läuft sie in 3 Minuten ab, sodass nur ein Zeitfenster von 3 Minuten zur Verfügung steht, um die nachfolgenden Schritte 3 bis 7 abzuschließen. Für jeden Face Liveness Check muss eine neue SessionID verwendet werden. Wenn eine bestimmte SessionID für nachfolgende Face Liveness-Checks verwendet wird, schlagen die Prüfungen fehl. Darüber hinaus SessionId läuft eine Datei 3 Minuten nach dem Senden ab, sodass alle mit der Sitzung verknüpften Liveness-Daten (z. B. SessionID, Referenzbild, Audit-Bilder usw.) nicht verfügbar sind.
3. Die Client-App rendert die FaceLivenessDetector Amplify-Komponente unter Verwendung der erhaltenen SessionId und entsprechenden Callbacks.
4. Die FaceLivenessDetector Komponente stellt eine Verbindung zum Amazon Rekognition-Streaming-Dienst her, rendert ein Oval auf dem Bildschirm des Benutzers und zeigt eine Abfolge von farbigen Lichtern an. FaceLivenessDetector zeichnet Videos in Echtzeit auf und streamt sie an den Amazon Rekognition Rekognition-Streaming-Service.

5. Der Amazon Rekognition Rekognition-Streaming-Service verarbeitet das Video in Echtzeit, speichert die Ergebnisse und gibt a DisconnectEvent an die FaceLivenessDetector Komponente zurück, wenn das Streaming abgeschlossen ist.
6. Die FaceLivenessDetector Komponente ruft den `onAnalysisComplete` Callback auf, um der Client-App zu signalisieren, dass das Streaming abgeschlossen ist und die Ergebnisse abgerufen werden können.
7. Die Client-App ruft das Backend des Kunden auf, um eine boolesche Markierung zu erhalten, die angibt, ob der Benutzer live war oder nicht. Das Kunden-Backend sendet eine Anforderung an den Amazon-Rekognition-Service, um den Vertrauenswert, die Referenz und die Auditbilder zu erhalten. Das Kunden-Backend verwendet diese Attribute, um festzustellen, ob der Benutzer aktiv ist, und sendet eine entsprechende Antwort an die Client-App zurück.
8. Schließlich leitet die Client-App die Antwort an die FaceLivenessDetector Komponente weiter, die die success/failure Nachricht entsprechend rendert, um den Ablauf abzuschließen.

Voraussetzungen

Zu den Voraussetzungen für die Nutzung von Amazon Rekognition Face Liveness gehören:

1. Richten Sie ein Konto ein AWS
2. Richten Sie Face Liveness ein AWS SDKs
3. AWS Amplify-Ressourcen einrichten

Schritt 1: Einrichten eines AWS -Kontos

Wenn Sie noch kein AWS Konto haben, führen Sie die Schritte unter aus, um eines Erstellen Sie ein AWS Konto und einen Benutzer zu erstellen.

Schritt 2: Face Liveness einrichten AWS SDKs

Falls Sie es noch nicht getan haben, installieren und konfigurieren Sie das AWS CLI und das AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

Es gibt mehrere Möglichkeiten, AWS SDK-Aufrufe zu authentifizieren. Bei den Beispielen in diesem Handbuch wird davon ausgegangen, dass Sie ein Standard-Anmeldeinformationsprofil für den Aufruf von AWS CLI-Befehlen und AWS SDK-API-Operationen verwenden.

Weitere Informationen dazu, wie Sie Ihrem Benutzerkonto Zugriff auf das von Ihnen gewählte AWS SDK gewähren können, finden Sie auf der Seite Gewährung von programmatischem Zugriff. Auf der Seite wird auch erklärt, wie Sie ein Profil auf Ihrem lokalen Computer verwenden und wie Sie den Beispielcode in AWS Umgebungen ausführen.

Stellen Sie sicher, dass der Benutzer, der die Face-Liveness-Operationen aufruft, über die richtigen Berechtigungen zum Aufrufen der Operationen verfügt, z. B. die Berechtigungen `AmazonRekognitionFullAccess` und `AmazonS3FullAccess`.

Schritt 3: AWS Amplify Resources einrichten

Um Amazon Rekognition Face Liveness in Ihre App zu integrieren, müssen Sie das AWS Amplify SDK für die Verwendung der Amplify-Komponente einrichten. FaceLivenessDetector

Falls Sie dies noch nicht getan haben, folgen Sie den Anweisungen zur Einrichtung der AWS Command Line Interface (AWS CLI) unter Erste Schritte mit der AWS-CLI. Führen Sie nach der Installation der CLI die Schritte Configure Auth auf der Amplify UI-Dokumentseite aus, um Ihre AWS Amplify-Ressourcen einzurichten.

Bewährte Methoden zur Erkennung von Face Liveness

Wir empfehlen, dass Sie mehrere bewährte Methoden befolgen, wenn Sie Amazon Rekognition Face Liveness verwenden. Zu den bewährten Methoden für Face Liveness gehören Richtlinien für den Ort, an dem Kontrollen zu Face Liveness durchgeführt werden sollten, die Verwendung von Auditbildern und die Auswahl von Schwellenwerten für die Vertrauensbewertung.

Die vollständige Liste der bewährten Verfahren finden Sie unter Empfehlungen für die Verwendung von Face Liveness.

Programmierung von Amazon Rekognition Face Liveness APIs

Um die Amazon-Rekognition-Face-Liveness-API zu verwenden, müssen Sie ein Backend erstellen, das die folgenden Schritte ausführt:

1. Rufen Sie an CreateFaceLivenessSession, um eine Face Liveness-Sitzung einzuleiten. Wenn die `CreateFaceLivenessSession`-Operation abgeschlossen ist, fordert die Benutzeroberfläche den Benutzer auf, ein Video-Selfie einzureichen. Die FaceLivenessDetector Komponente von AWS Amplify ruft dann auf, StartFaceLivenessSessionum eine Liveness-Erkennung durchzuführen.

2. Rufen Sie GetFaceLivenessSessionResultsauf, um die Erkennungsergebnisse für eine Face Liveness-Sitzung zurückzugeben.
3. Fahren Sie mit der Konfiguration Ihrer React-Anwendung für die Verwendung der FaceLivenessDetector Komponente fort, indem Sie die Schritte im Amplify Liveness Guide befolgen.

Bevor Sie Face Liveness verwenden, stellen Sie sicher, dass Sie ein AWS-Konto erstellt, die AWS-CLI und AWS eingerichtet und AWS SDKs Amplify eingerichtet haben. Sie sollten auch sicherstellen, dass die IAM-Richtlinie für Ihre Backend-API über Berechtigungen verfügt, die Folgendes abdecken: `GetFaceLivenessSessionResults` und `CreateFaceLivenessSession`. Weitere Informationen finden Sie im Abschnitt Voraussetzungen.

Schritt 1: CreateFaceLivenessSession

CreateFaceLivenessSession Der API-Vorgang erstellt eine Face Liveness-Sitzung und gibt eine eindeutige Sitzung zurück. `SessionId`

Als Teil der Eingabe für diese Operation ist es auch möglich, einen Amazon-S3-Bucket-Ort anzugeben. Dies ermöglicht die Speicherung eines Referenzbilds und von Auditbildern, die während der Face-Liveness-Sitzung generiert wurden. Der Amazon-S3-Bucket muss sich im AWS-Konto des Aufrufers befinden und sich in derselben Region befinden wie der Face-Liveness-Endpunkt. Darüber hinaus werden die S3-Objektschlüssel vom Face-Liveness-System generiert.

Es ist auch möglich, ein `AuditImagesLimit` anzugeben, was eine Zahl zwischen 0 und 4 ist. Standardmäßig ist der Wert 0 eingestellt. Die Anzahl der zurückgegebenen Bilder wurde nach bestem Wissen ermittelt und basiert auf der Dauer des Selfie-Videos.

Schließlich können Sie eine angeben ChallengePreference, einschließlich des Typs und der Version der Herausforderung, die Ihre Sitzung standardmäßig verwenden soll. Standardmäßig wird dies verwendet, kann FaceLivenessMovementAndLightChallenge aber auch auf eingestellt werden FaceMovementChallenge.

Anforderungsbeispiel

```
{
   "ClientRequestToken": "string",
   "KmsKeyId": "string",
```

```
   "Settings": {
      "AuditImagesLimit": number,
      "ChallengePreferences": [
         {
            "Type": "string",
            "Versions": {
               "Maximum": "string",
               "Minimum": "string"
            }
         }
      ],
      "OutputConfig": {
         "S3Bucket": "string",
         "S3KeyPrefix": "string"
      }
   }
}
```

Antwortbeispiel

```
{
    {"SessionId": "0f959dbb-37cc-45d8-a08d-dc42cce85fa8"}
}
```

Schritt 2: StartFaceLivenessSession

Wenn der CreateFaceLivenessSession API-Vorgang abgeschlossen ist, führt die AWS Amplify Amplify-Komponente den StartFaceLivenessSession API-Vorgang aus. Der Benutzer wird aufgefordert, ein Video-Selfie aufzunehmen. Für eine erfolgreiche Überprüfung muss der Benutzer sein Gesicht innerhalb des auf dem Bildschirm angezeigten Ovals positionieren und dabei für eine gute Beleuchtung sorgen. Weitere Informationen finden Sie unter Empfehlungen für die Verwendung von Face Liveness.

Dieser API-Vorgang erfordert das während der Face Liveness-Sitzung aufgenommene Video, die durch den CreateFaceLivenessSession API-Vorgang abgerufene sessionId und einen Rückruf. `onAnalysisComplete` Der Callback kann verwendet werden, um dem Backend zu signalisieren, den GetFaceLivenessSessionResults API-Vorgang aufzurufen, der eine Vertrauensbewertung, Referenz- und Auditbilder zurückgibt.

Beachten Sie, dass dieser Schritt von der AWS Amplify FaceLivenessDetector Amplify-Komponente in der Client-Anwendung ausgeführt wird. Sie müssen keine zusätzlichen Einstellungen vornehmen, um`StartFaceLivenessSession` aufzurufen.

Schritt 3: GetFaceLivenessSessionResults

Der GetFaceLivenessSessionResults API-Vorgang ruft die Ergebnisse einer bestimmten Face Liveness-Sitzung ab. Sie benötigt die sessionId als Eingabe und gibt den entsprechenden Face-Liveness-Zuverlässigkeitswert zurück. Sie bietet auch ein Referenzbild, das einen Gesichtsbegrenzungsrahmen enthält, sowie Audit-Bilder, die auch Gesichtsbegrenzungsrahmenen enthalten. Der Zuverlässigkeitswert für Face Liveness liegt zwischen 0 und 100.

Anforderungsbeispiel

```
{"SessionId": "0f959dbb-37cc-45d8-a08d-dc42cce85fa8"}
```

Antwortbeispiel

```
{
    "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"
}
```

Schritt 4: Reagieren Sie auf Ergebnisse

Vergleichen Sie nach der Face-Liveness-Sitzung den Zuverlässigkeitswert der Kontrolle mit dem angegebenen Schwellenwert. Wenn der Wert höher als der Schwellenwert ist, kann der Benutzer zum nächsten Bildschirm oder zur nächsten Aufgabe wechseln. Schlägt die Kontrolle fehl, wird der Benutzer benachrichtigt und aufgefordert, es erneut zu versuchen.

Ich nenne Face Liveness APIs

Sie können Amazon Rekognition Face Liveness mit jedem unterstützten AWS SDK testen, z. B. dem AWS Python SDK Boto3 oder dem AWS SDK for Java. Sie können das `CreateFaceLivenessSession` und `GetFaceLivenessSessionResults` APIs mit dem von Ihnen ausgewählten SDK aufrufen. Der folgende Abschnitt zeigt, wie man diese APIs mit Python und Java aufruft SDKs.

Um das Gesicht Liveness APIs zu nennen:

- Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`-Berechtigungen, falls Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Schritt 1: Ein AWS-Konto einrichten und einen Benutzer erstellen.
- Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie die AWS-CLI und AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: AWS-CLI und AWS einrichten SDKs.

Python

Der folgende Ausschnitt zeigt, wie Sie diese APIs in Ihren Python-Anwendungen aufrufen können. Beachten Sie, dass Sie zur Ausführung dieses Beispiels mindestens Version 1.26.110 des Boto3-SDK verwenden müssen, obwohl die neueste Version des SDK empfohlen wird.

```
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

Der folgende Ausschnitt zeigt, wie Sie diese APIs in Ihren Java-Anwendungen aufrufen können:

```
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

Das folgende Snippet zeigt, wie Sie Face Liveness APIs mit dem Java V2 SDK aufrufen können: 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

Das folgende Snippet zeigt, wie Face APIs Liveness mit dem SDK Node.Js aufgerufen wird: 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)

Das folgende Snippet zeigt, wie Face Liveness APIs mit dem Node.Js SDK für Javascript v3 aufgerufen wird AWS :

```
import { RekognitionClient, CreateFaceLivenessSessionCommand } from "@aws-sdk/
client-rekognition"; // ES Modules
import const { RekognitionClient, CreateFaceLivenessSessionCommand } =
 require("@aws-sdk/client-rekognition"); // CommonJS import
const client = new RekognitionClient(config);
const input = {
  KmsKeyId: "STRING_VALUE",
  Settings: {
    OutputConfig: { // LivenessOutputConfig
      S3Bucket: "STRING_VALUE", // required
      S3KeyPrefix: "STRING_VALUE",
    },
    AuditImagesLimit: Number("int"),
  },
  ClientRequestToken: "STRING_VALUE",
};
const command = new CreateFaceLivenessSessionCommand(input);
const response = await client.send(command);
// { // CreateFaceLivenessSessionResponse
//   SessionId: "STRING_VALUE", // required
// };
```

Konfiguration und Anpassung Ihrer Anwendung

Konfigurieren Ihrer -Anwendung

Ihre Face-Liveness-Anwendung kann auf Mobilgeräten oder Desktop-Webbrowsern ausgeführt werden. Sie sollten die Face-Liveness-Komponenten so konfigurieren, dass sie in die von Ihnen gewählte Lösung integriert werden können. Sie müssen außerdem sicherstellen, dass Ihre

Anwendung die Erlaubnis hat, die Kamera eines Geräts zu verwenden. Der Amplify-Liveness-Leitfaden enthält detaillierte Anweisungen zu folgenden Punkten:

- Installieren und Konfigurieren von AWS Amplify
- Importieren und rendern Sie die Komponente FaceLivenessDetector
- Rückrufe anhören
- Beispiel für Fehlermeldungen von Amplify rendern

Anpassen Ihrer Anwendung

Mit AWS Amplify können Sie bestimmte Komponenten Ihrer Liveness-Anwendung anpassen.

Informationen zur Übersetzung finden Sie in der Amplify-Authenticator-Dokumentation.

Informationen zum Anpassen von Amplify-Komponenten und Amazon-SNS-Themen finden Sie in der Amplify-Dokumentation zum Theming.

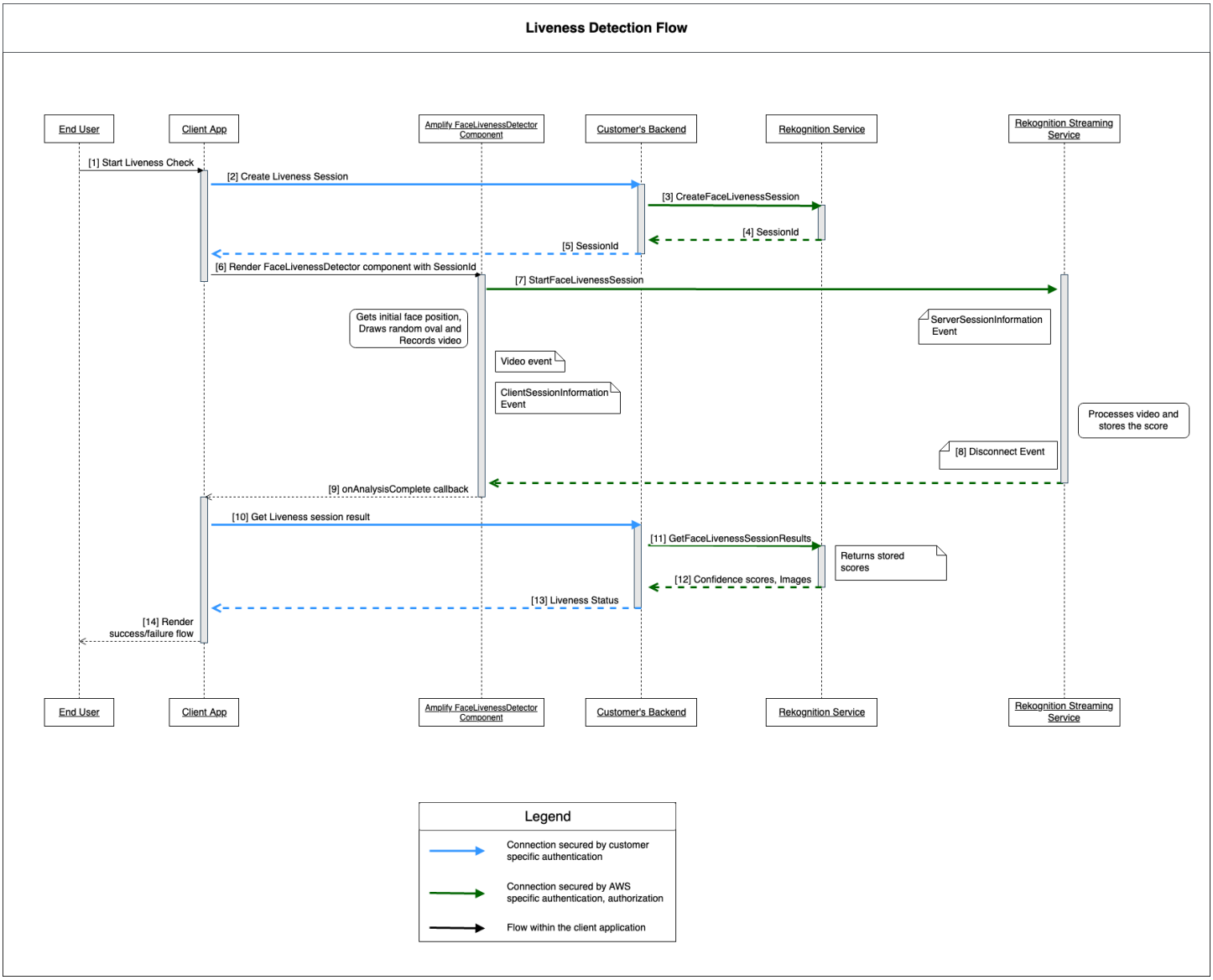
Modell der gemeinsamen Verantwortung für Face Liveness

Sicherheit und Compliance sind eine gemeinsame AWS Verantwortung von Ihnen, dem Kunden. Lesen Sie hier mehr über das Modell der AWS gemeinsamen Verantwortung.

1. Alle Aufrufe des AWS Dienstes (über die Client-Anwendung oder das Kunden-Backend) werden mit AWS Auth (AWS Authentifizierung) authentifiziert und autorisiert. Es liegt in der Verantwortung der Eigentümer des Face-Liveness-Dienstes, sicherzustellen, dass dies geschieht.
2. Alle Aufrufe an das Kunden-Backend (von der Client-Anwendung aus) werden über den Kunden authentifiziert und autorisiert. Diese Verantwortung liegt beim Kunden. Der Kunde muss sicherstellen, dass Aufrufe von der Client-Anwendung authentifiziert sind und in keiner Weise manipuliert wurden.
3. Das Kunden-Backend muss den Endbenutzer identifizieren, der den Face-Liveness-Vorgang durchführt. Es liegt in der Verantwortung des Kunden, einen Endbenutzer an eine Face-Liveness-Sitzung zu binden. Der Face-Liveness-Service unterscheidet nicht zwischen Endbenutzern. Es kann nur die AWS Identität des Anrufers identifizieren (die der Kunde verarbeitet).
4. AWS empfiehlt Kunden, zusätzlich zu Face Liveness zusätzliche Validierungsprüfungen wie Standortgeolokalisierung (z. B. basierend auf IP), One Time Pass-Codes (OTPs) usw. durchzuführen, die ihren Anwendungsfallanforderungen und ihrer Sicherheitslage entsprechen.

Die Einstellung 'FaceMovementAndLightChallenge' bietet die höchste Genauigkeit für Rekognition Liveness, da Benutzer ihr Gesicht in Richtung Bildschirm bewegen und still halten müssen, bis eine Reihe von blinkenden Lichtern angezeigt wird. Wir empfehlen Kunden, diese Standardeinstellung zu verwenden. Alternativ können Kunden die Einstellung 'FaceMovementChallenge' aktivieren, wodurch die Prüfzeit um mehrere Sekunden reduziert wird, indem die blinkenden Lichter entfernt werden. Die Einstellung 'FaceMovementAndLightChallenge' ist zwar nach wie vor die beste Einstellung, um die Genauigkeit zu maximieren, ermöglicht es den Kunden jedoch, schnelleren Prüfungen der Verfügbarkeit Priorität einzuräumen. FaceMovementChallenge Bei der Auswahl zwischen diesen Einstellungen sollten Kunden ihre jeweiligen Anwendungsfälle berücksichtigen, einschließlich erwarteter Angriffstypen, gewünschter Falschakzeptanz- und Falschablehnungsraten, und auch zusätzliche Prüfungen wie Geolokalisierung (z. B. basierend auf IP), One-Time-Pass-Codes (OTPs) usw. implementieren. Kunden sollten diese Entscheidung treffen, nachdem sie die Leistung von Liveness getestet haben, wobei sie je nach Anwendungsfall unterschiedliche Schwellenwerte für den Vertrauenswert festlegen. Die Kunden sind dafür verantwortlich, Kontrollen zur Sicherung des Geräts zu implementieren, von dem das Video gesendet wird

Das folgende Flussdiagramm zeigt, welche Aufrufe vom AWS-Service oder vom Kunden authentifiziert werden:



Alle Aufrufe an den Amazon Rekognition Face Liveness Service sind durch AWS Auth geschützt (mithilfe eines AWS Signaturmechanismus). Dazu gehören die folgenden Aufrufe:

- [3] CreateFaceLivenessSessionAPI-Aufruf (vom Backend des Kunden)
- [7] StartFaceLivenessSessionAPI-Aufruf (von der Client-Anwendung)
- [11] GetFaceLivenessSessionResultsAPI-Aufruf (vom Backend des Kunden)

Alle Aufrufe an das Backend des Kunden müssen über einen Authentifizierungs- und Autorisierungsmechanismus verfügen. Kunden müssen sicherstellen, dass der code/library/etc verwendete Drittanbieter aktiv gewartet und weiterentwickelt wird. Kunden müssen außerdem

sicherstellen, dass der richtige Endbenutzer Aufrufe zur richtigen Face-Liveness-Sitzung tätigt. Kunden müssen die folgenden Abläufe authentifizieren und autorisieren:

- [2] Face-Liveness-Sitzung erstellen (aus der Client-Anwendung)
- [10] Ergebnis der Face-Liveness-Sitzung abrufen (aus der Client-Anwendung)

Kunden können dem STRIDE-Sicherheitsmodell folgen, um sicherzustellen, dass ihre API-Aufrufe geschützt sind.

Typ	Beschreibung	Sicherheitskontrolle
Spoofing	Bedrohungsaktion, die darauf abzielt, auf die Anmeldein formationen eines anderen Benutzers wie Benutzername und Passwort zuzugreifen und diese zu verwenden.	Authentifizierung
Manipulation	Bedrohungsaktion, die darauf abzielt, persistente Daten in böswilliger Absicht zu ändern oder zu modifizieren. Beispiele hierfür sind Datensätze in einer Datenbank und die Änderung von Daten, die zwischen zwei Computern über ein offenes Netzwerk, z. B. das Internet, übertragen werden.	Integrität
Ablehnung	Bedrohungsmaßnahmen, die darauf abzielen, verbotene Operationen in einem System durchzuführen, das nicht in der Lage ist, die Operationen nachzuverfolgen.	Nichtabstreitbarkeit

Offenlegung von Informationen	Bedrohungsaktion, die beabsichtigt, eine Datei zu lesen, auf die man keinen Zugriff hat, oder Daten während der Übertragung zu lesen.	Vertraulichkeit
Verweigerung des Dienstes	Bedrohungsaktion, bei der versucht wird, gültigen Benutzern den Zugriff zu verweigern, z. B. indem ein Webserver vorüberge hend nicht verfügbar oder unbrauchbar gemacht wird.	Verfügbarkeit
Erhöhung von Rechten	Bedrohungsaktion, die darauf abzielt, sich privilegi erten Zugriff auf Ressource n zu verschaffen, um sich unbefugten Zugriff auf Informationen zu verschaffen oder ein System zu kompromit tieren.	Autorisierung

AWS sichert seine Verbindungen auf folgende Weise:

1. Berechnung der Anforderungssignatur und anschließende Überprüfung der Signatur auf der Serviceseite. Anforderungen werden mit dieser Signatur authentifiziert.
2. AWS Kunden müssen die richtigen IAM-Rollen einrichten, um bestimmte Aktionen/Operationen zu autorisieren. Diese IAM-Rollen werden benötigt, um Aufrufe für den AWS-Service zu senden.
3. Nur HTTPS-Anfragen an den Service sind zulässig. AWS Anforderungen werden im offenen Netzwerk mit TLS verschlüsselt. Dies schützt die Vertraulichkeit der Anforderungen und gewährleistet die Integrität der Anforderung.
4. AWS Der Service protokolliert ausreichend Daten, um Anrufe von Kunden zu identifizieren. Dadurch werden Ablehnungsangriffe verhindert.
5. AWS Der Service ist verantwortlich für die Aufrechterhaltung einer ausreichenden Verfügbarkeit

Der Kunde ist dafür verantwortlich, seinen Service und seine API-Aufrufe auf folgende Weise zu sichern:

1. Der Kunde muss sicherstellen, dass er sich an einen geeigneten Authentifizierungsmechanismus hält. Es gibt verschiedene Authentifizierungsmechanismen, mit denen eine Anforderung authentifiziert werden kann. Kunden können sich mit Digest-basierter Authentifizierung _OAuth , OpenID Connect und anderen Mechanismen vertraut machen.
2. Kunden müssen sicherstellen, dass ihr Service die richtigen Verschlüsselungskanäle (wie TLS/HTTPS) für Service-API-Aufrufe unterstützt.
3. Kunden müssen sicherstellen, dass sie die Daten protokollieren, die zur eindeutigen Identifizierung eines API-Aufrufs und des Aufrufers erforderlich sind. Sie sollten in der Lage sein, den Client, der ihre API aufruft, anhand definierter Parameter und der Uhrzeit der Aufrufe zu identifizieren.
4. Kunden müssen sicherstellen, dass ihr System verfügbar ist und dass sie vor DDoS-Angriffen geschützt sind. Hier sind einige Beispiele für Verteidigungstechniken gegen DDo S-Angriffe.

Kunden sind für die Aufbewahrung ihrer Anwendungen verantwortlich up-to-date. Weitere Informationen finden Sie unter Richtlinien für das Update von Face Liveness.

Richtlinien für das Update von Face Liveness

AWS aktualisiert regelmäßig Face Liveness AWS SDKs (wird im Kunden-Backend verwendet) und FaceLivenessDetector Komponenten von AWS Amplify SDKs (verwendet in Client-Anwendungen), um neue Funktionen, aktualisierte, verbesserte Sicherheit APIs, Bugfixes, Verbesserungen der Benutzerfreundlichkeit und mehr bereitzustellen. Wir empfehlen Ihnen, die beizubehalten, SDKs up-to-date um ein optimales Funktionieren der Funktion zu gewährleisten. Wenn Sie weiterhin ältere Versionen von verwenden SDKs, können Anfragen aus Wartungs- und Sicherheitsgründen blockiert werden.

Face Liveness erfordert, dass Sie die in AWS Amplify enthaltene FaceLivenessDetector Komponente verwenden SDKs (React, iOS, Android).

Versionsverwaltung und Zeitrahmen

Wir verwalten die Versionen für die folgenden Schlüsselkomponenten des Face-Liveness-Feature. Wir folgen einem semantischen Versionsverwaltungsformat. Zum Beispiel ein Versionsformat von X.Y.Z, wobei X für die Hauptversion, Y für die Nebenversion und Z für die Patch-Version steht.

- Benutzerherausforderungen von Face Liveness (z. B. FaceMovementAndLightChallenge Herausforderung) sind Teil der API StartFaceLivenessSession
- FaceLivenessDetector Komponenten, die über AWS Amplify bereitgestellt SDKs werden, werden in Client-Anwendungen verwendet

Hauptversionen: Wir behalten uns wichtige Versionsupdates für wichtige Sicherheits-, API-Probleme und herausragende Benutzerfreundlichkeitsupdates vor. Die Anwendungen und das Kunden-Backend müssen so schnell wie möglich aktualisiert werden, damit Sie die Features von Face Liveness weiterhin nutzen können. Sobald wir eine neue Hauptversion veröffentlicht haben, unterstützen wir die vorherige Hauptversion ab dem Tag der neuen Version 120 Tage lang. Wir können die Anforderungen, die von der vorherigen Hauptversion kommen, nach 120 Tagen blockieren.

Nebenversionen: Wir behalten uns kleinere Versionsupdates für wichtige Sicherheits- und Benutzerfreundlichkeitsfeatures und Verbesserungen vor. Wir empfehlen dringend, diese Updates zu installieren. Obwohl wir uns bemühen, sicherzustellen, dass kleinere Updates so lange wie möglich abwärtskompatibel sind, können wir 180 Tage nach der Veröffentlichung einer neuen Nebenversion eine vorherige Nebenversion ankündigen end-of-support.

Patch-Versionen: Wir behalten uns Patch-Versionsupdates für optionale Bugfixes und Verbesserungen vor. Wir empfehlen Ihnen zwar, Ihre Version beizubehalten, up-to-date um die beste Sicherheit und Benutzererfahrung zu gewährleisten, wir bemühen uns jedoch sicherzustellen, dass Patch-Updates vollständig abwärtskompatibel sind, bis wir eine neue Haupt- oder Nebenversion veröffentlichen.

Das Zeitfenster für die Versionsverwaltung (120 Tage für Hauptversionen und 180 Tage für Nebenversionen) gilt für die Aktualisierung des SDK in Ihrer App, das Hochladen Ihrer App in den App Store oder auf die Website und das Herunterladen der neuesten Version der App durch Benutzer.

Versionsveröffentlichung und Kompatibilitätsmatrix

Die Veröffentlichung einer Hauptversion als FaceLivenessDetector Komponente oder als Benutzerproblem fällt häufig zusammen. Um Ihnen zu helfen, den Überblick über die Versionsabhängigkeiten zu behalten, sehen Sie sich die in den folgenden Tabellen verlinkten Ressourcen an.

SDK-Versionen und Änderungsprotokolle:

FaceLivenessDetector für das Web-SDK		FaceLivenessDetector für iOS SDK	FaceLivenessDetector für Android SDK
Aktuelle Version	Changelog	Aktuelle Version/Changelog	Aktuelle Version/Changelog

Herausforderungen für Benutzer:

Name der Herausforderung	Version	Datum der Veröffentlichung	Datum der Pensionierung
FaceMovementAndLightChallenge	v1.0.0	10.04.2023	N/A
FaceMovementChallenge	v1.0.0	30.04.2025	N/A

Kommunikation von Neuversionen

AWS kommuniziert Neuerscheinungen über die folgenden Kanäle:

- E-Mail-Benachrichtigungen zum Servicestatus werden an die Konto-E-Mail-Adresse gesendet, die der Face-Liveness-Konto-ID zugeordnet ist.
- Veröffentlichte Updates AWS SDKs und zugehörige Benachrichtigungen in den jeweiligen GitHub Repos.
- Veröffentlichte Updates für AWS Amplify SDKs und zugehörige Benachrichtigungen in den jeweiligen GitHub Repos.

Wir empfehlen Ihnen, diese Kanäle zu abonnieren, um zu bleiben. up-to-date

Häufig gestellte Fragen zu Face Liveness

In den folgenden FAQs finden Sie Antworten auf häufig gestellte Fragen zu Rekognition Face Liveness.

- Was sind die Ergebnisse einer Face-Livness-Kontrolle?

  Rekognition Face Liveness bietet die folgenden Ausgaben für jede Livness-Kontrolle:

  - Zuverlässigkeitswert: Es wird ein numerischer Wert zwischen 0 und 100 zurückgegeben. Dieser Wert gibt die Wahrscheinlichkeit an, dass das Selfie-Video von einer echten Person stammt und nicht von einem Betrüger, der eine Fälschung verwendet.
  - Bild in hoher Qualität: Aus dem Selfie-Video wird ein einzelnes Bild mit hoher Qualität extrahiert. Dieser Rahmen kann für verschiedene Zwecke verwendet werden, z. B. für Gesichtsvergleiche, Altersschätzungen oder Gesichtssuchen.
  - Auditbilder: Aus dem Selfie-Video werden bis zu vier Bilder zurückgegeben, die für Audit Trails verwendet werden können.
- Entspricht Rekognition Face Liveness den Tests von iBeta Presentation Attack Detection (PAD)?

  Die Presentation Attack Detection (PAD) -Tests von iBeta Quality Assurance werden gemäß ISO/IEC 30107-3 durchgeführt. iBeta ist von NIST/NVLAP akkreditiert, um diesen PAD-Standard zu testen und Ergebnisse zu liefern. Rekognition Face Liveness hat die Konformitätstests von iBeta Presentation Attack Detection (PAD) der Stufen 1 und 2 mit einem perfekten PAD-Ergebnis bestanden. Den Bericht finden Sie auf der iBeta-Webseite hier.
- Wie kann ich einen qualitativ hochwertigen Frame und zusätzliche Frames erhalten?

  Der hochwertige Frame und zusätzliche Frames können je nach Konfiguration Ihrer CreateFaceLivenessSessionAPI-Anfrage als Roh-Bytes zurückgegeben oder in einen von Ihnen angegebenen Amazon S3 S3-Bucket hochgeladen werden.
- Kann ich die Position der ovalen und farbigen Lichter ändern?

  Nein. Die ovale Position und die farbigen Lichter sind Merkmale, die eine höhere Genauigkeit ermöglichen und daher nicht individuell angepasst werden können.
- Kann ich die Benutzeroberfläche gemäß unserer Anwendung anpassen?

  Ja, Sie können die meisten Bildschirmkomponenten wie Design, Farbe, Sprache, Textinhalt und Schriftart an Ihre Anwendung anpassen. Einzelheiten zur Anpassung dieser Komponenten finden Sie in der Dokumentation zu unseren React-, Swift- und Android-UI-Komponenten.
- Kann ich die Countdown-Zeit und die Zeit für ein Gesicht im Oval anpassen?

Nein, die Countdown-Zeit und die Face Fit-Zeit wurden auf der Grundlage umfangreicher interner Studien mit Tausenden von Benutzern im Voraus festgelegt, mit dem Ziel, ein optimales Gleichgewicht zwischen diesen Faktoren zu finden, die eine höhere Genauigkeit und Latenz ermöglichen. Aus diesem Grund können diese Zeiteinstellungen nicht angepasst werden.

- Erfüllen die verschiedenfarbigen Lichter die Richtlinien für Barrierefreiheit?

  Ja, die verschiedenen Farblichter in unserem Produkt entsprechen den Richtlinien für Barrierefreiheit, die in WCAG 2.1 beschrieben sind. Wie bei mehr als 1000 Benutzerkontrollen überprüft wurde, zeigt das Benutzererlebnis ungefähr zwei Farben pro Sekunde an, was der Empfehlung entspricht, die Farben auf drei pro Sekunde zu begrenzen. Dadurch verringert sich die Wahrscheinlichkeit, dass bei der Mehrheit der Bevölkerung epileptische Anfälle ausgelöst werden.

- Passt das SDK die Bildschirmhelligkeit an, um optimale Ergebnisse zu erzielen?

  Das Face Liveness Mobile SDKs (für Android und iOS) passt die Helligkeit automatisch an, wenn die Überprüfung eingeleitet wird. Für das Web-SDK gibt es jedoch Einschränkungen auf Webseiten, die eine automatische Helligkeitsanpassung verhindern. In solchen Fällen erwarten wir, dass die Webanwendung die Endbenutzer anweist, die Bildschirmhelligkeit manuell zu erhöhen, um optimale Ergebnisse zu erzielen.

- Muss es ein Oval sein? Könnten wir andere ähnliche Formen verwenden?

  Nein, Größe, Form und Position des Ovals sind nicht anpassbar. Das spezifische ovale Design wurde sorgfältig ausgewählt, da es effektiv bei der genauen Erfassung und Analyse von Gesichtsbewegungen ist. Daher kann die ovale Form nicht verändert werden.

- Was ist die end-to-end Latenz?

  Wir messen die end-to-end Latenz von dem Zeitpunkt an, zu dem der Benutzer die Aktion startet, die zum Abschluss der Verfügbarkeitsprüfung erforderlich ist, bis zu dem Zeitpunkt, zu dem der Benutzer das Ergebnis erhält (bestanden oder nicht bestanden). Im besten Fall beträgt die Latenz 5 s. Im Durchschnitt erwarten wir, dass es etwa 7 s sind. Im schlimmsten Fall beträgt die Latenz 11 s. Wir stellen fest, dass die end-to-end Latenz schwankt, da sie von folgenden Faktoren abhängt: der Zeit, in der der Benutzer die erforderliche Aktion ausführt (d. h. sein Gesicht in das Oval bewegt), der Netzwerkkonnektivität, der Anwendungslatenz usw.

- Kann ich das Face-Liveness-Feature ohne Amplify SDK verwenden?

Nein, das Amplify SDK ist erforderlich, um das Rekognition-Face-Liveness-Feature zu verwenden.

- Wo finde ich die Fehlerstatus im Zusammenhang mit Face Liveness?

  Die verschiedenen Fehlerstatus von Face Liveness finden Sie hier.

- Face Liveness ist in meiner Region nicht verfügbar. Wie kann ich das Feature nutzen?

  Sie können Face Liveness in jeder Region aufrufen, in der es verfügbar ist, abhängig von Ihrer Datenverkehrsbelastung und der Nähe. Face Liveness ist derzeit in den folgenden AWS Regionen verfügbar:

  - USA Ost (Nord-Virginia)
  - USA West (Oregon)
  - Europa (Irland)
  - Asien-Pazifik (Tokio, Mumbai)

  Auch wenn sich Ihr AWS Konto in einer anderen Region befindet, ist nicht zu erwarten, dass der Latenzunterschied signifikant ist. Sie können hochwertige Selfie-Frame- und Audit-Bilder über den Amazon S3 S3-Standort oder als Rohbytes abrufen, aber Ihr Amazon S3 S3-Bucket muss der AWS Region von Face Liveness entsprechen. Wenn sie unterschiedlich sind, müssen Sie die Bilder als Rohbytes erhalten.

- Verwendet Amazon Rekognition Liveness Detection Kundeninhalte, um den Service zu verbessern?

  Sie können die Verwendung Ihrer Bild- und Videoeingaben zur Verbesserung oder Weiterentwicklung der Qualität von Rekognition und anderen Amazon-Technologien für Machine Learning/künstliche Intelligenz deaktivieren, indem Sie eine AWS -Organisations-Opt-out-Richtlinie verwenden. Informationen darüber, wie Sie sich abmelden können, finden Sie unter Opt-out-Richtlinie für KI-Dienste verwalten.

Massenanalyse

Mit Amazon Rekognition Bulk Analysis können Sie eine große Sammlung von Bildern asynchron verarbeiten, indem Sie für den Vorgang eine Manifestdatei verwenden. StartMediaAnalysisJob Die Ausgabe für jedes einzelne Bild entspricht der Ausgabe, die von der Operation zurückgegeben wurde, den Sie für die Analyse verwenden.

Derzeit unterstützt Rekognition die Analyse bei der DetectModerationLabelsOperation.

Ihnen wird die Anzahl der Bilder in Rechnung gestellt, die im Rahmen des Auftrags erfolgreich verarbeitet wurden. Die Ergebnisse eines abgeschlossenen Auftrags werden in einen angegebenen Amazon-S3-Bucket ausgegeben.

Beachten Sie, dass die Massenanalyse die Amazon-A2I-Integration nicht unterstützt.

Die API kann animierte oder illustrierte Inhaltstypen erkennen, und Informationen über den erkannten Inhaltstyp werden als Teil der Antwort zurückgegeben.

Themen

- Bilder werden in großen Mengen verarbeitet
- Die Ergebnisse der Massenanalyse werden angezeigt
- Inhaltstyp
- Überprüfung von Prognosen und Trainingsadaptern

Bilder werden in großen Mengen verarbeitet

Sie können einen neuen Massenanalyseauftrag starten, indem Sie eine Manifestdatei einreichen und den StartMediaAnalysisJob Vorgang aufrufen. Die Eingabe-Manifestdatei enthält Verweise auf Bilder in einem Amazon-S3-Bucket und ist wie folgt formatiert:

```
{"source-ref": "s3://foo/bar/1.jpg"}
```

So erstellen Sie einen Massenanalyseauftrag (CLI)

1. Wenn Sie dies noch nicht getan haben:

a. Erstellen oder aktualisieren Sie einen Benutzer mit `AmazonRekognitionFullAccess`- und `AmazonS3ReadOnlyAccess`-Berechtigungen. Weitere Informationen finden Sie unter Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Benutzers.

b. Installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Laden Sie Bilder in Ihren S3-Bucket hoch.

   Weitere Anleitungen finden Sie unter Upload eines Objekts in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service.

3. Verwenden Sie die folgenden Befehle, um Massenanalyseaufträge zu erstellen und abzurufen.

CLI

Verwenden Sie den folgenden Befehl, um die StartMediaAnalysisJobOperation zur Analyse mit der DetectModerationLabels Operation aufzurufen:

```
# Requests
# Starting DetectModerationLabels job with default settings
aws rekognition start-media-analysis-job \
--operations-config "DetectModerationLabels={MinConfidence='1'}" \
--input "S3Object={Bucket=amzn-s3-demo-source-bucket,Name=my-input.jsonl}" \
--output-config "S3Bucket=amzn-s3-demo-destination-bucket;,S3KeyPrefix=my-results"
```

Mithilfe des GetMediaAnalysisJobVorgangs können Sie Informationen zu einem bestimmten Job abrufen, z. B. den Amazon S3 S3-Pfad des Buckets, in dem Ergebnisse und Übersichtsdateien gespeichert sind. Sie geben ihm eine Job-ID, die von StartMediaAnalysisJob oder zurückgegeben wurde ListMediaAnalysisJob. Details zu einzelnen Aufträgen werden nur für ein Jahr aufbewahrt.

```
# Request
aws rekognition get-media-analysis-job \
--job-id customer-job-id
```

Sie können alle Ihre Massenanalysen auflisten, indem Sie den ListMediaAnalysisJobsJobvorgang verwenden, der Seiten mit Aufträgen zurückgibt. Mit dem `max-results`-Argument können Sie die maximale Anzahl von Aufträgen angeben, die pro Seite zurückgegeben werden sollen, begrenzt auf den Wert von `max-results`. Pro Seite werden maximal 100 Ergebnisse zurückgegeben. Details zu einzelnen Aufträgen werden nur für ein Jahr aufbewahrt.

```
# Request
# Specify number of jobs to return per page, limited to max-results.
aws rekognition list-media-analysis-jobs --max-results 1
```

Die Ergebnisse der Massenanalyse werden angezeigt

Der Massenanalyseauftrag generiert eine Ausgabe-Manifestdatei, die die Auftragsergebnisse sowie eine Manifestzusammenfassung enthält, die Statistiken und Details zu Fehlern bei der Verarbeitung der Eingabemanifesteinträge enthält.

Wenn doppelte Einträge in das Eingabemanifest aufgenommen wurden, versucht der Auftrag nicht, eindeutige Eingaben herauszufiltern, sondern verarbeitet stattdessen alle bereitgestellten Einträge.

Die Ausgabemanifestdatei ist wie folgt formatiert:

```
// 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"}]}}
```

Die Zusammenfassung des Ausgabemanifests ist wie folgt formatiert:

```
{
    "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.
        }
```

```
    ]
}
```

Inhaltstyp

Informationen über die Art des Medieninhalts, der durch den Vorgang analysiert wurde, werden vom StartMediaAnalysisJob Vorgang zurückgegeben. GetMediaAnalysisJob ContentType kann eine von zwei verschiedenen Kategorien sein:

- Animierte Inhalte, einschließlich Videospiele und Animationen (z. B. Zeichentrick, Comics, Manga, Anime).
- Illustrierter Inhalt, der Zeichnen, Malen und Skizzen umfasst.

Überprüfung von Prognosen und Trainingsadaptern

Die Massenanalyse kann auch über die Rekognition-Konsole genutzt werden, um Vorhersagen für einen Stapel von Bildern zu erhalten, diese Vorhersagen zu überprüfen und dann anhand der verifizierten Vorhersagen einen Adapter zu erstellen. Mit Adaptern können Sie die Genauigkeit aller unterstützten Rekognition-Operationen verbessern.

Derzeit können Sie Adapter für die Verwendung mit des Features Rekognition Custom Moderation erstellen. Indem Sie einen Adapter erstellen und für den DetectModerationLabelsVorgang bereitstellen, können Sie die Aufgaben der Inhaltsmoderation, die sich auf Ihren speziellen Anwendungsfall beziehen, genauer ausführen.

Weitere Informationen zu Custom Moderation finden Sie unter Verbesserung der Genauigkeit mit benutzerdefinierter Moderation. Eine Erläuterung, wie Sie mithilfe der Massenanalyse getroffene Vorhersagen verifizieren können, finden Sie unter Massenanalyse und -überprüfung. Eine Anleitung zur Verwendung der Rekognition-Konsole zur Überprüfung von Vorhersagen und zur Erstellung eines Adapters finden Sie unter Tutorial zum benutzerdefinierten Moderationsadapter.

Tutorials

Diese serviceübergreifenden Tutorials zeigen, wie Sie die API-Operationen von Rekognition zusammen mit anderen AWS Diensten verwenden können, um Beispielanwendungen zu erstellen und eine Vielzahl von Aufgaben zu erledigen. Die meisten dieser Tutorials verwenden Amazon S3 zum Speichern von Bildern oder Videos. Andere häufig verwendete Dienste umfassen AWS Lambda.

Themen

- Speichern von Amazon-Rekognition-Daten mit Amazon RDS und DynamoDB
- Verwenden von Amazon Rekognition und Lambda zum Taggen von Assets in einem Amazon-S3-Bucket
- AWS Videoanalysator-Anwendungen erstellen
- Erstellen einer Lambda-Funktion für Amazon Rekognition
- Amazon Rekognition zur Identitätsprüfung verwenden
- Verwendung von Amazon Rekognition (REK) zur Erkennung von Etiketten für Marketinganwendungen
- Label in einem Bild mit Lambda und Python erkennen

Speichern von Amazon-Rekognition-Daten mit Amazon RDS und DynamoDB

Bei der Verwendung von Amazon Rekognition ist es wichtig APIs, sich daran zu erinnern, dass die API-Operationen keine der generierten Labels speichern. Sie können diese Labels speichern, indem Sie sie zusammen mit den Kennungen für die jeweiligen Bilder in der Datenbank platzieren.

Dieses Tutorial zeigt das Erkennen von Labels und das Speichern dieser erkannten Labels in einer Datenbank. Die in diesem Tutorial entwickelte Beispielanwendung liest Bilder aus einem Amazon S3 S3-Bucket, ruft den DetectLabelsVorgang für diese Bilder auf und speichert die resultierenden Labels in einer Datenbank. Die Anwendung speichert Daten entweder in einer Amazon-RDS-Datenbank-Instance oder einer DynamoDB-Datenbank, je nachdem, welchen Datenbanktyp Sie verwenden möchten.

Für dieses Tutorial verwenden Sie das AWS -SDK für Python. Weitere Python-Tutorials finden Sie auch im AWS Documentation SDK Examples GitHub Repo.

Themen

- Voraussetzungen
- Labels für Bilder in einem Amazon-S3-Bucket abrufen
- Erstellen einer Amazon DynamoDB-Tabelle
- Hochladen von Daten in DynamoDB
- Eine MySQL-Datenbank in Amazon RDS erstellen
- Daten in eine Amazon-RDS-MySQL-Tabelle hochladen

Voraussetzungen

Bevor Sie mit diesem Tutorial beginnen, müssen Sie Python installieren und die zur Einrichtung des AWS Python-SDK erforderlichen Schritte ausführen. Stellen Sie darüber hinaus sicher, dass Sie über Folgendes verfügen:

Sie haben ein AWS-Konto und eine IAM-Rolle erstellt

Sie haben das Python-SDK (Boto3) installiert

Haben Sie Ihre AWS Zugangsdaten richtig konfiguriert

Sie haben den Amazon-S3-Bucket erstellt und ihn mit Bildern gefüllt

Es wurde eine RDS-Datenbank-Instance erstellt, wenn RDS zum Speichern von Daten verwendet wurde

Labels für Bilder in einem Amazon-S3-Bucket abrufen

Schreiben Sie zunächst eine Funktion, die den Namen eines Bilds in Ihrem Amazon-S3-Bucket übernimmt und dieses Bild abruft. Dieses Bild wird angezeigt, um zu bestätigen, dass die richtigen Bilder an einen Aufruf übergeben werden DetectLabels, der auch in der Funktion enthalten ist.

1. Suchen Sie den Amazon-S3-Bucket, den Sie verwenden möchten, und notieren Sie sich seinen Namen. Sie rufen diesen Amazon-S3-Bucket auf und lesen die darin enthaltenen Bilder. Stellen Sie sicher, dass Ihr Bucket einige Bilder enthält, die an den DetectLabelsVorgang übergeben werden sollen.
2. Schreiben Sie den Code für die Verbindung mit Ihrem Amazon-S3-Bucket. Sie können mit Boto3 eine Verbindung zur Amazon-S3-Ressource herstellen, um ein Bild aus einem Amazon-S3-Bucket abzurufen. Sobald eine Verbindung mit der Amazon-S3-Ressource hergestellt ist, können

Sie auf Ihren Bucket zugreifen, indem Sie der Bucket-Methode den Namen Ihres Amazon-S3-Buckets angeben. Nachdem Sie eine Verbindung zum Amazon-S3-Bucket hergestellt haben, rufen Sie mithilfe der Objekt-Methode Bilder aus dem Bucket ab. Mithilfe von Matplotlib können Sie diese Verbindung verwenden, um Ihre Bilder während der Verarbeitung zu visualisieren. Boto3 wird auch verwendet, um eine Verbindung zum Rekognition-Client herzustellen.

Geben Sie im folgenden Code Ihre Region für den Parameter region_name an. Sie übergeben den Amazon S3 S3-Bucket-Namen und den Image-Namen an DetectLabels, der die Labels für das entsprechende Bild zurückgibt. Nachdem Sie nur die Labels aus der Antwort ausgewählt haben, werden sowohl der Name des Bildes als auch die Labels zurückgegeben.

```
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. Speichern Sie diesen Code in einer Datei namens get_images.py.

Erstellen einer Amazon DynamoDB-Tabelle

Der folgende Code verwendet Boto3, um eine Verbindung zu DynamoDB herzustellen, und verwendet die `CreateTable`-DynamoDB-Methode, um eine Tabelle mit dem Namen „Bilder“ zu erstellen. Die Tabelle hat einen zusammengesetzten Primärschlüssel, der aus einem Partitionsschlüssel namens Image und einem Sortierschlüssel namens Labels besteht. Der Bildschlüssel enthält den Namen des Bilds, während der Labelschlüssel die diesem Bild zugewiesenen Labels speichert.

```
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)
```

Speichern Sie diesen Code in einem Editor und führen Sie ihn einmal aus, um eine DynamoDB-Tabelle zu erstellen.

Hochladen von Daten in DynamoDB

Nachdem die DynamoDB-Datenbank erstellt wurde und Sie über eine Funktion zum Abrufen von Labels für Bilder verfügen, können Sie die Labels in DynamoDB speichern. Der folgende Code ruft alle Bilder in einem S3-Bucket ab, ruft Labels für sie ab und speichert die Daten in DynamoDB.

1. Sie müssen den Code für das Hochladen der Daten auf DynamoDB schreiben. Eine Funktion namens `get_image_names` wird verwendet, um eine Verbindung zu Ihrem Amazon-S3-Bucket herzustellen. Sie gibt die Namen aller Bilder im Bucket als Liste zurück. Sie übergeben diese Liste an die `read_image_from_S3`-Funktion, die aus der von Ihnen erstellten `get_images.py`-Datei importiert wird.

   ```
   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. Die `read_image_from_S3`-Funktion, die wir zuvor erstellt haben, gibt den Namen des zu verarbeitenden Bilds und das Wörterbuch der mit diesem Bild verknüpften Labels zurück. Eine Funktion namens `find_values` wird verwendet, um nur die Labels aus der Antwort abzurufen. Der Name des Bilds und seine Labels können dann in Ihre DynamoDB-Tabelle hochgeladen werden.

```
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. Sie verwenden eine dritte Funktion namens `load_data`, um die Bilder und Labels tatsächlich in die von Ihnen erstellte DynamoDB-Tabelle zu laden.

```
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. Hier werden die drei Funktionen aufgerufen, die wir zuvor definiert haben, und die Operationen werden ausgeführt. Fügen Sie die drei oben definierten Funktionen zusammen mit dem folgenden Code zu einer Python-Datei hinzu. Führen Sie den Code aus.

```
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!")
```

Sie haben gerade DetectLabelsLabels für Ihre Bilder generiert und diese Labels in einer DynamoDB-Instanz gespeichert. Stellen Sie sicher, dass Sie alle Ressourcen, die Sie während der Bearbeitung dieses Tutorials erstellt haben, beenden. Dadurch wird verhindert, dass Ihnen Ressourcen in Rechnung gestellt werden, die Sie nicht nutzen.

Eine MySQL-Datenbank in Amazon RDS erstellen

Bevor Sie fortfahren, stellen Sie sicher, dass Sie den Einrichtungsvorgang für Amazon RDS abgeschlossen und eine MySQL-DB-Instance mit Amazon RDS erstellt haben.

Der folgende Code verwendet die PyMySQL-Bibliothek und Ihre Amazon RDS-DB-Instance. Er erstellt eine Tabelle, die die Namen Ihrer Bilder und die mit diesen Bildern verknüpften Labels enthält. Amazon RDS empfängt Befehle zum Erstellen von Tabellen und zum Einfügen von Daten in Tabellen. Um Amazon RDS verwenden zu können, müssen Sie sich mit Ihrem Hostnamen, Benutzernamen und Passwort mit dem Amazon-RDS-Host verbinden. Sie stellen eine Verbindung zu Amazon RDS her, indem Sie diese Argumente für die PyMy `connect` SQL-Funktion bereitstellen und eine Instanz eines Cursors erstellen.

1. Ersetzen Sie im folgenden Code den Wert von Host durch Ihren Amazon-RDS-Host-Endpunkt und den Wert von Benutzer durch den Master-Benutzernamen, der Ihrer Amazon-RDS-Instance zugeordnet ist. Außerdem müssen Sie das Passwort durch das Master-Passwort für Ihren Hauptbenutzer ersetzen.

   ```
   import pymysql
   ```

```
host = "host-endpoint"
user = "username"
password = "master-password"
```

2. Erstellen Sie eine Datenbank und eine Tabelle, in die Sie Ihre Bild- und Labeldaten einfügen können. Führen Sie dazu eine Erstellungsabfrage aus und bestätigen Sie sie. Mit dem folgenden Code wird eine Datenbank erstellt. Führen Sie diesen Code nur einmal aus.

```
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. Sobald die Datenbank erstellt wurde, müssen Sie eine Tabelle erstellen, in die Sie Ihre Bildnamen und Labels einfügen können. Um eine Tabelle zu erstellen, übergeben Sie zunächst den Befehl „use SQL“ zusammen mit dem Namen Ihrer Datenbank an die `execute`-Funktion. Nachdem die Verbindung hergestellt wurde, wird eine Abfrage zum Erstellen einer Tabelle ausgeführt. Der folgende Code stellt eine Verbindung mit der Datenbank her und erstellt dann eine Tabelle mit einem Primärschlüssel namens `image_id`, der aufgerufen wird, und einem Textattribut, in dem die Labels gespeichert sind. Verwenden Sie die Importe und Variablen, die Sie zuvor definiert haben, und führen Sie diesen Code aus, um eine Tabelle in Ihrer Datenbank zu erstellen.

```
# 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!")
```

Daten in eine Amazon-RDS-MySQL-Tabelle hochladen

Nachdem Sie die Amazon-RDS-Datenbank und eine Tabelle in der Datenbank erstellt haben, können Sie Labels für Ihre Bilder abrufen und diese Labels in der Amazon-RDS-Datenbank speichern.

1. Stellen Sie eine Verbindung zu Ihrem Amazon-S3-Bucket her und rufen Sie die Namen aller Bilder im Bucket ab. Diese Bildnamen werden an die `read_image_from_s3`-Funktion übergeben, die Sie zuvor erstellt haben, um die Labels für alle Ihre Bilder abzurufen. Der folgende Code stellt eine Verbindung zu Ihrem Amazon-S3-Bucket her und gibt eine Liste aller Bilder in Ihrem Bucket zurück.

```
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. Die Antwort von der DetectLabelsAPI enthält mehr als nur die Labels. Schreiben Sie also eine Funktion, um nur die Labelwerte zu extrahieren. Die folgende Funktion gibt eine Liste zurück, die nur die Labels enthält.

```
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. Sie benötigen eine Funktion, um die Bildnamen und Labels in Ihre Tabelle einzufügen. Die folgende Funktion führt eine Einfügeabfrage aus und fügt ein beliebiges Paar von Bildnamen und Labels ein.

```
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. Schließlich müssen Sie die Funktionen ausführen, die Sie oben definiert haben. Im folgenden Code werden die Namen aller Bilder in Ihrem Bucket gesammelt und der aufrufenden Funktion zur Verfügung gestellt DetectLabels. Anschließend werden die Labels und der Name des Bilds, auf das sie zutreffen, in Ihre Amazon-RDS-Datenbank hochgeladen. Kopieren Sie die drei oben definierten Funktionen zusammen mit dem folgenden Code in eine Python-Datei. Führen Sie die Python-Datei aus.

```
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!")
```

Sie haben erfolgreich Labels für Ihre Bilder generiert und diese Labels mithilfe von Amazon RDS in einer MySQL-Datenbank gespeichert. DetectLabels Stellen Sie sicher, dass Sie alle Ressourcen, die Sie während der Bearbeitung dieses Tutorials erstellt haben, beenden. Dadurch wird verhindert, dass Ihnen Ressourcen in Rechnung gestellt werden, die Sie nicht nutzen.

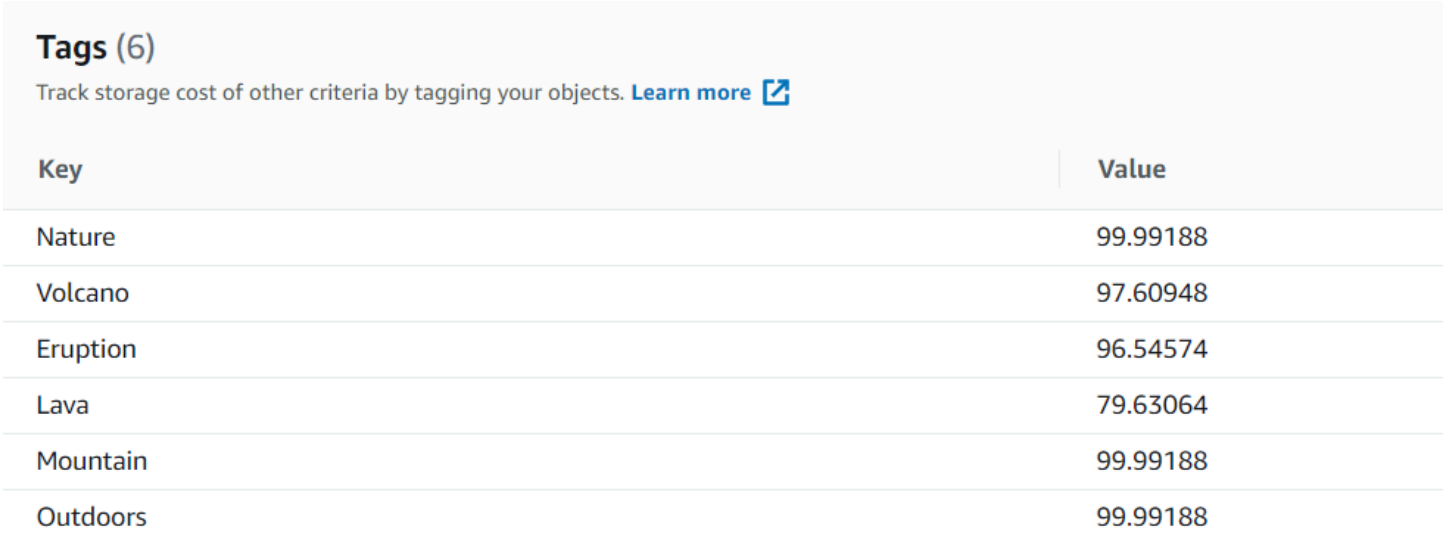
Weitere AWS Multiservice-Beispiele finden Sie im AWS Documentation SDK Examples GitHub Repository.

Verwenden von Amazon Rekognition und Lambda zum Taggen von Assets in einem Amazon-S3-Bucket

In diesem Tutorial erstellen Sie eine AWS Lambda Funktion, die digitale Assets, die sich in einem Amazon S3-Bucket befinden, automatisch kennzeichnet. Die Lambda-Funktion liest alle Objekte aus einem bestimmten Amazon-S3-Bucket. Für jedes Objekt im Bucket wird das Bild an den Amazon-Rekognition-Service übergeben, um eine Reihe von Labels zu generieren. Jedes Label wird verwendet, um ein Tag zu erstellen, das auf das Bild angewendet wird. Nachdem Sie die Lambda-Funktion ausgeführt haben, erstellt sie automatisch Tags, die auf allen Bildern in einem bestimmten Amazon-S3-Bucket basieren, und wendet sie auf die Bilder an.

Nehmen wir zum Beispiel an, Sie führen die Lambda-Funktion aus und haben dieses Bild in einem Amazon-S3-Bucket.

Die Anwendung erstellt dann automatisch Tags und wendet sie auf das Bild an.



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**
>
> Die Dienste, die Sie in diesem Tutorial nutzen, sind Teil des AWS kostenlosen Kontingents. Wenn Sie mit dem Tutorial fertig sind, empfehlen wir, alle Ressourcen, die Sie während des Tutorials erstellt haben, zu beenden, damit Ihnen nichts berechnet wird.

Dieses Tutorial verwendet das AWS SDK for Java Version 2. Weitere Anleitungen zu Java V2 finden Sie im AWS Documentation SDK Examples GitHub Repository.

Themen

- Voraussetzungen
- Konfigurieren der IAM-Lambda-Rolle
- Erstellen des Projekts
- Schreiben des Codes
- Projekt verpacken
- Stellen Sie die Lambda-Funktion bereit
- Lambda-Methode testen

Voraussetzungen

Bevor Sie beginnen, müssen Sie die Schritte unter Einrichten des AWS SDK for Java ausführen. Dann stellen Sie sicher, dass Sie Folgendes haben:

- Java 1.8 JDK.
- Maven 3.6 oder höher.
- Ein Amazon-S3-Bucket mit 5 bis 7 Naturbildern. Diese Bilder werden von der Lambda-Funktion gelesen.

Konfigurieren der IAM-Lambda-Rolle

In diesem Tutorial werden die Dienste Amazon Rekognition und Amazon S3 verwendet. Konfigurieren Sie die lambda-support-Rolle so, dass sie über Richtlinien verfügt, die es ihr ermöglichen, diese Dienste von einer Lambda-Funktion aus aufzurufen.

Konfigurieren Sie die Rolle wie folgt:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Rollen und dann Rolle erstellen aus.
3. Wählen Sie AWS -Service und anschließend Lambda aus.
4. Wählen Sie die Registerkarte Berechtigungen.
5. Suchen Sie nach AWSLambdaBasicExecutionRole.
6. Wählen Sie Nächste Tags.
7. Wählen Sie Überprüfen.
8. Nennen Sie die Rolle lambda-support.
9. Wählen Sie Rolle erstellen aus.
10. Wählen Sie lambda-support, um die Übersichtsseite aufzurufen.
11. Wählen Sie Richtlinien anfügen.
12. Wählen Sie AmazonRekognitionFullAccessaus der Liste der Richtlinien aus.
13. Wählen Sie Richtlinie anfügen aus.
14. Suchen Sie nach AmazonS3 FullAccess und wählen Sie dann Attach policy aus.

Erstellen des Projekts

Erstellen Sie ein neues Java-Projekt und konfigurieren Sie dann die Datei Maven pom.xml mit den erforderlichen Einstellungen und Abhängigkeiten. Stellen Sie sicher, dass Ihre pom.xml-Datei wie folgt aussieht:

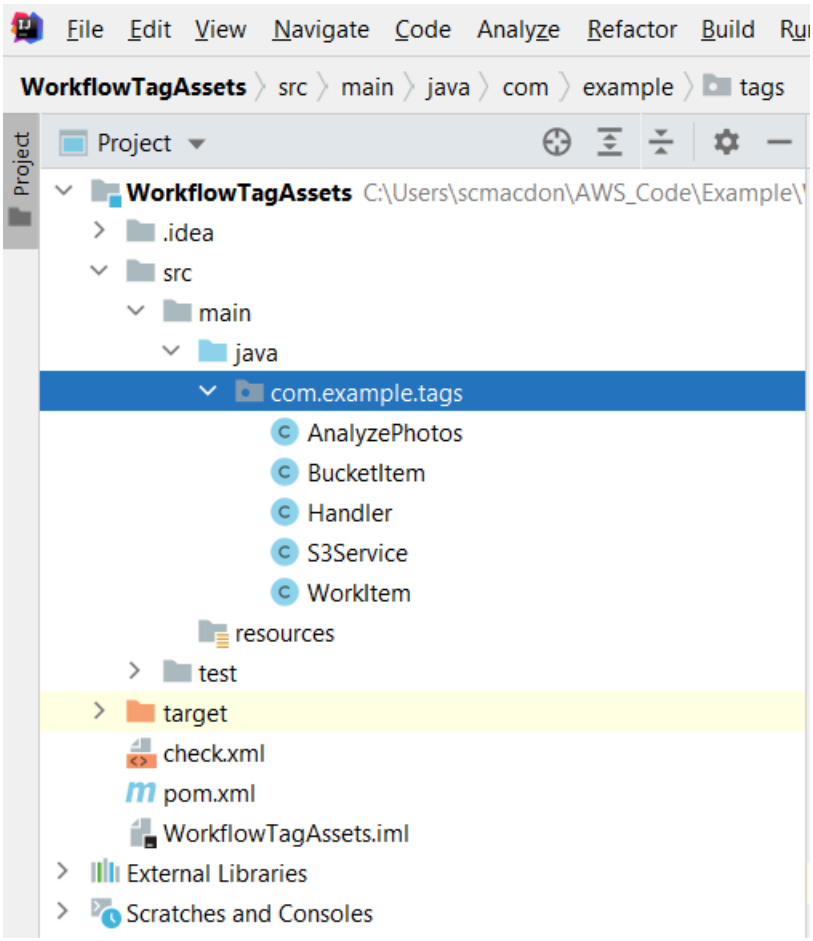
```
<?xml version="1.0" encoding="UTF-8"?>
 <project xmlns="http://maven.apache.org/POM/4.0.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>org.example</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>
```

Schreiben des Codes

Verwenden Sie die AWS Lambda Runtime-Java-API, um die Java-Klasse zu erstellen, die die Lambda-Funktion definiert. In diesem Beispiel gibt es eine Java-Klasse für die Lambda-Funktion namens Handler und zusätzliche Klassen, die für diesen Anwendungsfall erforderlich sind. In der folgenden Abbildung sind die Java-Klassen im Projekt dargestellt. Beachten Sie, dass sich alle Java-Klassen in einem Paket mit dem Namen com.example.tags befinden.



Erstellen Sie die folgenden Java-Klassen für den Code:

- Handler verwendet die Lambda-Java-Laufzeit-API und führt den in diesem Tutorial beschriebenen Anwendungsfall durch. AWS Die ausgeführte Anwendungslogik befindet sich in der Methode handleRequest.
- S3Service verwendet die Amazon-S3-API, um S3-Operationen durchzuführen.
- AnalyzePhotosverwendet die Amazon Rekognition API, um die Bilder zu analysieren.
- BucketItemdefiniert ein Modell, das Amazon S3 S3-Bucket-Informationen speichert.
- WorkItemdefiniert ein Modell, das Amazon Rekognition Rekognition-Daten speichert.

Handler-Klasse

Dieser Java-Code repräsentiert die Handler-Klasse. Die Klasse liest ein Flag, das an die Lambda-Funktion übergeben wird. Der S3-Service. ListBucketObjectsDie Methode gibt ein List-Objekt zurück, wobei jedes Element ein Zeichenkettenwert ist, der den Objektschlüssel darstellt. Wenn der Flag-

Wert wahr ist, werden Tags angewendet, indem durch die Liste iteriert und Tags auf jedes Objekt angewendet werden, indem die Methode s3Service.tagAssets aufgerufen wird. Wenn der Flag-Wert falsch ist, dann der S3Service. deleteTagFromEs wird eine Objektmethode aufgerufen, die die Tags löscht. Beachten Sie auch, dass Sie Nachrichten mithilfe eines LambdaLoggerObjekts in CloudWatch Amazon-Protokollen protokollieren können.

> **Note**
>
> Stellen Sie sicher, dass Sie Ihren Bucket-Namen der Variablen bucketName zuweisen.

```
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;
  }
}
```

S3Service-Klasse

Die folgende Klasse verwendet die Amazon-S3-API, um S3-Operationen durchzuführen. Die getObjectBytesMethode gibt beispielsweise ein Byte-Array zurück, das das Bild darstellt. Ebenso gibt die listBucketObjectsMethode ein List-Objekt zurück, bei dem jedes Element ein Zeichenkettenwert ist, der den Schlüsselnamen angibt.

```
package com.example.tags;

import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.services.s3.model.Tagging;
```

```
import software.amazon.awssdk.services.s3.model.Tag;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectTaggingRequest;

public class S3Service {

private S3Client getClient() {

   Region region = Region.US_WEST_2;
   return S3Client.builder()
           .region(region)
           .build();
}

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

   S3Client s3 = getClient();

   try {

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

       // Return the byte[] from this object.
       ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
       return objectBytes.asByteArray();

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

// Returns the names of all images in the given bucket.
public List<String> listBucketObjects(String bucketName) {

   S3Client s3 = getClient();
   String keyName;
```

```
    List<String> keys = new ArrayList<>();

    try {
        ListObjectsRequest listObjects = ListObjectsRequest
                .builder()
                .bucket(bucketName)
                .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();

        for (S3Object myValue: objects) {
            keyName = myValue.key();
            keys.add(keyName);
        }
        return keys;

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

// Tag assets with labels in the given list.
public void tagAssets(List myList, String bucketName) {

    try {

        S3Client s3 = getClient();
        int len = myList.size();

        String assetName = "";
        String labelName = "";
        String labelValue = "";

        // Tag all the assets in the list.
        for (Object o : myList) {

            // Need to get the WorkItem from each list.
            List innerList = (List) o;
            for (Object value : innerList) {

                WorkItem workItem = (WorkItem) value;
```

```
                assetName = workItem.getKey();
                labelName = workItem.getName();
                labelValue = workItem.getConfidence();
                tagExistingObject(s3, bucketName, assetName, labelName, labelValue);
            }
        }

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

// This method tags an existing object.
private void tagExistingObject(S3Client s3, String bucketName, String key, String
label, String LabelValue) {

    try {

        // First need to get existing tag set; otherwise the existing tags are
overwritten.
        GetObjectTaggingRequest getObjectTaggingRequest =
GetObjectTaggingRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        GetObjectTaggingResponse response =
s3.getObjectTagging(getObjectTaggingRequest);

        // Get the existing immutable list - cannot modify this list.
        List<Tag> existingList = response.tagSet();
        ArrayList<Tag> newTagList = new ArrayList(new ArrayList<>(existingList));

        // Create a new tag.
        Tag myTag = Tag.builder()
                .key(label)
                .value(LabelValue)
                .build();

        // push new tag to list.
        newTagList.add(myTag);
        Tagging tagging = Tagging.builder()
                .tagSet(newTagList)
```

```
                .build();

        PutObjectTaggingRequest taggingRequest = PutObjectTaggingRequest.builder()
                .key(key)
                .bucket(bucketName)
                .tagging(tagging)
                .build();

        s3.putObjectTagging(taggingRequest);
        System.out.println(key + " was tagged with " + label);

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

 // Delete tags from the given object.
 public void deleteTagFromObject(String bucketName, String key) {

    try {

        DeleteObjectTaggingRequest deleteObjectTaggingRequest =
 DeleteObjectTaggingRequest.builder()
                .key(key)
                .bucket(bucketName)
                .build();

        S3Client s3 = getClient();
        s3.deleteObjectTagging(deleteObjectTaggingRequest);

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

AnalyzePhotos Klasse

Der folgende Java-Code repräsentiert die AnalyzePhotosKlasse. Diese Klasse verwendet die Amazon-Rekognition-API, um die Bilder zu analysieren.

```
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 Klasse

Der folgende Java-Code stellt die BucketItemKlasse dar, die Amazon S3 S3-Objektdaten speichert.

```
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 Klasse

Der folgende Java-Code repräsentiert die WorkItemKlasse.

```
 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;
}
```

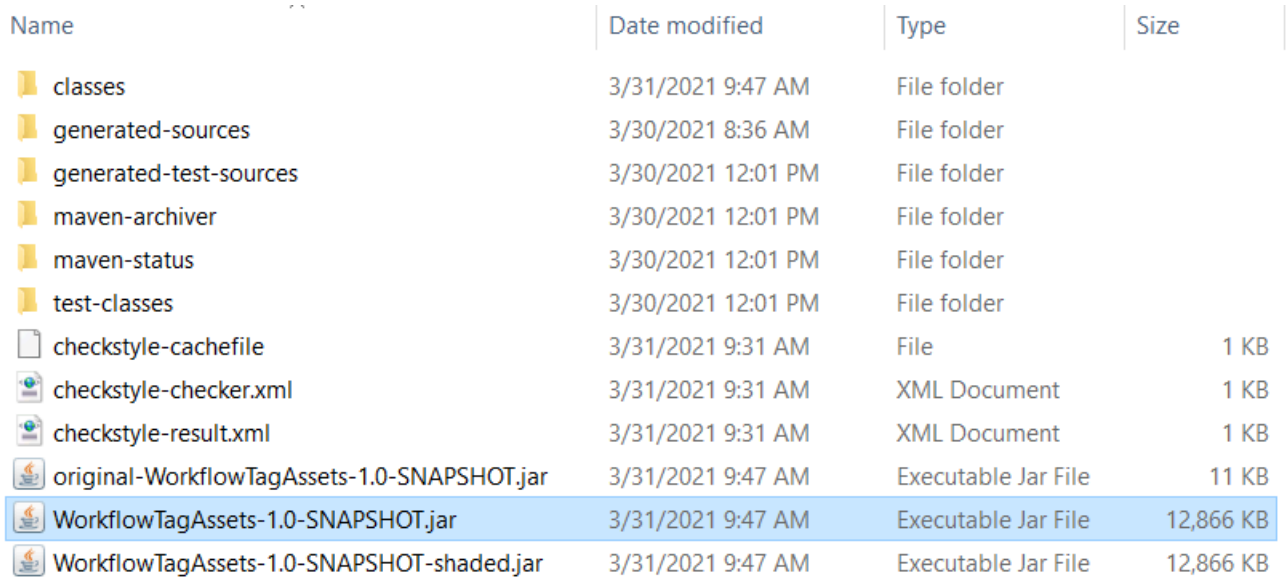
```
public void setName (String name) {
    this.name = name;
}

public String getName() {
    return this.name;
}

public void setConfidence (String confidence) {
    this.confidence = confidence;
}

public String getConfidence() {
    return this.confidence;
}

}
```

Projekt verpacken

Package Sie das Projekt mithilfe des folgenden Maven-Befehls in eine JAR-Datei (JAR).

```
mvn package
```

Die JAR-Datei befindet sich im Zielordner (der ein untergeordneter Ordner des Projektordners ist).



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**
>
> Beachten Sie die Verwendung von maven-shade-pluginin der POM-Datei des Projekts. Dieses Plug-In ist dafür verantwortlich, ein JAR zu erstellen, das die erforderlichen Abhängigkeiten enthält. Wenn Sie versuchen, das Projekt ohne dieses Plugin zu verpacken, sind die erforderlichen Abhängigkeiten nicht in der JAR-Datei enthalten und Sie werden auf eine stoßen. ClassNotFoundException

Stellen Sie die Lambda-Funktion bereit

1. Öffnen Sie die Lambda-Konsole.
2. Wählen Sie Funktion erstellen.
3. Wählen Sie Von Grund auf neu schreiben aus.
4. Geben Sie im Abschnitt Grundinformationen cron als Namen ein.
5. Wählen Sie in Laufzeit die Option Java 8 aus.
6. Wählen Sie Vorhandene Rolle verwenden und wählen Sie dann lambda-support (die von Ihnen erstellte IAM-Rolle).
7. Wählen Sie Funktion erstellen aus.
8. Wählen Sie für Codeeingabetyp die Option ZIP- oder JAR-Datei hochladen aus.
9. Wählen Sie Hochladen aus und suchen Sie dann die JAR-Datei, die Sie erstellt haben.
10. Geben Sie für Handler den vollqualifizierten Namen der Funktion ein, z. B. com.example.tags.handler:HandleRequest (com.example.tags gibt das Paket an, Handler ist die Klasse, gefolgt von :: und Methodenname).
11. Wählen Sie Save (Speichern) aus.

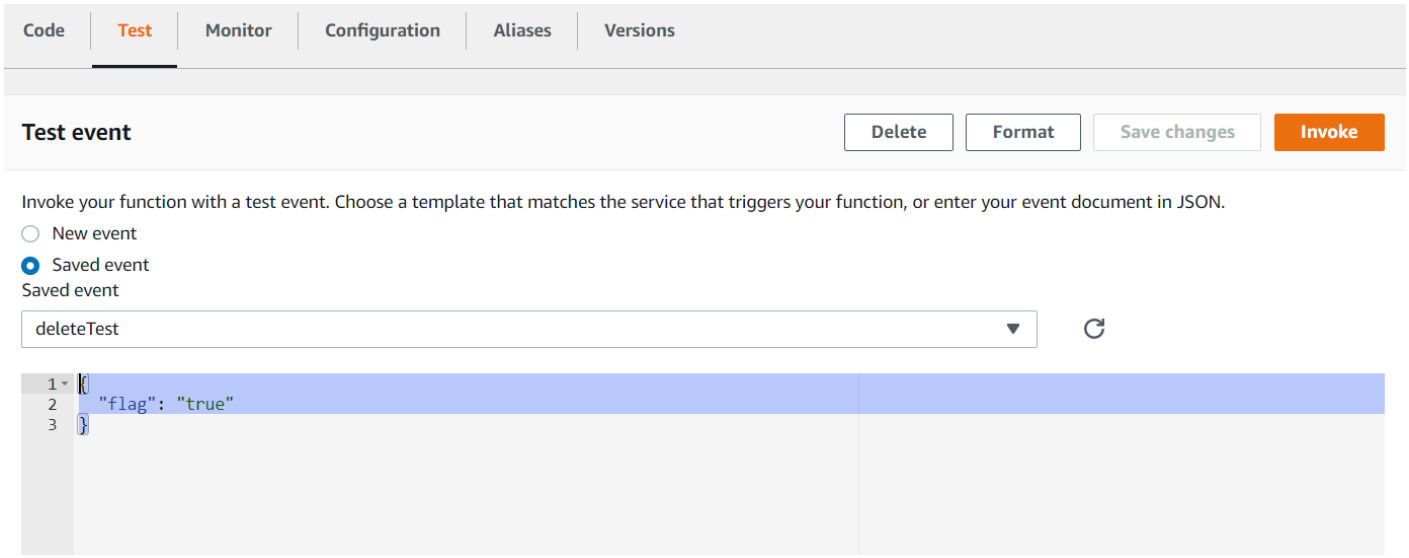
Lambda-Methode testen

An diesem Punkt des Tutorials können Sie die Lambda-Funktion testen.

1. Klicken Sie in der Lambda-Konsole auf die Registerkarte Test und geben Sie dann den folgenden JSON-Code ein.

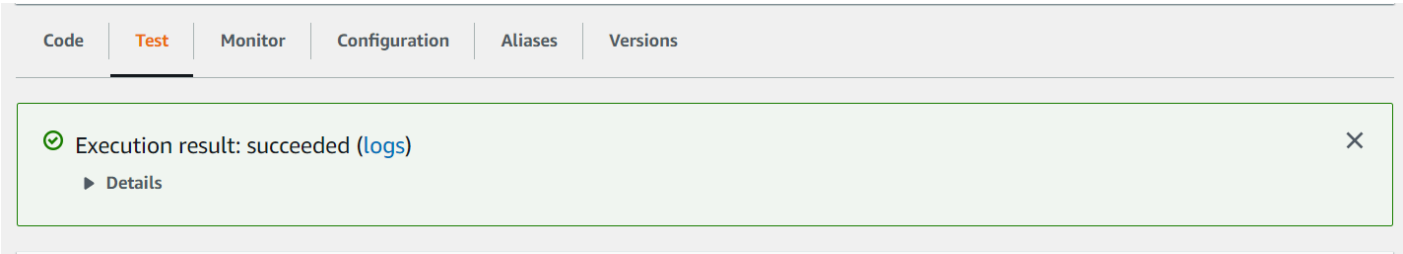
```
{
```

```
"flag": "true"
 }
```



> **Note**
>
> Die Übergabe von wahr kennzeichnet die digitalen Assets und die Übergabe von falsch löscht die Tags.

2. Wählen Sie die Schaltfläche Aufrufen. Nachdem die Lambda-Funktion aufgerufen wurde, wird eine Erfolgsmeldung angezeigt.



Herzlichen Glückwunsch, Sie haben eine AWS Lambda Funktion erstellt, die automatisch Tags auf digitale Assets anwendet, die sich in einem Amazon S3 S3-Bucket befinden. Wie zu Beginn dieses Tutorials erwähnt, sollten Sie darauf achten, alle Ressourcen zu beenden, die Sie während der Bearbeitung dieses Tutorials erstellt haben, um sicherzustellen, dass Ihnen nichts in Rechnung gestellt wird.

Weitere AWS Multiservice-Beispiele finden Sie im Documentation SDK Examples Repository.AWS GitHub

AWS Videoanalysator-Anwendungen erstellen

Mit dem AWS SDK for Java Version 2 können Sie eine Java-Webanwendung erstellen, die Videos zur Labelerkennung analysiert. Mit der in diesem AWS Tutorial erstellten Anwendung können Sie ein Video (MP4 Datei) in einen Amazon S3 S3-Bucket hochladen. Die Anwendung nutzt anschließend den Amazon-Rekognition-Dienst, um das Video zu analysieren. Die Ergebnisse werden verwendet, um ein Datenmodell zu füllen. Anschließend wird ein Bericht generiert und mithilfe des Amazon Simple Email Service per E-Mail an einen bestimmten Benutzer gesendet.

Die folgende Abbildung zeigt einen Bericht, der generiert wird, nachdem die Anwendung die Analyse des Videos abgeschlossen hat. Die Spalten in der Tabelle unten zeigen Altersgruppe, Bart, Brille und Augen offen sowie Konfidenzwerte für verschiedene Attributvorhersagen.

	Age Range	Beard	Eye glasses	Eyes open
2				
3	AgeRange(Low=38, High=56)	Beard(Value=false, Confidence=83.07253)	Eyeglasses(Value=true, Confidence=55.965977)	EyeOpen(Value=true, Confidence=94.691696)
4	AgeRange(Low=36, High=52)	Beard(Value=true, Confidence=50.721912)	Eyeglasses(Value=false, Confidence=63.886036)	EyeOpen(Value=true, Confidence=95.906364)
5	AgeRange(Low=51, High=69)	Beard(Value=true, Confidence=58.38352)	Eyeglasses(Value=false, Confidence=96.39576)	EyeOpen(Value=true, Confidence=53.580643)
6	AgeRange(Low=49, High=67)	Beard(Value=false, Confidence=81.41662)	Eyeglasses(Value=true, Confidence=65.28722)	EyeOpen(Value=true, Confidence=95.11523)
7	AgeRange(Low=51, High=69)	Beard(Value=true, Confidence=61.533833)	Eyeglasses(Value=false, Confidence=97.51163)	EyeOpen(Value=true, Confidence=82.21834)
8	AgeRange(Low=29, High=45)	Beard(Value=false, Confidence=74.22591)	Eyeglasses(Value=true, Confidence=64.906685)	EyeOpen(Value=true, Confidence=98.48175)
9	AgeRange(Low=51, High=69)	Beard(Value=true, Confidence=65.9394)	Eyeglasses(Value=false, Confidence=94.14824)	EyeOpen(Value=true, Confidence=94.857346)
10	AgeRange(Low=44, High=62)	Beard(Value=true, Confidence=78.648)	Eyeglasses(Value=true, Confidence=65.83134)	EyeOpen(Value=true, Confidence=98.538666)
11				

In diesem Tutorial erstellen Sie eine Spring Boot-Anwendung, die verschiedene AWS Dienste aufruft. Die Spring Boot APIs werden verwendet, um ein Modell, verschiedene Ansichten und einen Controller zu erstellen. Weitere Informationen finden Sie unter Spring Boot.

Dieser Dienst verwendet die folgenden AWS Dienste:

- Amazon Rekognition
- Amazon S3
- Amazon SES
- AWS Elastic Beanstalk

Die in diesem Tutorial enthaltenen AWS Dienste sind im AWS kostenlosen Kontingent enthalten. Wir empfehlen, dass Sie alle Ressourcen, die Sie im Tutorial erstellt haben, beenden, wenn Sie mit ihnen fertig sind, um zu vermeiden, dass Ihnen Gebühren berechnet werden.

Voraussetzungen

Bevor Sie beginnen, müssen Sie die Schritte unter Einrichten des AWS SDK for Java ausführen. Dann stellen Sie sicher, dass Sie Folgendes haben:

- Java 1.8 JDK.
- Maven 3.6 oder höher.
- Ein Amazon-S3-Bucket mit dem Namen video[somevalue]. Achten Sie darauf, diesen Bucket-Namen in Ihrem Amazon-S3-Java-Code zu verwenden. Weitere Informationen finden Sie unter Bucket erstellen.
- Eine IAM-Rolle. Sie benötigen dies für die VideoDetectFacesKlasse, die Sie erstellen werden. Weitere Informationen finden Sie unter Amazon Rekognition Video konfigurieren.
- Ein gültiges Amazon-SNS-Thema. Sie benötigen dies für die VideoDetectFacesKlasse, die Sie erstellen werden. Weitere Informationen finden Sie unter Amazon Rekognition Video konfigurieren.

Verfahren

Im Verlauf des Tutorials führen Sie folgende Aufgaben aus:

1. Erstellen eines Projekts
2. Ihrem Projekt die POM-Abhängigkeiten hinzufügen
3. Erstellen von Java-Klassen
4. Erstellen von HTML-Dateien
5. Erstellen von Skriptdateien
6. Verpacken des Projekts in eine JAR-Datei
7. Stellen Sie die Anwendung bereit für AWS Elastic Beanstalk

Folgen Sie den detaillierten Anweisungen im AWS GitHub Dokumentations-SDK-Beispiel-Repository, um mit dem Tutorial fortzufahren.

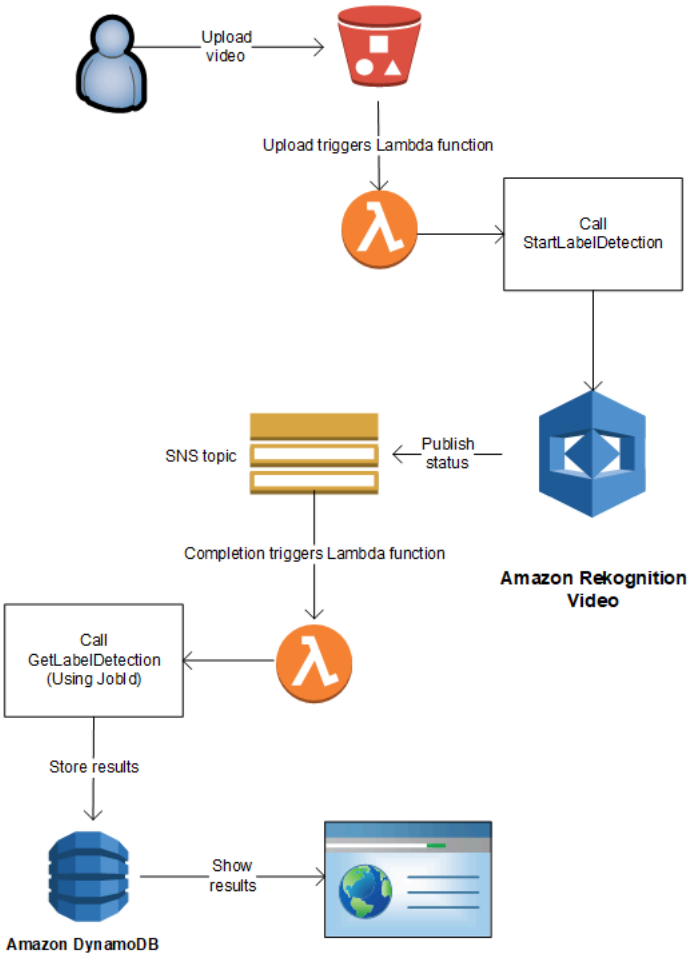
Erstellen einer Lambda-Funktion für Amazon Rekognition

In diesem Tutorial wird gezeigt, wie Sie die Ergebnisse einer Videoanalyse-Operation für eine Label-Erkennung mit einer Java-Lambda-Funktion abrufen.

> **Note**
>
> Dieses Tutorial verwendet das AWS SDK for Java 1.x. Ein Tutorial zur Verwendung von Rekognition und dem AWS SDK for Java Version 2 finden Sie im AWS Documentation SDK Examples GitHub Repository.

Sie können Lambda-Funktionen mit Amazon-Rekognition-Video-Operationen verwenden. Das folgende Diagramm zeigt z. B. eine Website, die eine Lambda-Funktion verwendet, um automatisch die Analyse eines Videos zu starten, wenn es in einen Amazon-S3-Bucket hochgeladen wird. Wenn die Lambda-Funktion ausgelöst wird, ruft sie auf, StartLabelDetectionum mit der Erkennung von Labels im hochgeladenen Video zu beginnen. Informationen zur Verwendung von Lambda für die Verarbeitung von Ereignisbenachrichtigungen aus einem Amazon-S3-Bucket finden Sie unter Verwenden von AWS Lambda mit Amazon-S3-Ereignissen.

Eine zweite Lambda-Funktion wird ausgelöst, wenn der Abschlussstatus für die Analyse an das registrierte Amazon-SNS-Thema gesendet wird. Die zweite Lambda-Funktion ruft GetLabelDetectionauf, um die Analyseergebnisse abzurufen. Die Ergebnisse werden in einer Datenbank gespeichert, um ihre Anzeige auf einer Webseite vorzubereiten. Diese zweite Lambda-Funktion bildet den Schwerpunkt dieses Tutorials.



In diesem Tutorial wird die Lambda-Funktion ausgelöst, wenn Amazon Rekognition Video den Abschlussstatus für die Videoanalyse an das registrierte Amazon-SNS-Thema sendet. Die Videoanalyseergebnisse werden danach durch Aufrufen von GetLabelDetection gesammelt. Zu Demonstrationszwecken schreibt dieses Tutorial die Ergebnisse der Etikettenerkennung in ein Protokoll. CloudWatch In der Lambda-Funktion in Ihrer Anwendung sollten Sie die Analyseergebnisse zur späteren Verwendung speichern. Dazu können Sie beispielsweise Amazon DynamoDB verwenden. Weitere Informationen finden Sie im Thema zum Arbeiten mit DynamoDB.

In den nachstehenden Verfahren wird Folgendes veranschaulicht:

- Erstellen eines Amazon-SNS-Themas und Einrichten von Berechtigungen
- Erstellen Sie die Lambda-Funktion mithilfe von AWS Management Console und abonnieren Sie sie für das Amazon SNS SNS-Thema.
- Konfigurieren der Lambda-Funktion mithilfe der AWS Management Console.
- Fügen Sie Beispielcode zu einem AWS Toolkit for Eclipse Projekt hinzu und laden Sie ihn in die Lambda-Funktion hoch.

- Testen der Lambda-Funktion mithilfe der AWS CLI.

> **Note**
>
> Verwenden Sie während des gesamten Tutorials dieselbe AWS Region.

Voraussetzungen

In diesem Tutorial wird vorausgesetzt, dass Sie mit AWS Toolkit for Eclipse vertraut sind. Weitere Informationen finden Sie unter AWS Toolkit for Eclipse.

Erstellen eines SNS-Themas

Der Abschlussstatus einer Amazon-Rekognition-Video-Videoanalyse-Operation wird an ein Amazon-SNS-Thema gesendet. In diesem Verfahren erstellen Sie das Amazon-SNS-Thema und die IAM-Servicerolle, durch die Amazon-Rekognition-Video Zugriff auf Ihre Amazon-SNS-Themen erhält. Weitere Informationen finden Sie unter Amazon-Rekognition-Video-Operationen aufrufen.

Erstellen eines Amazon SNS-Themas

1. Wenn Sie es noch nicht getan haben, erstellen Sie eine IAM-Servicerolle, um Amazon-Rekognition-Video Zugriff auf Ihre Amazon-SNS-Themen zu geben. Notieren Sie den Amazon-Ressourcennamen (ARN). Weitere Informationen finden Sie unter Den Zugriff auf mehrfache Amazon-SNS-Themen ermöglichen.
2. Erstellen Sie ein Amazon-SNS-Thema über die Amazon-SNS-Konsole, wobei Sie nur den Namen des Themas angeben müssen. Stellen Sie dem Themennamen ein. AmazonRekognition Notieren Sie den ARN des Themas.

So erstellen Sie die Lambda-Funktion:

Sie erstellen die Lambda-Funktion mithilfe der AWS Management Console. Sie verwenden anschließend ein AWS Toolkit for Eclipse -Projekt, um das Lambda-Funktionspaket auf AWS Lambda hochzuladen. Sie können die Lambda-Funktion auch mit dem AWS Toolkit for Eclipse erstellen. Weitere Informationen finden Sie unter Tutorial: Erstellen, Hochladen und Aufrufen einer AWS Lambda-Funktion.

So erstellen Sie die Lambda-Funktion:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die AWS Lambda-Konsole unter https://console.aws.amazon.com/lambda/.
2. Wählen Sie Funktion erstellen.
3. Wählen Sie Von Grund auf neu schreiben aus.
4. Geben Sie in Funktionsname einen Namen für Ihre Funktion ein.
5. Wählen Sie in Laufzeit die Option Java 8 aus.
6. Wählen Sie Ausführungsrolle auswählen oder erstellen aus.
7. Wählen Sie in Ausführungsrolle die Option Neue Rolle mit Lambda-Basisberechtigungen erstellen aus.
8. Notieren Sie den Namen der neuen Rolle, der unten im Abschnitt Basisinformationen angezeigt wird.
9. Wählen Sie Funktion erstellen.

Konfigurieren der Lambda-Funktion

Nachdem Sie die Lambda-Funktion erstellt haben, konfigurieren Sie ihre Auslösung durch das Amazon-SNS-Thema, das Sie im Abschnitt Erstellen eines SNS-Themas fertiggestellt haben. Sie passen außerdem die Speicheranforderungen und den Timeout-Zeitraum für die Lambda-Funktion an.

Konfigurieren der Lambda-Funktion

1. Geben Sie in Funktionscode folgenden Text für den `com.amazonaws.lambda.demo.JobCompletionHandler`Handler ein:
2. Wählen Sie in Basiseinstellungen die Option Bearbeiten aus. Anschließend wird das Dialogfeld Basiseinstellungen bearbeiten angezeigt.
   a. Wählen Sie 1024 für Speicher.
   b. Wählen Sie für Timeout 10 Sekunden.
   c. Wählen Sie Speichern.
3. Wählen Sie in Designer die Option + Auslöser hinzufügen aus. Anschließend wird das Dialogfeld „Auslöser hinzufügen“ angezeigt.
4. Wählen Sie in Auslöserkonfiguration die Option SNS aus.

Wählen Sie in SNS-Thema das Amazon-SNS-Thema aus, das Sie in Erstellen eines SNS-Themas erstellt haben.

5. Klicken Sie auf Auslöser aktivieren.
6. Wählen Sie Hinzufügen aus, um den Auslöser hinzuzufügen.
7. Wählen Sie Speichern, um die Lambda-Funktion zu speichern.

Konfigurieren der IAM-Lambda-Rolle

Um Amazon Rekognition Video Operations aufzurufen, fügen Sie die von AmazonRekognitionFullAccessAWS verwaltete Richtlinie zur IAM-Lambda-Rolle hinzu. Startvorgänge erfordern beispielsweise auch StartLabelDetectionPass-Rollen-Berechtigungen für die IAM-Servicerolle, die Amazon Rekognition Video für den Zugriff auf das Amazon SNS SNS-Thema verwendet.

Konfigurieren Sie die Rolle wie folgt:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter. https://console.aws.amazon.com/iam/
2. Wählen Sie im Navigationsbereich Rollen.
3. Wählen Sie in der Liste den Namen der Ausführungsrolle aus, die Sie in So erstellen Sie die Lambda-Funktion: erstellt haben.
4. Wählen Sie die Registerkarte Berechtigungen.
5. Wählen Sie Richtlinien anfügen.
6. Wählen Sie AmazonRekognitionFullAccessaus der Liste der Richtlinien aus.
7. Wählen Sie Richtlinie anfügen aus.
8. Wählen Sie die Ausführungsrolle erneut aus.
9. Wählen Sie Inline-Richtlinie hinzufügen.
10. Wählen Sie den Tab JSON.
11. Ersetzen Sie die vorhandene Richtlinie durch die unten stehende. Tauschen Sie `servicerole` durch die IAM-Servicerolle aus, die Sie im Abschnitt Erstellen eines SNS-Themas erstellt haben.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "mysid",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:servicerole"
        }
    ]
}
```

12. Wählen Sie Richtlinie prüfen.
13. Geben Sie in Name* einen Namen für die Richtlinie ein.
14. Wählen Sie Richtlinie erstellen aus.

Erstellen Sie das AWS Toolkit for Eclipse -Lambda-Projekt

Wenn die Lambda-Funktion ausgelöst wird, ruft der folgende Code den Abschlussstatus aus dem Amazon SNS SNS-Thema ab und ruft GetLabelDetectionauf, um die Analyseergebnisse abzurufen. Die Anzahl der erkannten Labels und eine Liste der erkannten Labels werden in ein CloudWatch Protokoll geschrieben. Die Lambda-Funktion speichert die Ergebnisse der Videoanalyse zur späteren Verwendung.

Um das AWS Toolkit for Eclipse Lambda-Projekt zu erstellen

1. Erstellen Sie ein AWS Toolkit for EclipseAWS Lambda-Projekt.
   - Geben Sie in Projektname: einen Namen Ihrer Wahl ein.
   - Geben JobCompletionHandlerSie als Klassenname: ein.
   - Wählen Sie unter Eingabetyp: die Option SNS-Ereignis aus.
   - Lassen Sie die anderen Felder unverändert.
2. Öffnen Sie im Eclipse-Projektexplorer die generierte Lambda-Handler-Methode (JobCompletionHandler.java) und ersetzen Sie den Inhalt durch Folgendes:

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

package com.amazonaws.lambda.demo;

import com.amazonaws.services.lambda.runtime.Context;
```

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



public class JobCompletionHandler implements RequestHandler<SNSEvent, String> {

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

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

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

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

            if (jsonResultTree.get("Status").asText().equals("SUCCEEDED")){
               GetResultsLabels(jsonResultTree.get("JobId").asText(), context);
            }
            else{
               String errorMessage = "Video analysis failed for job "
                     + jsonResultTree.get("JobId")
                     + "State " + jsonResultTree.get("Status");
```

```
               throw new Exception(errorMessage);
            }

         } else
            logger.log("Operation not StartLabelDetection");

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


      }

      return message;
   }

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

      LambdaLogger logger = context.getLogger();

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

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

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

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

.withSortBy(LabelDetectionSortBy.NAME).withMaxResults(maxResults).withNextToken(pagination

         labelDetectionResult = rek.getLabelDetection(labelDetectionRequest);

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

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

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

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

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

   }


}
```

3. Die Rekognition-Namespaces werden nicht aufgelöst. Korrigieren Sie dies wie folgt:

   - Platzieren Sie den Mauszeiger über dem unterstrichenen Teil der Zeile `import com.amazonaws.services.rekognition.AmazonRekognition;`.
   - Wählen Sie Projekteinrichtung korrigieren... aus.
   - Wählen Sie die neueste Version des Amazon-Rekognition-Archivs aus.
   - Klicken Sie auf OK, um dem Projekt das Archiv hinzuzufügen.
4. Speichern Sie die Datei.
5. Klicken Sie mit der rechten Maustaste in Ihr Eclipse-Code-Fenster und wählen AWS Lambda und dann Upload function to AWS Lambda aus.
6. Wählen Sie auf der Seite Ziel-Lambda-Funktion auswählen die zu verwendende AWS-Region aus.

7. Wählen Sie Vorhandene Lambda-Funktion auswählen aus und klicken Sie auf die Lambda-Funktion, die Sie im Abschnitt So erstellen Sie die Lambda-Funktion: fertiggestellt haben.
8. Wählen Sie Weiter. Das Dialogfeld Funktionskonfiguration wird angezeigt.
9. Wählen Sie in IAM-Rolle die IAM-Rolle aus, die Sie in So erstellen Sie die Lambda-Funktion: erstellt haben.
10. Klicken Sie auf Beenden. Die Lambda-Funktion wird in AWS hochgeladen.

Lambda-Funktion testen

Verwenden Sie den folgenden AWS CLI Befehl, um die Lambda-Funktion zu testen, indem Sie die Labelerkennungsanalyse eines Videos starten. Nach Abschluss der Analyse wird die Lambda-Funktion ausgelöst. Vergewissern Sie sich, dass die Analyse erfolgreich war, indem Sie die CloudWatch Log-Logs überprüfen.

Lambda-Funktion testen

1. Laden Sie eine Videodatei im MOV- oder MPEG-4-Format in Ihren S3-Bucket hoch. Laden Sie zu Testzwecken ein Video hoch, das nicht länger als 30 Sekunden ist.

   Weitere Anleitungen finden Sie unter Upload eines Objekts in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service.
2. Führen Sie den folgenden AWS CLI Befehl aus, um mit der Erkennung von Labels in einem Video zu beginnen.

   ```
   aws rekognition start-label-detection --video
    "S3Object={Bucket="bucketname",Name="videofile"}" \
   --notification-channel "SNSTopicArn=TopicARN,RoleArn=RoleARN" \
   --region Region
   ```

   Aktualisieren Sie die folgenden Werte:

   - Ändern Sie `bucketname` und `videofile` in den Amazon-S3-Bucket-Namen und Dateinamen des Videos, in dem Sie Label erkennen möchten.
   - Ändern Sie `TopicARN` in den ARN des Amazon-SNS-Themas, das Sie im Abschnitt Erstellen eines SNS-Themas fertiggestellt haben.

- Ändern Sie `RoleARN` in den ARN der IAM-Rolle, die Sie im Abschnitt Erstellen eines SNS-Themas angelegt haben.
- Wechseln Sie `Region` zu der AWS Region, die Sie verwenden.

3. Notieren Sie den `JobId`-Wert in der Antwort. Die Antwort sollte dem folgenden JSON-Beispiel ähnlich sein.

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

4. Öffnen Sie die https://console.aws.amazon.com/cloudwatch/-Konsole.
5. Wenn die Analyse abgeschlossen ist, wird ein Protokolleintrag für die Lambda-Funktion in der Protokollgruppe angezeigt.
6. Wählen Sie die Lambda-Funktion aus, um die Protokollstreams anzuzeigen.
7. Wählen Sie den neuesten Protokollstream aus, um die Protokolleinträge der Lambda-Funktion anzusehen. Wenn der Vorgang erfolgreich war, ähnelt er der folgenden Ausgabe, in der die Details des Videoerkennungsvorgangs angezeigt werden, einschließlich der Auftrags-ID, des Vorgangstyps "StartLabelDetection„und einer Liste der erkannten Etikettenkategorien wie Flasche, Kleidung, Menschenmenge und Lebensmittel:

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:

Der Wert für Aufgaben-ID sollte mit dem Wert für `JobId` übereinstimmen, den Sie in Schritt 3 notiert haben.

Amazon Rekognition zur Identitätsprüfung verwenden

Amazon Rekognition bietet Benutzern verschiedene Funktionen, die die einfache Erstellung von Systemen zur Identitätsverifizierung ermöglichen. Amazon Rekognition ermöglicht es dem

Benutzer, Gesichter in einem Bild zu erkennen und dann alle erkannten Gesichter mit anderen Gesichtern zu vergleichen, indem Gesichtsdaten verglichen werden. Diese Gesichtsdaten werden in serverseitigen Containern gespeichert, die als Sammlungen bezeichnet werden. Mithilfe der Funktionen Gesichtserkennung, Gesichtsvergleich und Sammlungsverwaltung von Amazon Rekognition können Sie eine Anwendung mit einer Lösung zur Identitätsprüfung erstellen.

In diesem Tutorial werden zwei gängige Workflows für die Erstellung von Anwendungen demonstriert, für die eine Identitätsprüfung erforderlich ist.

Der erste Workflow beinhaltet die Registrierung eines neuen Benutzers in einer Sammlung. Der zweite Workflow beinhaltet das Durchsuchen einer vorhandenen Sammlung, um einen wiederkehrenden Benutzer anzumelden.

Für dieses Tutorial verwenden Sie das AWS -SDK für Python. Weitere Python-Tutorials finden Sie auch im AWS Documentation SDK Examples GitHub Repo.

Themen

- Voraussetzungen
- Erstellen einer Sammlung
- Registrierung neuer Benutzer
- Anmeldung bestehender Benutzer

Voraussetzungen

Bevor Sie mit diesem Tutorial beginnen, müssen Sie Python installieren und die zur Einrichtung des AWS Python-SDK erforderlichen Schritte ausführen. Stellen Sie darüber hinaus sicher, dass Sie über Folgendes verfügen:

- Sie haben ein AWS -Konto und eine IAM-Rolle erstellt
- Sie haben das Python-SDK (Boto3) installiert
- Haben Sie Ihre AWS Zugangsdaten richtig konfiguriert
- Sie haben einen Amazon-Simple-Storage-Service-Bucket erstellt und ein Bild hochgeladen, das Sie als ID für die Identitätsprüfung verwenden möchten.
- Sie haben ein zweites Bild ausgewählt, das als Zielbild für die Identitätsprüfung dienen soll.

Erstellen einer Sammlung

Bevor Sie einen neuen Benutzer in einer Sammlung registrieren oder in einer Sammlung nach einem Benutzer suchen können, benötigen Sie eine Sammlung, mit der Sie arbeiten können. Eine Amazon-Rekognition-Sammlung ist ein serverseitiger Container, in dem Informationen über erkannte Gesichter gespeichert werden.

Erstellen der Sammlung

Sie beginnen damit, eine Funktion zu schreiben, die eine Sammlung zur Verwendung durch Ihre Anwendung erstellt. Amazon Rekognition speichert Informationen über erkannte Gesichter in serverseitigen Containern namens Sammlungen. Sie können in einer Sammlung gespeicherte Gesichtsinformationen nach bekannten Gesichtern durchsuchen. Um Gesichtsinformationen zu speichern, müssen Sie zunächst mithilfe dieser `CreateCollection`-Operation eine Sammlung erstellen.

1. Wählen Sie einen Namen für die Sammlung aus, die Sie erstellen möchten. Ersetzen Sie im folgenden Code den Wert von `collection_id` durch den Namen der Sammlung, die Sie erstellen möchten, und ersetzen Sie den Wert von `region` durch den Namen der Region, die in Ihren Benutzeranmeldeinformationen definiert ist. Sie können das `Tags`-Argument verwenden, um beliebige Tags auf die Sammlung anzuwenden, obwohl dies nicht erforderlich ist. Die `CreateCollection`-Operation gibt Informationen über die von Ihnen erstellte Sammlung zurück, einschließlich des ARN der Sammlung. Notieren Sie sich den ARN, den Sie als Ergebnis der Ausführung des Codes erhalten.

```
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. Speichern Sie den Code und führen Sie ihn aus. Schreiben Sie den Sammlungs-ARN auf.

   Nachdem die Rekognition-Sammlung erstellt wurde, können Sie Gesichtsinformationen und -identifikatoren in dieser Sammlung speichern. Zur Überprüfung können Sie auch Gesichter mit den gespeicherten Informationen vergleichen.

Registrierung neuer Benutzer

Sie sollten in der Lage sein, neue Benutzer zu registrieren und ihre Informationen zu einer Sammlung hinzuzufügen. Die Registrierung eines neuen Benutzers umfasst in der Regel die folgenden Schritte:

Aufrufen der **`DetectFaces`**-Operation

Schreiben Sie den Code, um die Qualität des Gesichtsbildes während der `DetectFaces`-Operation zu überprüfen. Mithilfe der `DetectFaces`-Operation können Sie feststellen, ob ein mit der Kamera aufgenommenes Bild für die Verarbeitung durch die `SearchFacesByImage`-Operation geeignet ist. Das Bild sollte nur ein Gesicht enthalten. Sie stellen der `DetectFaces` Operation eine lokale Eingabebilddatei zur Verfügung und erhalten Details zu den im Bild erkannten Gesichtern. Der folgende Beispielcode stellt das Eingabebild an `DetectFaces` bereit und überprüft dann, ob im Bild nur ein Gesicht erkannt wurde.

1. Ersetzen Sie im folgenden Codebeispiel `photo` durch den Namen des Zielbilds, in dem Sie Gesichter erkennen möchten. Außerdem müssen Sie den Wert von `region` durch den Namen der Region ersetzen, die Ihrem Konto zugeordnet ist.

```
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. Speichern Sie den nachfolgenden Code und führen Sie ihn aus.

Aufrufen der **`CompareFaces`**-Operation

Ihre Anwendung muss in der Lage sein, neue Benutzer in einer Sammlung zu registrieren und die Identität wiederkehrender Benutzer zu bestätigen. Sie erstellen zunächst die Funktionen, mit denen ein neuer Benutzer registriert wird. Zunächst verwenden Sie den `CompareFaces` Vorgang, um ein lokales input/target image of the user and a ID/stored Bild zu vergleichen. Wenn es eine Übereinstimmung zwischen dem auf beiden Bildern erkannten Gesicht gibt, können Sie die Sammlung durchsuchen, um zu sehen, ob der Benutzer darin registriert ist.

Schreiben Sie zunächst eine Funktion, die ein Eingabebild mit dem ID-Bild vergleicht, das Sie in Ihrem Amazon-S3-Bucket gespeichert haben. Im folgenden Codebeispiel müssen Sie das Eingabebild selbst angeben, das nach der Verwendung eines Echtheitsdetektors aufgenommen

werden sollte. Sie müssen auch den Namen eines Bildes übergeben, das in Ihrem Amazon-S3-Bucket gespeichert ist.

1. Ersetzen Sie den Wert von `bucket` durch den Namen des Amazon-S3-Buckets, der Ihre Quelldatei enthält. Sie müssen auch den Wert von `source_file` durch den Namen des Quellbilds ersetzen, das Sie verwenden. Ersetzen Sie den Wert von `target_file` durch den Namen der Zieldatei, die Sie angegeben haben. Ersetzen Sie den Wert von `region` durch den Namen der `region`, die in Ihren Benutzeranmeldeinformationen definiert ist.

   Sie werden auch eine Mindestwahrscheinlichkeit für die in der Antwort zurückgegebene Übereinstimmung angeben wollen, indem Sie das `similarityThreshold`-Argument verwenden. Erkannte Gesichter werden im `FaceMatches`-Array nur zurückgegeben, wenn die Zuverlässigkeit über diesem Schwellenwert liegt. Ihr ausgewählter `similarityThreshold` sollte die Art Ihres spezifischen Anwendungsfalls widerspiegeln. Für jeden Anwendungsfall, der kritische Sicherheitsanwendungen betrifft, sollte 99 als ausgewählter Schwellenwert verwendet werden.

```
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. Speichern Sie den nachfolgenden Code und führen Sie ihn aus.

   Es wird ein Antwortobjekt zurückgegeben, das Informationen über das übereinstimmende Gesicht und das Zuverlässigkeitsniveau enthält.

Aufrufen der **`SearchFacesByImage`**-Operation

Wenn das Zuverlässigkeitsniveau der `CompareFaces`-Operation über dem von Ihnen ausgewählten `SimilarityThreshold` liegt, sollten Sie Ihre Sammlung nach einem Gesicht durchsuchen, das dem Eingabebild entsprechen könnte. Wenn in Ihrer Sammlung eine Übereinstimmung gefunden wird, bedeutet das, dass der Benutzer wahrscheinlich bereits in der Sammlung registriert ist und es nicht notwendig ist, einen neuen Benutzer in Ihrer Sammlung zu registrieren. Wenn es keine Übereinstimmung gibt, können Sie den neuen Benutzer in Ihrer Sammlung registrieren.

1. Schreiben Sie zunächst den Code, der die `SearchFacesByImage`-Operation aufruft. Die Operation verwendet eine lokale Bilddatei als Argument und durchsucht dann Ihre `Collection` nach einem Gesicht, das den größten erkannten Gesichtern im bereitgestellten Bild entspricht.

   Ändern Sie im folgenden Codebeispiel den Wert von `collectionId` in die Sammlung, nach der Sie suchen möchten. Ersetzen Sie den Wert von `region` durch den Namen der Region, die Ihrem Konto zugeordnet ist. Sie müssen auch den Wert von `photo` durch den Namen Ihrer Eingabedatei ersetzen. Sie sollten auch einen Ähnlichkeitsschwellenwert angeben, indem Sie den Wert von `threshold` durch ein ausgewähltes Perzentil ersetzen.

```
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. Speichern Sie den nachfolgenden Code und führen Sie ihn aus. Wenn eine Übereinstimmung gefunden wurde, bedeutet dies, dass die Person, die auf dem Bild erkannt wird, bereits Teil der Sammlung ist und Sie nicht mit den nächsten Schritten fortfahren müssen. In diesem Fall können Sie dem Benutzer einfach Zugriff auf die Anwendung gewähren.

Aufrufen der **`IndexFaces`**-Operation

Unter der Annahme, dass in der von Ihnen gesuchten Sammlung keine Übereinstimmung gefunden wurde, sollten Sie das Gesicht des Benutzers zu Ihrer Sammlung hinzufügen. Sie tun dies, indem Sie die `IndexFaces`-Operation aufrufen. Wenn Sie anrufen IndexFaces, extrahiert Amazon Rekognition die Gesichtszüge eines Gesichts, das in Ihrem Eingabebild identifiziert wurde, und speichert die Daten in der angegebenen Sammlung.

1. Schreiben Sie zunächst den Code für den Aufruf von `IndexFaces`. Ersetzen Sie den Wert von `image` durch den Namen der lokalen Datei, die Sie als Eingabebild für den IndexFaces Vorgang verwenden möchten. Sie müssen auch den Wert von `photo_name` durch den gewünschten Namen für Ihr Eingabebild ersetzen. Achten Sie darauf, dass Sie den Wert von `collection_id` durch die ID der Sammlung ersetzen, die Sie zuvor erstellt haben. Ersetzen Sie dann den Wert von `region` durch den Namen der Region, die Ihrem Konto zugeordnet ist. Sie sollten auch einen Wert für den `MaxFaces`-Eingabeparameter angeben, der die maximale Anzahl von

Gesichtern in einem Bild definiert, die indexiert werden sollen. Der Standardwert für diesen Parameter ist 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. Speichern Sie den nachfolgenden Code und führen Sie ihn aus. Stellen Sie fest, ob Sie die durch die `IndexFaces`-Operation zurückgegebenen Daten speichern möchten, z. B. die Gesichts-ID, die der Person im Bild zugewiesen wurde. Im nächsten Abschnitt wird untersucht, wie diese Daten gespeichert werden. Kopieren Sie die zurückgegebenen `FaceId`-, `ImageId`- und `Confidence`-Werte, bevor Sie fortfahren.

Speichern Sie Bild- und Gesichts–ID-Daten in Amazon S3 und Amazon DynamoDB

Sobald die Gesichts-ID für das Eingabebild abgerufen wurde, können die Bilddaten in Amazon S3 gespeichert werden, während die Gesichtsdaten und die Bild-URL in eine Datenbank wie DynamoDB eingegeben werden können.

1. Schreiben Sie den Code, um das Eingabebild in Ihre Amazon-S3-Datenbank hochzuladen. Ersetzen Sie im folgenden Codebeispiel den Wert von `bucket` durch den Namen des Buckets, in den Sie die Datei hochladen möchten, und ersetzen Sie dann den Wert von `file_name` durch den Namen der lokalen Datei, die Sie in Ihrem Amazon-S3-Bucket speichern möchten. Geben Sie einen Schlüsselnamen an, der die Datei im Amazon-S3-Bucket identifiziert, indem Sie den Wert von `key_name` durch einen Namen ersetzen, den Sie der Bilddatei geben möchten. Die Datei, die Sie hochladen möchten, ist dieselbe, die in früheren Codebeispielen definiert wurde. Dies ist die Eingabedatei, für die Sie verwendet haben IndexFaces. Ersetzen Sie zum Schluss den Wert von `region` durch den Namen der Region, die Ihrem Konto zugeordnet ist.

```
import boto3
import logging
from botocore.exceptions import ClientError

# store local file in S3 bucket
bucket = "amzn-s3-demo-bucket"
file_name = "file-name"
key_name = "key-name"
region = "region-name"
s3 = boto3.client('s3', region_name=region)
# Upload the file
try:
    response = s3.upload_file(file_name, bucket, key_name)
    print("File upload successful!")
except ClientError as e:
```

```
    logging.error(e)
```

2. Speichern Sie das nachfolgende Codebeispiel und führen Sie es aus, um Ihr Eingabebild auf Amazon S3 hochzuladen.
3. Sie sollten die zurückgegebene Gesichts-ID auch in einer Datenbank speichern. Dies kann erreicht werden, indem eine DynamoDB-Datenbanktabelle erstellt und dann die Gesichts-ID in diese Tabelle hochgeladen wird. Im folgenden Codebeispiel wird eine DynamoDB-Tabelle erstellt. Beachten Sie, dass Sie den Code, der diese Tabelle erstellt, nur einmal ausführen müssen. Ersetzen Sie im folgenden Code den Wert von `region` durch den Wert der Region, die Ihrem Konto zugeordnet ist. Außerdem müssen Sie den Wert von `database_name` durch den Namen ersetzen, den Sie der DynamoDB-Tabelle geben möchten.

```
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. Speichern Sie den nachfolgenden Code und führen Sie ihn aus, um Ihre Tabelle zu erstellen.

5. Nachdem Sie die Tabelle erstellt haben, können Sie die FaceId zurückgegebene Datei hochladen. Dazu stellen Sie mit der Tabellenfunktion eine Verbindung zur Tabelle her und verwenden dann die `put_item`-Funktion, um die Daten hochzuladen.

   Ersetzen Sie im folgenden Codebeispiel den Wert von `bucket` durch den Namen des Buckets, der das Eingabebild enthält, das Sie auf Amazon S3 hochgeladen haben. Außerdem müssen Sie den Wert von `file_name` durch den Namen der Eingabedatei ersetzen, die Sie in Ihren Amazon-S3-Bucket hochgeladen haben, und den Wert von `key_name` durch den Schlüssel, den Sie zuvor zur Identifizierung der Eingabedatei verwendet haben. Ersetzen Sie zum Schluss den Wert von `region` durch den Namen der Region, die Ihrem Konto zugeordnet ist. Diese Werte sollten mit den in Schritt 1 angegebenen Werten übereinstimmen.

   `AddDBEntry`In werden die Werte FaceId ImageId, und Confidence gespeichert, die einem Gesicht in einer Sammlung zugewiesen wurden. Geben Sie für die unten stehende Funktion die Werte ein, die Sie in Schritt 2 des nächsten `IndexFaces`-Abschnitts gespeichert haben.

```
import boto3
from pprint import pprint
from decimal import Decimal
import json

# The local file that was stored in S3 bucket
bucket = "amzn-s3-demo-bucket"
file_name = "file-name"
key_name = "key-name"
region = "region-name"
# Get URL of file
file_url = "https://s3.amazonaws.com/{}/{}".format(bucket, key_name)
print(file_url)

# upload face-id, face info, and image url
def AddDBEntry(file_name, file_url, face_id, image_id, confidence):
    dynamodb = boto3.resource('dynamodb', region_name=region)
    table = dynamodb.Table('FacesDB-4')
    response = table.put_item(
       Item={
            'ExternalImageID': file_name,
            'ImageURL': file_url,
            'FaceID': face_id,
            'ImageID': image_id,
            'Confidence': json.loads(json.dumps(confidence), parse_float=Decimal)
```

```
        }
    )
    return response

# Mock values for face ID, image ID, and confidence - replace them with actual
 values from your collection results
dynamodb_resp = AddDBEntry(file_name, file_url, "FACE-ID-HERE",
    "IMAGE-ID-HERE", confidence-here)
print("Database entry successful.")
pprint(dynamodb_resp, sort_dicts=False)
```

6. Speichern Sie das nachfolgende Codebeispiel und führen Sie es aus, um die zurückgegebenen Gesichts-ID-Daten in einer Tabelle zu speichern.

Anmeldung bestehender Benutzer

Nachdem ein Benutzer in einer Sammlung registriert wurde, kann er bei seiner Rückkehr mithilfe der `SearchFacesByImage`-Operation authentifiziert werden. Sie müssen ein Eingabebild abrufen und dann die Qualität des Eingabebilds mithilfe von `DetectFaces` überprüfen. Dadurch wird festgestellt, ob vor dem Ausführen der `SearchFacesbyImage` Operation ein geeignetes Bild verwendet wurde.

Rufen Sie die DetectFaces Operation auf

1. Mit dieser `DetectFaces`-Operation überprüfen Sie die Qualität des Gesichtsbildes und stellen fest, ob ein mit der Kamera aufgenommenes Bild für die Verarbeitung durch die `SearchFacesByImage`-Operation geeignet ist. Das Eingabebild sollte nur ein Gesicht enthalten. Im folgenden Codebeispiel wird ein Eingabebild verwendet und für die `DetectFaces`-Operation bereitgestellt.

   Ersetzen Sie im folgenden Codebeispiel den Wert von `photo` durch den Namen des lokalen Zielbilds und ersetzen Sie den Wert von `region` durch den Namen der Region, die Ihrem Konto zugeordnet ist.

   ```
   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. Speichern Sie den Code und führen Sie ihn aus.

Ruf die SearchFacesByImage Operation an

1. Schreiben Sie den Code, um das erkannte Gesicht mit den Gesichtern in der Sammlung mit dem `SearchFacesByImage` zu vergleichen. Sie verwenden den Code, der im Abschnitt „Registrierung neuer Benutzer“ angezeigt wird, und stellen das Eingabebild für die `SearchFacesByImage`-Operation bereit.

   Ändern Sie im folgenden Codebeispiel den Wert von `collectionId` in die Sammlung, nach der Sie suchen möchten. Außerdem ändern Sie den Wert von `bucket` in den Namen eines Amazon-

S3-Buckets und den Wert von `fileName` in eine Bilddatei in diesem Bucket. Ersetzen Sie den Wert von `region` durch den Namen der Region, die Ihrem Konto zugeordnet ist. Sie sollten auch einen Ähnlichkeitsschwellenwert angeben, indem Sie den Wert von `threshold` durch ein ausgewähltes Perzentil ersetzen.

```
import boto3

bucket = 'amzn-s3-demo-bucket'
collectionId = 'collection-id-name'
region = "region-name"
fileName = 'file-name'
threshold = 70
maxFaces = 1
client = boto3.client('rekognition', region_name=region)

# input image should be local file here, not s3 file
with open(fileName, 'rb') as image:
    response = client.search_faces_by_image(CollectionId=collectionId,
    Image={'Bytes': image.read()},
    FaceMatchThreshold=threshold, MaxFaces=maxFaces)
```

2. Speichern Sie den Code und führen Sie ihn aus.

Suchen Sie nach der zurückgegebenen Gesichts-ID und dem Zuverlässigkeitsniveau

Sie können nun nach Informationen zu den übereinstimmenden Objekten suchen FaceId, FaceId indem Sie Antwortelemente wie die Attribute Ähnlichkeit und Konfidenz ausdrucken.

```
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']))
```

Verwendung von Amazon Rekognition (REK) zur Erkennung von Etiketten für Marketinganwendungen

Dieses Tutorial führt Sie durch die Erstellung einer Python-Beispielanwendung, mit der Sie E-Mails an Personen senden können, basierend auf Bildern, die auf eine Website hochgeladen wurden. Die Beispielanwendung wurde entwickelt, um Benutzer in eine Social-Media-Marketingkampagne einzubeziehen, indem ihnen personalisierte E-Mails mit Reiseangeboten gesendet werden, wenn Sehenswürdigkeiten auf ihren Fotos erkannt werden.

Die Lösung integriert verschiedene AWS Dienste, darunter Amazon Rekognition, Amazon S3, DynamoDB und Amazon SES CloudWatch. Die Anwendung verwendet Amazon Rekognition, um Labels in Bildern zu erkennen, die von Konten hochgeladen wurden, die in einer DynamoDB-Datenbank erkannt wurden. Anschließend erhält der Kontoinhaber eine Marketing-E-Mail, die auf den im Bild erkannten Labels basiert. Die vollständige Lösungsarchitektur sieht wie folgt aus:

- Speichern Sie Benutzerdaten in der DynamoDB-Datenbank.
- Der Benutzer lädt Bilddaten und Metadaten (Benutzerkontonummer) auf Amazon S3 hoch.
- Eine Lambda-Funktion wird aufgerufen, DetectLabelsum die Labels im hochgeladenen Bild zu identifizieren und aufzuzeichnen, die Benutzer-E-Mail-Adresse in DynamoDB nachzuschlagen und eine Marketing-E-Mail an den Benutzer zu senden, der das Bild hochgeladen hat.
- Die Ergebnisse der Lambda-Funktion werden CloudWatch zur späteren Überprüfung protokolliert.

Hier ist eine Übersicht über alle Schritte des Tutorials:

1. Erstellen Sie eine DynamoDB-Tabelle und füllen Sie sie auf.
2. Schreiben Sie eine Lambda-Funktion mit Logging und Notification.
3. Richten Sie die Lambda-Funktion und die Berechtigungen ein.
4. Laden Sie Bilder und Metadaten mit Amazon S3 hoch.
5. CloudWatch Umfrageprotokolle.

Themen

- Voraussetzungen
- DynamoDB-Tabelle erstellen und auffüllen
- Eine Lambda-Funktion mit Protokollierung und Benachrichtigung erstellen

- Berechtigungen und die Lambda-Funktion einrichten
- Bilder mit Metadaten auf Amazon S3 hochladen
- CloudWatch Abfrageprotokolle vom Client

Voraussetzungen

Bevor Sie mit diesem Tutorial beginnen, benötigen Sie:

- Ein AWS Konto und die entsprechenden IAM-Berechtigungen.
- Python und Boto3 sind in Ihrer Entwicklungsumgebung installiert.
- Grundlegendes Verständnis von Lambda, Amazon S3, DynamoDB, Amazon SES und. CloudWatch
- Grundkenntnisse in Lambda und Amazon SES

DynamoDB-Tabelle erstellen und auffüllen

Zu Beginn dieses Tutorials erstellen wir eine DynamoDB-Tabelle zum Speichern von Kundendaten wie E-Mail-Adresse, Alter, Telefonnummer und Mitgliedschaftsstatus, wobei eine "AccountNumber" -Variable als Primärschlüssel verwendet wird. Wir werden auch einige Beispieldaten in die Tabelle einfügen.

DynamoDB-Tabelle erstellen

Zuerst erstellen wir die DynamoDB-Tabelle. So richten Sie es mit Python und Boto3 ein. Wir erstellen eine aufgerufene Funktion `create_user_table` und stellen eine Verbindung zu der darin DynamoDB DynamoDB-Ressource her. Ersetzen Sie im folgenden Codebeispiel den Wert von „region_name“ durch den Namen der Region, in der Ihr Konto tätig ist, und führen Sie dann die Codezelle aus, um Ihre Tabelle zu erstellen.

```
import boto3
def create_user_table(dynamodb=None):
    if not dynamodb:
        dynamodb = boto3.resource('dynamodb', region_name='us-east-1')

    table = dynamodb.create_table(
        TableName='CustomerDataTable',
        KeySchema=[
            {
```

```
                'AttributeName': 'AccountNumber',
                'KeyType': 'HASH'  # Partition key
            },
        ],
        AttributeDefinitions=[
            {
                'AttributeName': 'AccountNumber',
                'AttributeType': 'S'
            },
        ],
        ProvisionedThroughput={
            'ReadCapacityUnits': 10,
            'WriteCapacityUnits': 10
        }
    )

    # Wait until the table exists.
    table.wait_until_exists()
    print("Table status:", table.table_status)

# Create the DynamoDB table.
create_user_table()
```

Durch die Ausführung dieses Skripts wurde eine DynamoDB-Tabelle `CustomerDataTable` mit dem Namen mit `AccountNumber` als Primärschlüssel eingerichtet.

Beispieldaten werden eingefügt

Jetzt wollen wir einige Beispieldaten in die Tabelle einfügen. Diese Beispieldaten helfen uns, die gesamte Funktionalität der Anwendung zu testen.

Wir werden eine zweite Funktion erstellen, die Beispieldaten zu den zuvor `CustomerDataTable` erstellten Funktionen hinzufügt. Mit dem folgenden Code werden drei Beispieleinträge erstellt, und jeder Eintrag enthält eine Kontonummer, eine E-Mail-Adresse, ein Alter, eine Telefonnummer und einen Mitgliedsstatus. Ersetzen Sie im folgenden Codebeispiel den Wert von `region_name` durch den Namen der Region, in der Ihr Konto tätig ist, und führen Sie dann die Codezelle aus, um Ihre Tabelle zu erstellen. Wenn Sie den Teil der App für die E-Mail-Zustellung testen möchten, ersetzen Sie den Wert von `EmailAddress` im ersten Kundeneintrag unten durch eine E-Mail-Adresse, unter der Sie E-Mails empfangen können. Speichern Sie den Code und führen Sie ihn aus.

```
import boto3

def insert_sample_data(dynamodb=None):
    if not dynamodb:
        dynamodb = boto3.resource('dynamodb', region_name='us-east-1')

    table = dynamodb.Table('CustomerDataTable')

    # Sample data
    customers = [
        {
            'AccountNumber': 'ACC1000',
            'EmailAddress': 'email-for-delivery-here',
            'Age': 30,
            'PhoneNumber': '123-456-7890',
            'MembershipStatus': 'Active'
        },
        {
            'AccountNumber': 'ACC1001',
            'EmailAddress': 'jane.doe@example.com',
            'Age': 25,
            'PhoneNumber': '098-765-4321',
            'MembershipStatus': 'Inactive'
        },
        {
            'AccountNumber': 'ACC1002',
            'EmailAddress': 'pat.candella@example.com',
            'Age': 35,
            'PhoneNumber': '555-555-5555',
            'MembershipStatus': 'Active'
        }
    ]
        # Inserting data
    for customer in customers:
            print(f"Adding customer: {customer['AccountNumber']}")
            table.put_item(Item=customer)

# Insert sample data into DynamoDB
insert_sample_data()
```

Nachdem die DynamoDB-Tabelle eingerichtet und gefüllt ist, können wir nun damit fortfahren, diesen Datenabruf in die Lambda-Funktion zu integrieren. Auf diese Weise kann unsere Anwendung

Benutzerdaten auf der Grundlage der Kontonummer abrufen, die im nächsten Bildverarbeitungs-Workflow identifiziert wird.

Eine Lambda-Funktion mit Protokollierung und Benachrichtigung erstellen

Jetzt können wir eine Lambda-Funktion erstellen. Wir möchten sicherstellen, dass die Lambda-Funktion, die beim Bild-Upload ausgelöst wird, sowohl die Bilddaten als auch die Konto-Metadaten lesen und damit eine Suche in der DynamoDB-Tabelle nach den zugehörigen Benutzerdaten durchführen kann. Das bedeutet, dass wir nicht nur die Funktion von Amazon Rekognition aufrufen müssen, sondern auch eine DetectLabels Funktion benötigen, die das aufnimmt AccountNumber und verwendet, um die zugehörige E-Mail-Adresse von DynamoDB abzurufen. Zusätzlich benötigen wir eine Funktion, die mit Amazon SES eine E-Mail an die E-Mail-Adresse sendet. Schließlich verwenden wir einen Logger, um Informationen über den Prozess zu protokollieren. Diese Daten werden von angezeigt CloudWatch.

Erstellen der Lambda-Funktion

Im Folgenden finden Sie einen Überblick darüber, wie eine Lambda-Funktion mit diesen Anforderungen umgehen könnte. Stellen Sie im folgenden Codebeispiel sicher, dass die richtige DynamoDB-Tabelle angegeben ist, wenn Sie etwas anderes CustomerDataTable als den Namen Ihrer Tabelle verwendet haben. Darüber hinaus sollten Sie in der Funktion „send_marketing_email“ den Wert des Arguments „Source“ durch eine E-Mail-Adresse ersetzen, auf die Sie Zugriff haben und die als sendende E-Mail fungiert.

```
import json
import boto3
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    s3_bucket = event['Records'][0]['s3']['bucket']['name']
    s3_object_key = event['Records'][0]['s3']['object']['key']
    print(s3_bucket)
    print(s3_object_key)

    s3 = boto3.client('s3')
    try:
        s3_response = s3.head_object(Bucket=s3_bucket, Key=s3_object_key)
        account_number = s3_response['Metadata']['account_number']
```

```
    except Exception as e:
        logger.error(f"Failed to retrieve object or metadata: {str(e)}")
        raise e  # Optionally re-raise to handle the error upstream or signal a failure

    rekognition = boto3.client('rekognition')
    try:
        labels_response = rekognition.detect_labels(Image={'S3Object': {'Bucket':
 s3_bucket, 'Name': s3_object_key}})
        #logger.info(f"Detected labels: {json.dumps(labels_response['Labels'],
 indent=4)}")
    except Exception as e:
        #logger.info(f"Detected label: {label['Name']}")
        raise e

    #logger.info(f"Detected labels: {json.dumps(labels_response['Labels'], indent=4)}")

    landmark_detected = any(label['Name'] == 'Landmark' and label['Confidence'] > 20
 for label in labels_response['Labels'])
    if landmark_detected:
        result = notify_user_based_on_landmark(account_number)
        print(result)
        #logger.info(f"Detected label: {label['Name']}")
        #logger.info(f"Notification sent: {result}")
    return {
        'statusCode': 200,
        'body': json.dumps('Process completed successfully!')
    }

def notify_user_based_on_landmark(account_number):
     # Retrieve user data from DynamoDB
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('CustomerDataTable')
    user_info = table.get_item(Key={'AccountNumber': account_number})

    # Send email if user is found
    if 'Item' in user_info:
        send_marketing_email(user_info['Item']['EmailAddress'])
    return "User notified with relevant travel information."

def send_marketing_email(email):
    ses = boto3.client('ses')
    response = ses.send_email(
        Source='your-email@example.com',
        Destination={'ToAddresses': [email]},
```

```
        Message={
            'Subject': {'Data': 'Explore New Destinations!'},
            'Body': {
                'Text': {'Data': 'Check out our exclusive travel packages inspired by
 the landmark in your uploaded image!'}
            }
        }
    )
    return f"Email sent to {email} with status {response['ResponseMetadata']
['HTTPStatusCode']}"
    print("succeess")
```

Nachdem wir die Lambda-Funktion geschrieben haben, müssen wir die Berechtigungen für Lambda konfigurieren und eine Instanz unserer Lambda-Funktion in der AWS Management Console erstellen.

Berechtigungen und die Lambda-Funktion einrichten

IAM-Rolle erstellen oder aktualisieren

Bevor wir eine Lambda-Funktion verwenden können, um von Benutzern hochgeladene Bilder zu verarbeiten, müssen wir die entsprechenden Berechtigungen verwalten. Die Lambda-Funktion benötigt eine auf sie zugeschnittene IAM-Rolle mit Richtlinien, die es ihr ermöglichen, mit Amazon S3, DynamoDB, Amazon SES, Amazon Rekognition und zu interagieren. CloudWatch

So konfigurieren Sie Ihre IAM-Rolle für die Lambda-Funktion:

1. Öffnen Sie die AWS Management Console.
2. Gehen Sie zu IAM > Rollen > Rolle erstellen.

   Wählen Sie „Rolle erstellen“.
3. Wählen Sie Lambda als Dienst aus, der diese Rolle verwenden soll. Klicken Sie auf „Weiter“.
4. Fügen Sie die folgenden Richtlinien bei (Beachten Sie, dass diese Richtlinien ausschließlich zu Demonstrationszwecken ausgewählt wurden. In einer echten Produktionsumgebung sollten Sie die Berechtigungen nur auf diejenigen beschränken, die benötigt werden.):

   - Amazonen S 3 ReadOnlyAccess
   - AmazonRekognitionReadOnlyAccess
   - AmazonDynamoDBFullZugriff

- SESFullZugriff auf Amazon
- AWSLambdaAusführen
- AWSLambdaBasicExecutionRole (zum CloudWatch Loggen)

Klicken Sie auf „Weiter“.

5. Geben Sie der Rolle einen Namen und eine Beschreibung und erstellen Sie dann die Rolle, indem Sie „Rolle erstellen“ wählen.
6. Sobald wir die richtigen Berechtigungen für Lambda konfiguriert haben, können wir mit der AWS Management Console eine Instanz der Lambda-Funktion erstellen.

   Rufen Sie den Lambda-Service in der AWS-Managementkonsole auf.
7. Klicken Sie auf „Funktion erstellen“. Wählen Sie „Von Grund auf neu erstellen“.
8. Geben Sie einen Funktionsnamen ein und wählen Sie die Python-Laufzeit aus. Wählen Sie unter „Standardausführungsrolle ändern“ die Option „Eine bestehende Rolle verwenden“ und wählen Sie dann die IAM-Rolle aus, die Sie zuvor erstellt haben.
9. Jetzt müssen Sie den Code auf der Registerkarte Lambda-Funktionen erstellen und aktualisieren. Gehen Sie zur Registerkarte „lambda_function“ und ersetzen Sie den dortigen Code durch das nachfolgende Lambda-Codebeispiel.

   Speichern Sie die Änderungen und stellen Sie die Änderungen bereit.
10. Jetzt müssen Sie ein Amazon S3 S3-Ereignis als Lambda-Trigger konfigurieren.

    Gehen Sie auf der Registerkarte bzw. Seite „Konfiguration“ Ihrer Lambda-Funktion zu Trigger und klicken Sie auf „Trigger hinzufügen“.
11. Wählen Sie Amazon S3 aus der Liste der verfügbaren Auslöser aus.
12. Um den Trigger zu konfigurieren: Wählen Sie den Bucket aus, von dem aus die Funktion ausgelöst werden soll.

    Wählen Sie den Ereignistyp PUT. Optional können Sie ein Präfix oder Suffix angeben, wenn Sie nur Dateien mit bestimmten Namen oder Typen verarbeiten möchten.
13. Aktivieren Sie den Trigger, indem Sie auf „Hinzufügen“ klicken und speichern Sie die Konfiguration.

Amazon SES SES-E-Mail-Adressen verifizieren

Bevor Sie Amazon SES zum Senden von E-Mails verwenden können, müssen Sie die E-Mail-Adressen sowohl des Absenders als auch des Empfängers überprüfen. So gehen Sie vor:

1. Gehen Sie zur Amazon SES SES-Konsole. Navigieren Sie zu „Identity Management“ und dann zu „E-Mail-Adressen“.
2. Klicken Sie auf „Neue E-Mail-Adresse verifizieren“. Geben Sie die E-Mail-Adresse ein, die Sie verifizieren möchten, und klicken Sie auf „Diese E-Mail-Adresse verifizieren“. Sie erhalten eine E-Mail mit einem Bestätigungslink. Klicken Sie auf den Link, um den Bestätigungsprozess abzuschließen.
3. Nachdem Sie die E-Mail-Adressen für beide Konten verifiziert, Ihre Lambda-Funktion mit den richtigen Berechtigungen eingerichtet und einige Beispielkundendaten erstellt haben, können Sie die Lambda-Funktion testen, indem Sie ein Testimage in den ausgewählten Amazon S3 S3-Bucket hochladen.

Bilder mit Metadaten auf Amazon S3 hochladen

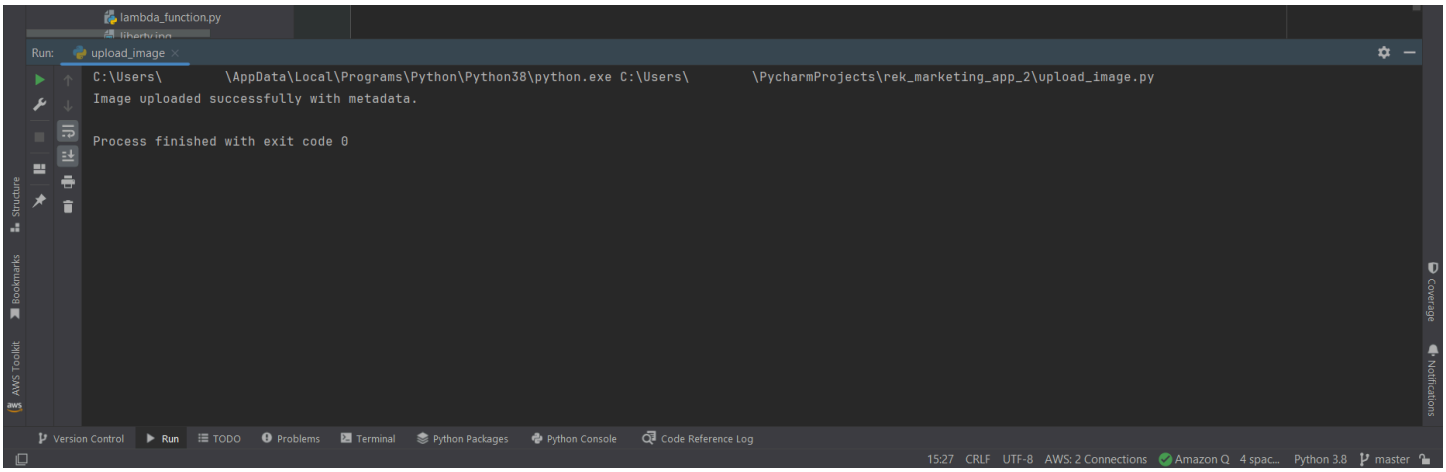
Jetzt können wir mithilfe der AWS Konsole oder des zuvor erstellten Skripts ein Test-Image in den von uns ausgewählten Amazon S3 S3-Bucket hochladen. Schreiben Sie ein Skript, das ein Bild in den Amazon S3 S3-Bucket hochlädt, den Sie zuvor in Ihrer Lambda-Funktion angegeben haben. Geben Sie im folgenden Codebeispiel den Pfad Ihres Images als erstes Argument für „upload_image_to_s3", den Bucket-Namen als zweites Argument und die Kontonummer des Benutzers, der das Bild hochlädt, als letztes Argument an.

```
import boto3
def upload_image_to_s3(file_name, bucket, account_number):
    s3 = boto3.client('s3')
    try:
        with open(file_name, 'rb') as data:
            s3.upload_fileobj(
                Fileobj=data,
                Bucket=bucket,
                Key=file_name,
                ExtraArgs={
                    'Metadata': {'account_number': account_number}
                }
            )
        print("Image uploaded successfully with metadata.")
```

```
    except Exception as e:
        print("Failed to upload image")
        print(e)

# Usage
upload_image_to_s3('path-to-image-here', 's3-bucket-name-here', 'user-account-number-
here')
```

In dieser Funktion wird der `ExtraArgs` Parameter in der `upload_fileobj` Methode verwendet, um benutzerdefinierte Metadaten () zusammen mit dem Bild einzuschließen. `account_number` Diese Metadaten können später verwendet werden AWS , um das Bild entsprechend zu verarbeiten.

Speichern Sie das Skript und führen Sie es aus. Dadurch wird das Bild hochgeladen.



Wenige Minuten nach dem Hochladen des Bildes sollten Sie eine E-Mail an die Adresse erhalten, die Sie zuvor mit dem hier angegebenen Konto verknüpft haben.

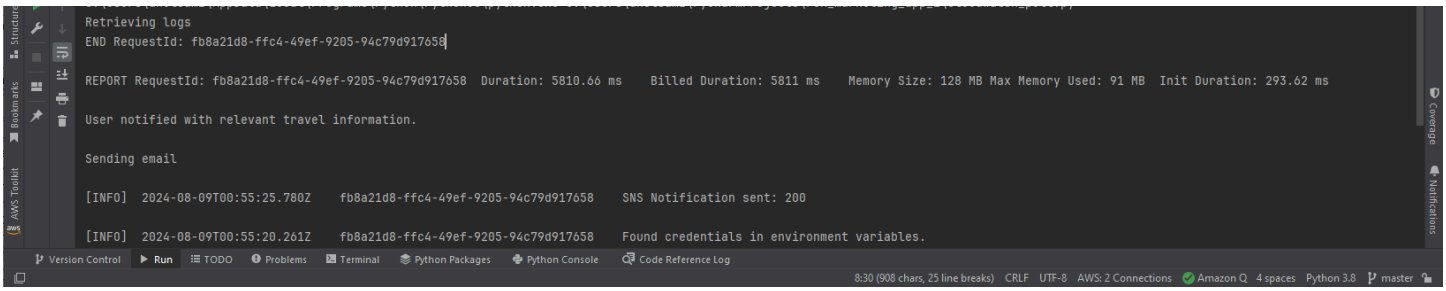
CloudWatch Abfrageprotokolle vom Client

Prüfen Sie in den CloudWatch Protokollen Ihrer Lambda-Funktion, ob sie wie erwartet ausgelöst und ausgeführt wurde. Protokolle finden Sie unter CloudWatch > Logs > Log groups >/AWS/lambda/ your_function_name. Sie können auch ein Skript schreiben, um auf die Protokolle zuzugreifen und sie auszudrucken. Das folgende Codebeispiel fragt die Lambda-Gruppe nach Protokollen ab und druckt die Protokolle aus, die in der letzten Stunde generiert wurden. Speichern Sie den Code und führen Sie ihn aus.

```
import boto3
import time
```

```
def fetch_lambda_logs(log_group_name, start_time):
    client = boto3.client('logs')
    query = "fields @timestamp, @message | sort @timestamp desc | limit 20"
    start_query_response = client.start_query(
        logGroupName=log_group_name,
        startTime=int(start_time),
        endTime=int(time.time()),
        queryString=query,
    )
    query_id = start_query_response['queryId']
    response = None
    while response is None or response['status'] == 'Running':
        time.sleep(1)  # Wait for 1 second before checking the query status again
        response = client.get_query_results(queryId=query_id)
    return response['results']

# Usage
log_group = '/aws/lambda/RekMediaFunction'
logs = fetch_lambda_logs(log_group, int(time.time()) - 3600)  # Fetch logs from the
 last hour
print("Retrieving logs")
for log in logs:
    #print(log)
    print(log[1]['value'])
```

Wenn Sie den Code ausführen, sollten die Protokolle ausgedruckt werden und Sie sollten sehen können, dass der Benutzer per E-Mail mit den entsprechenden Reiseinformationen benachrichtigt wurde.



Sie haben erfolgreich eine Anwendung erstellt, die Labels in Bildern erkennen kann, die in einen Amazon S3 S3-Bucket hochgeladen wurden, und anschließend dem Benutzer, der das Bild hochgeladen hat, eine Werbebotschaft per E-Mail senden kann. Stellen Sie sicher, dass Sie alle Ressourcen löschen, die Sie nicht mehr benötigen, damit sie Ihnen nicht in Rechnung gestellt werden.

Label in einem Bild mit Lambda und Python erkennen

AWS Lambda ist ein Rechendienst, mit dem Sie Code ausführen können, ohne Server bereitzustellen oder zu verwalten. Sie können Rekognition-API-Operationen von einer Lambda-Funktion aus aufrufen. Die folgenden Anweisungen zeigen, wie eine Lambda-Funktion in Python erstellt wird, die `DetectLabels` aufruft.

Die Lambda-Funktion ruft `DetectLabels` auf und gibt ein Array der im Bild erkannten Markierungen zurück, sowie den Vertrauensgrad, mit dem sie erkannt wurden.

Die Anweisungen enthalten einen Python-Beispielcode, der Ihnen zeigt, wie Sie die Lambda-Funktion aufrufen und sie mit einem Bild versorgen, das aus einem Amazon-S3-Bucket oder Ihrem lokalen Computer stammt.

Stellen Sie sicher, dass die von Ihnen ausgewählten Bilder die Limits von Rekognition einhalten. Informationen zu Bilddateityp - und Größenbeschränkungen finden Sie unter Richtlinien und Kontingente in Rekognition und in der DetectLabels API-Referenz.

Erstellen einer Lambda-Funktion (Konsole)

In diesem Schritt erstellen Sie eine leere Lambda-Funktion und eine IAM-Ausführungsrolle, mit der Ihre Lambda-Funktion die `DetectLabels`-Operation aufrufen kann. In späteren Schritten fügen Sie den Quellcode und optional eine Ebene zur Lambda-Funktion hinzu.

Wenn Sie Dokumente verwenden, die in einem Amazon-S3-Bucket gespeichert sind, zeigt dieser Schritt auch, wie Sie Zugriff auf den Bucket gewähren, in dem Ihre Dokumente gespeichert sind.

So erstellen Sie eine AWS Lambda Funktion (Konsole)

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die AWS Lambda Konsole unter https://console.aws.amazon.com/lambda/.
2. Wählen Sie Funktion erstellen aus. Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.
3. Wählen Sie aus den folgenden Optionen:

   - Wählen Sie Von Grund auf neu schreiben aus.
   - Geben Sie einen Wert für Funktionsname ein.
   - Wählen Sie für Laufzeit die neueste Version von Python aus.
   - Wählen Sie für Architektur x86_64 aus.

4. Wählen Sie Funktion erstellen, um die AWS Lambda -Funktion zu erstellen.
5. Wählen Sie auf der Seite der Funktion die Registerkarte Konfiguration.
6. Wählen Sie im Bereich Berechtigungen unter Ausführungsrolle den Rollennamen aus, um die Rolle in der IAM-Konsole zu öffnen.
7. Wählen Sie auf der Registerkarte Berechtigungen die Optionen Berechtigungen hinzufügen und dann (Inline-Richtlinie erstellen aus.
8. Wählen Sie die Registerkarte JSON und ersetzen Sie die Richtlinie durch die folgende Richtlinie:

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

9. Wählen Sie Richtlinie prüfen.
10. Geben Sie einen Namen für die Richtlinie ein, zum Beispiel DetectLabels-access.
11. Wählen Sie Create Policy (Richtlinie erstellen) aus.
12. Wenn Sie Dokumente zur Analyse in einem Amazon-S3-Bucket speichern, müssen Sie eine Amazon-S3-Zugriffsrichtlinie hinzufügen. Wiederholen Sie dazu die Schritte 7 bis 11 in der AWS Lambda -Konsole und nehmen Sie die folgenden Änderungen vor.

    a. Verwenden Sie für Schritt 8 die folgende Richtlinie. *bucket/folder path*Ersetzen Sie durch den Amazon S3 S3-Bucket und den Ordnerpfad zu den Dokumenten, die Sie analysieren möchten.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3Access",
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::bucket/folder path/*"
```

```
        }
    ]
}
```

b. Wählen Sie für Schritt 10 einen anderen Richtliniennamen, z. B. S3Bucket-access.

(Optional) Erstellen einer Ebene (Konsole)

Sie müssen diesen Schritt nicht ausführen, um eine Lambda-Funktion zu verwenden und `DetectLabels` aufzurufen.

Die `DetectLabels` Operation ist in der standardmäßigen Lambda-Python-Umgebung als Teil des AWS SDK für Python (Boto3) enthalten.

Wenn für andere Teile Ihrer Lambda-Funktion aktuelle AWS Service-Updates erforderlich sind, die sich nicht in der Standard-Lambda-Python-Umgebung befinden, können Sie diesen Schritt ausführen, um die neueste Boto3-SDK-Version als Ebene zu Ihrer Funktion hinzuzufügen.

Um das SDK als Ebene hinzuzufügen, erstellen Sie zunächst ein ZIP-Dateiarchiv, das das Boto3-SDK enthält. Anschließend erstellen Sie eine Ebene und fügen der Ebene das ZIP-Dateiarchiv hinzu. Weitere Informationen finden Sie unter Verwenden von Ebenen mit Ihrer Lambda-Funktion.

So erstellen Sie eine Ebene und fügen eine hinzu (Konsole)

1. Öffnen Sie eine Befehlszeile und geben Sie die folgenden Befehle ein, um ein Bereitstellungspaket mit der neuesten Version des SDK zu erstellen. AWS

   ```
   pip install boto3 --target python/.
   zip boto3-layer.zip -r python/
   ```

2. Notieren Sie sich den Namen der ZIP-Datei (boto3-layer.zip), die Sie in Schritt 8 dieses Verfahrens verwenden.
3. Öffnen Sie die AWS Lambda Konsole unter https://console.aws.amazon.com/lambda/.
4. Wählen Sie im Navigationsbereich Ebenen aus.
5. Wählen Sie Ebene erstellen aus.
6. Geben Sie einen Namen und eine Beschreibung ein.
7. Wählen Sie als Code-Eingabetyp Hochladen einer ZIP-Datei und wählen Sie Hochladen.

8. Wählen Sie im Dialogfeld das ZIP-Dateiarchiv (boto3-layer.zip) aus, das Sie in Schritt 1 dieses Verfahrens erstellt haben.
9. Wählen Sie für Kompatible Laufzeiten die neueste Version von Python aus.
10. Wählen Sie Erstellen, um die Ebene zu erstellen.
11. Wählen Sie das Menüsymbol im Navigationsbereich.
12. Wählen Sie im Navigationsbereich Funktionen aus.
13. Wählen Sie in der Ressourcenliste die Funktion aus, die Sie zuvor in ??? erstellt haben.
14. Wählen Sie die Registerkarte Code.
15. Wählen Sie im Bereich Ebenen die Option Ebene hinzufügen aus.
16. Wählen Sie Benutzerdefinierte Ebenen.
17. Wählen Sie unter Benutzerdefinierte Ebenen den Namen der Ebene aus, den Sie in Schritt 6 eingegeben haben.
18. Wählen Sie unter Version die Version der Ebene aus, die 1 sein sollte.
19. Wählen Sie Hinzufügen aus.

Python-Code hinzufügen (Konsole)

In diesem Schritt fügen Sie Ihren Python-Code über den Code-Editor der Lambda-Konsole zu Ihrer Lambda-Funktion hinzu. Der Code erkennt mithilfe der `DetectLabels`-Operation Labels in einem Bild. Er gibt eine Reihe von Label zurück, die im Bild erkannt wurden, sowie den Grad der Zuverlässigkeit der erkannten Label.

Das Dokument, das Sie für die `DetectLabels`-Operation bereitstellen, kann sich in einem Amazon-S3-Bucket oder auf einem lokalen Computer befinden.

So fügen Sie Python-Code hinzu (Konsole)

1. Navigieren Sie zur Registerkarte Code.
2. Ersetzen Sie im Code-Editor den Code in lambda_function.py durch den folgenden Code:

```
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. Wählen Sie Bereitstellen aus, um Ihre Lambda-Funktion bereitzustellen.

So fügen Sie Python-Code hinzu (Konsole)

Nach dem Erstellen Ihrer Lambda-Funktion können Sie sie aufrufen, um Labels in einem Bild zu erkennen.

In diesem Schritt führen Sie Python-Code auf Ihrem Computer aus, der ein lokales Bild oder ein Bild in einem Amazon-S3-Bucket an Ihre Lambda-Funktion übergibt.

Stellen Sie sicher, dass Sie den Code in derselben AWS Region ausführen, in der Sie die Lambda-Funktion erstellt haben. Sie können die AWS Region für Ihre Lambda-Funktion in der Navigationsleiste der Funktionsdetailseite in der Lambda-Konsole anzeigen.

Wenn die Lambda-Funktion einen Zeitüberschreitungsfehler zurückgibt, verlängern Sie den Zeitüberschreitungs-Zeitraum für die Lambda-Funktion. Weitere Informationen erhalten Sie unter Zeitüberschreitung der Funktion konfigurieren (Konsole).

Weitere Informationen zum Aufrufen einer Lambda-Funktion aus Ihrem Code finden Sie unter Aufrufen von AWS-Lambda-Funktionen.

So testen Sie Ihre Lambda-Funktion

1. Sofern Sie das noch nicht getan haben, führen Sie die folgenden Schritte aus:

   a. Stellen Sie sicher, dass der Benutzer über die entsprechende `lambda:InvokeFunction`-Berechtigung verfügt. Sie können die folgende Richtlinie verwenden:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "InvokeLambda",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "ARN for lambda function"
        }
    ]
}
```

   Sie können den ARN für Ihre Lambda-Funktion in der Funktionsübersicht in der Lambda-Konsole abrufen.

   Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

   - Benutzer und Gruppen in: AWS IAM Identity Center

     Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center -Benutzerhandbuch.

   - Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

     Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter Eine Rolle für einen externen Identitätsanbieter (Verbund) erstellen im IAM-Benutzerhandbuch.

   - IAM-Benutzer:

     - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter Eine Rolle für einen IAM-Benutzer erstellen im IAM-Benutzerhandbuch.

- (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

b. Installieren und konfigurieren Sie AWS das SDK für Python. Weitere Informationen finden Sie unter Schritt 2: Richten Sie das AWS CLI und ein AWS SDKs.

2. Speichern Sie den folgenden Code in eine Datei mit dem Namen `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. Führen Sie den Code aus. Wenn sich das Dokument in einem Amazon-S3-Bucket befindet, stellen Sie sicher, dass es sich um denselben Bucket handelt, den Sie zuvor in Schritt 12 ??? von angegeben haben.

Bei Erfolg gibt Ihr Code für jeden im Dokument erkannten Blocktyp eine teilweise JSON-Antwort zurück.

Codebeispiele für Amazon Rekognition mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Amazon Rekognition mit einem AWS Software Development Kit (SDK) verwendet wird. Die Codebeispiele in diesem Kapitel sollen die Codebeispiele im Rest dieses Handbuchs ergänzen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarios anzeigen.

Szenarien sind Code-Beispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erste Schritte

Hallo Amazon Rekognition

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Rekognition beginnen können.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rekognition)
```

```
# Set this project's name.
project("hello_rekognition")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                 # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_rekognition.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei 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;
}
```

- Einzelheiten zur API finden Sie ListCollectionsunter AWS SDK für C++ API-Referenz.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Rekognition AWS SDKs
  - Hallo Amazon Rekognition
  - Aktionen für Amazon Rekognition mit AWS SDKs
    - Verwendung CompareFaces mit einem AWS SDK oder CLI
    - Verwendung CreateCollection mit einem AWS SDK oder CLI
    - Verwendung DeleteCollection mit einem AWS SDK oder CLI
    - Verwendung DeleteFaces mit einem AWS SDK oder CLI
    - Verwendung DescribeCollection mit einem AWS SDK oder CLI
    - Verwendung DetectFaces mit einem AWS SDK oder CLI
    - Verwendung DetectLabels mit einem AWS SDK oder CLI
    - Verwendung DetectModerationLabels mit einem AWS SDK oder CLI
    - Verwendung DetectText mit einem AWS SDK oder CLI
    - Verwendung DisassociateFaces mit einem AWS SDK oder CLI
    - Verwendung GetCelebrityInfo mit einem AWS SDK oder CLI
    - Verwendung IndexFaces mit einem AWS SDK oder CLI
    - Verwendung ListCollections mit einem AWS SDK oder CLI
    - Verwendung ListFaces mit einem AWS SDK oder CLI
    - Verwendung RecognizeCelebrities mit einem AWS SDK oder CLI
    - Verwendung SearchFaces mit einem AWS SDK oder CLI
    - Verwendung SearchFacesByImage mit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon Rekognition AWS SDKs

- Erstellen Sie eine Amazon Rekognition Rekognition-Sammlung und suchen Sie darin mithilfe eines SDK nach Gesichtern AWS
- Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
- Ermitteln Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines SDK AWS
- Erkennen und Anzeigen von Elementen in Bildern mit Amazon Rekognition mithilfe eines SDK AWS
- Erkennen Sie Gesichter in einem Bild mithilfe eines AWS SDK
- Informationen in Videos mithilfe von Amazon Rekognition und dem SDK erkennen AWS
- Objekte in Bildern mit Amazon Rekognition mithilfe eines SDK erkennen AWS
- Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines SDK AWS
- Speichern Sie EXIF und andere Bildinformationen mit einem SDK AWS

Grundlegende Beispiele für die Verwendung von Amazon Rekognition AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon Rekognition mit verwenden können. AWS SDKs

Beispiele

- Hallo Amazon Rekognition
- Aktionen für Amazon Rekognition mit AWS SDKs
  - Verwendung CompareFaces mit einem AWS SDK oder CLI
  - Verwendung CreateCollection mit einem AWS SDK oder CLI
  - Verwendung DeleteCollection mit einem AWS SDK oder CLI
  - Verwendung DeleteFaces mit einem AWS SDK oder CLI
  - Verwendung DescribeCollection mit einem AWS SDK oder CLI
  - Verwendung DetectFaces mit einem AWS SDK oder CLI
  - Verwendung DetectLabels mit einem AWS SDK oder CLI
  - Verwendung DetectModerationLabels mit einem AWS SDK oder CLI

- Verwendung DetectText mit einem AWS SDK oder CLI
- Verwendung DisassociateFaces mit einem AWS SDK oder CLI
- Verwendung GetCelebrityInfo mit einem AWS SDK oder CLI
- Verwendung IndexFaces mit einem AWS SDK oder CLI
- Verwendung ListCollections mit einem AWS SDK oder CLI
- Verwendung ListFaces mit einem AWS SDK oder CLI
- Verwendung RecognizeCelebrities mit einem AWS SDK oder CLI
- Verwendung SearchFaces mit einem AWS SDK oder CLI
- Verwendung SearchFacesByImage mit einem AWS SDK oder CLI

Hallo Amazon Rekognition

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Rekognition beginnen können.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rekognition)

# Set this project's name.
project("hello_rekognition")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)
```

```
# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_rekognition.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei 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;
}
```

- Einzelheiten zur API finden Sie ListCollectionsunter AWS SDK für C++ API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterRekognition mit einem SDK verwenden AWS. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Aktionen für Amazon Rekognition mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie einzelne Amazon Rekognition Rekognition-Aktionen mit ausführen können. AWS SDKs Jedes Beispiel enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-Rekognition-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Rekognition AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon-Rekognition-API-Referenz.

Beispiele

- Verwendung CompareFaces mit einem AWS SDK oder CLI
- Verwendung CreateCollection mit einem AWS SDK oder CLI
- Verwendung DeleteCollection mit einem AWS SDK oder CLI
- Verwendung DeleteFaces mit einem AWS SDK oder CLI
- Verwendung DescribeCollection mit einem AWS SDK oder CLI
- Verwendung DetectFaces mit einem AWS SDK oder CLI
- Verwendung DetectLabels mit einem AWS SDK oder CLI
- Verwendung DetectModerationLabels mit einem AWS SDK oder CLI
- Verwendung DetectText mit einem AWS SDK oder CLI
- Verwendung DisassociateFaces mit einem AWS SDK oder CLI
- Verwendung GetCelebrityInfo mit einem AWS SDK oder CLI
- Verwendung IndexFaces mit einem AWS SDK oder CLI

- Verwendung ListCollections mit einem AWS SDK oder CLI
- Verwendung ListFaces mit einem AWS SDK oder CLI
- Verwendung RecognizeCelebrities mit einem AWS SDK oder CLI
- Verwendung SearchFaces mit einem AWS SDK oder CLI
- Verwendung SearchFacesByImage mit einem AWS SDK oder CLI

Verwendung **`CompareFaces`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CompareFaces` verwendet wird.

Weitere Informationen finden Sie unter Vergleich von Gesichtern in Bildern.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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.");
        }
    }
```

- Einzelheiten zur API finden Sie CompareFacesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um Gesichter in zwei Bildern zu vergleichen

Der folgende `compare-faces` Befehl vergleicht Gesichter in zwei Bildern, die in einem Amazon S3 S3-Bucket gespeichert sind.

```
aws rekognition compare-faces \
    --source-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"source.jpg"}}' \
    --target-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"target.jpg"}}'
```

Ausgabe:

```
{
    "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
    }
}
```

Weitere Informationen finden Sie unter Gesichter in Bildern vergleichen im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie CompareFacesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

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

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

            Where:
                bucketName - The name of the S3 bucket where the images are
 stored.
                sourceKey  - The S3 key (file name) for the source image.
                targetKey  - The S3 key (file name) for the target image.
           """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceKey = args[1];
        String targetKey = args[2];

        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
        compareTwoFaces(rekClient, bucketName, sourceKey, targetKey);
    }

   /**
    * Compares two faces from images stored in an Amazon S3 bucket using AWS
 Rekognition.
    *
    * <p>This method takes two image keys from an S3 bucket and compares the
 faces within them.
```

```
    * It prints out the confidence level of matched faces and reports the number
of unmatched faces.</p>
    *
    * @param rekClient   The {@link RekognitionClient} used to call AWS
Rekognition.
    * @param bucketName  The name of the S3 bucket containing the images.
    * @param sourceKey   The object key (file path) for the source image in the
S3 bucket.
    * @param targetKey   The object key (file path) for the target image in the
S3 bucket.
    * @throws RuntimeException If the Rekognition service returns an error.
    */
   public static void compareTwoFaces(RekognitionClient rekClient, String
bucketName, String sourceKey, String targetKey) {
       try {
           Float similarityThreshold = 70F;
           S3Object s3ObjectSource = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceKey)
                   .build();

           Image sourceImage = Image.builder()
                   .s3Object(s3ObjectSource)
                   .build();

           S3Object s3ObjectTarget = S3Object.builder()
                   .bucket(bucketName)
                   .name(targetKey)
                   .build();

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

           CompareFacesRequest facesRequest = CompareFacesRequest.builder()
                   .sourceImage(sourceImage)
                   .targetImage(targetImage)
                   .similarityThreshold(similarityThreshold)
                   .build();

           // Compare the two images.
           CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
```

```
            List<CompareFacesMatch> faceDetails =
 compareFacesResult.faceMatches();

            for (CompareFacesMatch match : faceDetails) {
                ComparedFace face = match.face();
                BoundingBox position = face.boundingBox();
                System.out.println("Face at " + position.left().toString()
                        + " " + position.top()
                        + " matches with " + face.confidence().toString()
                        + "% confidence.");
            }

            List<ComparedFace> unmatchedFaces =
 compareFacesResult.unmatchedFaces();
            System.out.println("There were " + unmatchedFaces.size() + " face(s)
 that did not match.");

        } catch (RekognitionException e) {
            System.err.println("Error comparing faces: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Einzelheiten zur API finden Sie CompareFacesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun compareTwoFaces(
    similarityThresholdVal: Float,
    sourceImageVal: String,
    targetImageVal: String,
```

```
) {
    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage =
        Image {
            bytes = sourceBytes
        }

    val tarImage =
        Image {
            bytes = targetBytes
        }

    val facesRequest =
        CompareFacesRequest {
            sourceImage = souImage
            targetImage = tarImage
            similarityThreshold = similarityThresholdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->

        val compareFacesResult = rekClient.compareFaces(facesRequest)
        val faceDetails = compareFacesResult.faceMatches

        if (faceDetails != null) {
            for (match: CompareFacesMatch in faceDetails) {
                val face = match.face
                val position = face?.boundingBox
                if (position != null) {
                    println("Face at ${position.left} ${position.top} matches
 with ${face.confidence} % confidence.")
                }
            }
        }

        val uncompared = compareFacesResult.unmatchedFaces
        if (uncompared != null) {
            println("There was ${uncompared.size} face(s) that did not match")
        }
```

```
        println("Source image rotation:
 ${compareFacesResult.sourceImageOrientationCorrection}")
        println("target image rotation:
 ${compareFacesResult.targetImageOrientationCorrection}")
    }
}
```

- Einzelheiten zur API finden Sie CompareFacesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie CompareFacesin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`CreateCollection`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateCollection` verwendet wird.

Weitere Informationen finden Sie unter Erstellen einer Sammlung.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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}");
        }
    }
```

- Einzelheiten zur API finden Sie CreateCollectionin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um eine Sammlung zu erstellen

Der folgende `create-collection` Befehl erstellt eine Sammlung mit dem angegebenen Namen.

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

Ausgabe:

```
{
    "CollectionArn": "aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
    "FaceModelVersion": "4.0",
    "StatusCode": 200
}
```

Weitere Informationen finden Sie unter Creating a Collection im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie unter CreateCollection AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

            Usage: <collectionName>\s

            Where:
                collectionName - The name of the collection.\s
            """;

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

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

        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Creates a new Amazon Rekognition collection.
     *
     * @param rekClient    the Amazon Rekognition client used to interact with
 the Rekognition service
     * @param collectionId the unique identifier for the collection to be created
     */
    public static void createMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            CreateCollectionRequest collectionRequest =
 CreateCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            CreateCollectionResponse collectionResponse =
 rekClient.createCollection(collectionRequest);
            System.out.println("CollectionArn: " +
 collectionResponse.collectionArn());
            System.out.println("Status code: " +
 collectionResponse.statusCode().toString());

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

- Einzelheiten zur API finden Sie CreateCollectionin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- Einzelheiten zur API finden Sie CreateCollectionin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie CreateCollectionin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DeleteCollection`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteCollection` verwendet wird.

Weitere Informationen finden Sie unter Löschen einer Sammlung.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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}");
    }
```

```
    }
```

- Einzelheiten zur API finden Sie DeleteCollectionin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um eine Sammlung zu löschen

Der folgende `delete-collection` Befehl löscht die angegebene Sammlung.

```
aws rekognition delete-collection \
    --collection-id MyCollection
```

Ausgabe:

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

Weitere Informationen finden Sie unter Löschen einer Sammlung im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie DeleteCollectionin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
```

```
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

            Where:
                collectionId - The id of the collection to delete.\s
            """;

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

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

        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Deletes an Amazon Rekognition collection.
     *
     * @param rekClient      An instance of the {@link RekognitionClient} class,
 which is used to interact with the Amazon Rekognition service.
```

```
     * @param collectionId   The ID of the collection to be deleted.
     */
    public static void deleteMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            DeleteCollectionRequest deleteCollectionRequest =
 DeleteCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            DeleteCollectionResponse deleteCollectionResponse =
 rekClient.deleteCollection(deleteCollectionRequest);
            System.out.println(collectionId + ": " +
 deleteCollectionResponse.statusCode().toString());

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

- Einzelheiten zur API finden Sie DeleteCollectionin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
```

```
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- Einzelheiten zur API finden Sie DeleteCollectionin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie DeleteCollectionin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DeleteFaces`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteFaces` verwendet wird.

Weitere Informationen finden Sie unter Löschen von Gesichtern aus einer Sammlung.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to delete one or more faces from
    /// a Rekognition collection.
    /// </summary>
    public class DeleteFaces
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection";
            var faces = new List<string> { "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx" };

            var rekognitionClient = new AmazonRekognitionClient();

            var deleteFacesRequest = new DeleteFacesRequest()
            {
                CollectionId = collectionId,
                FaceIds = faces,
            };

            DeleteFacesResponse deleteFacesResponse = await
 rekognitionClient.DeleteFacesAsync(deleteFacesRequest);
            deleteFacesResponse.DeletedFaces.ForEach(face =>
            {
```

```
                Console.WriteLine($"FaceID: {face}");
            });
        }
    }
```

- Einzelheiten zur API finden Sie DeleteFacesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um Gesichter aus einer Sammlung zu löschen

Der folgende `delete-faces` Befehl löscht die angegebene Fläche aus einer Sammlung.

```
aws rekognition delete-faces \
    --collection-id MyCollection
    --face-ids '["0040279c-0178-436e-b70a-e61b074e96b0"]'
```

Ausgabe:

```
{
    "DeletedFaces": [
        "0040279c-0178-436e-b70a-e61b074e96b0"
    ]
}
```

Weitere Informationen finden Sie unter Löschen von Gesichtern aus einer Sammlung im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie unter DeleteFaces AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

            Where:
                collectionId - The id of the collection from which faces are
 deleted.\s
                faceId - The id of the face to delete.\s
           """;

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

        String collectionId = args[0];
        String faceId = args[1];
```

```
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }

    /**
     * Deletes a face from the specified Amazon Rekognition collection.
     *
     * @param rekClient     an instance of the Amazon Rekognition client
     * @param collectionId  the ID of the collection from which the face should
 be deleted
     * @param faceId        the ID of the face to be deleted
     * @throws RekognitionException if an error occurs while deleting the face
     */
    public static void deleteFacesCollection(RekognitionClient rekClient,
            String collectionId,
            String faceId) {

        try {
            DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceIds(faceId)
                    .build();

            rekClient.deleteFaces(deleteFacesRequest);
            System.out.println("The face was deleted from the collection.");

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

- Einzelheiten zur API finden Sie DeleteFacesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
        println("$faceIdVal was deleted from the collection")
    }
}
```

- Einzelheiten zur API finden Sie DeleteFacesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie DeleteFacesin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DescribeCollection`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeCollection` verwendet wird.

Weitere Informationen finden Sie unter Beschreiben einer Sammlung.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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}");
        }
    }
```

- Einzelheiten zur API finden Sie DescribeCollectionin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um eine Sammlung zu beschreiben

Im folgenden `describe-collection` Beispiel werden die Details zur angegebenen Sammlung angezeigt.

```
aws rekognition describe-collection \
    --collection-id MyCollection
```

Ausgabe:

```
{
    "FaceCount": 200,
    "CreationTimestamp": 1569444828.274,
    "CollectionARN": "arn:aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
    "FaceModelVersion": "4.0"
}
```

Weitere Informationen finden Sie unter Beschreibung einer Sammlung im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie DescribeCollectionin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
```

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

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
            """;

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

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

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    /**
     * Describes an Amazon Rekognition collection.
     *
     * @param rekClient         The Amazon Rekognition client used to make the
 request.
     * @param collectionName    The name of the collection to describe.
     *
     * @throws RekognitionException If an error occurs while describing the
 collection.
     */
    public static void describeColl(RekognitionClient rekClient, String
 collectionName) {
        try {
```

```
            DescribeCollectionRequest describeCollectionRequest =
 DescribeCollectionRequest.builder()
                    .collectionId(collectionName)
                    .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
                    .describeCollection(describeCollectionRequest);
            System.out.println("Collection Arn : " +
 describeCollectionResponse.collectionARN());
            System.out.println("Created : " +
 describeCollectionResponse.creationTimestamp().toString());

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

- Einzelheiten zur API finden Sie DescribeCollectionin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
```

```
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

- Einzelheiten zur API finden Sie DescribeCollectionin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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()
```

- Einzelheiten zur API finden Sie DescribeCollectionin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DetectFaces`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectFaces` verwendet wird.

Weitere Informationen finden Sie unter Erkennen von Gesichtern in einem Bild.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DetectFaces
{
    public static async Task Main()
    {
        string photo = "input.jpg";
        string bucket = "amzn-s3-demo-bucket";

        var rekognitionClient = new AmazonRekognitionClient();
```

```
            var detectFacesRequest = new DetectFacesRequest()
            {
                Image = new Image()
                {
                    S3Object = new S3Object()
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },

                // Attributes can be "ALL" or "DEFAULT".
                // "DEFAULT": BoundingBox, Confidence, Landmarks, Pose, and
 Quality.
                // "ALL": See https://docs.aws.amazon.com/sdkfornet/v3/apidocs/
items/Rekognition/TFaceDetail.html
                Attributes = new List<string>() { "ALL" },
            };

            try
            {
                DetectFacesResponse detectFacesResponse = await
 rekognitionClient.DetectFacesAsync(detectFacesRequest);
                bool hasAll = detectFacesRequest.Attributes.Contains("ALL");
                foreach (FaceDetail face in detectFacesResponse.FaceDetails)
                {
                    Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
 left={face.BoundingBox.Top} width={face.BoundingBox.Width}
 height={face.BoundingBox.Height}");
                    Console.WriteLine($"Confidence: {face.Confidence}");
                    Console.WriteLine($"Landmarks: {face.Landmarks.Count}");
                    Console.WriteLine($"Pose: pitch={face.Pose.Pitch}
 roll={face.Pose.Roll} yaw={face.Pose.Yaw}");
                    Console.WriteLine($"Brightness:
 {face.Quality.Brightness}\tSharpness: {face.Quality.Sharpness}");

                    if (hasAll)
                    {
                        Console.WriteLine($"Estimated age is between
 {face.AgeRange.Low} and {face.AgeRange.High} years old.");
                    }
                }
            }
```

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

Zeigt Informationen zum Begrenzungsrahmen für alle Gesichter in einem Bild an.

```
    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}");
        }
    }
```

- Einzelheiten zur API finden Sie DetectFacesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um Gesichter in einem Bild zu erkennen

Der folgende `detect-faces` Befehl erkennt Gesichter in dem angegebenen Bild, das in einem Amazon S3 S3-Bucket gespeichert ist.

```
aws rekognition detect-faces \
    --image '{"S3Object":{"Bucket":"MyImageS3Bucket","Name":"MyFriend.jpg"}}' \
    --attributes "ALL"
```

Ausgabe:

```
{
    "FaceDetails": [
        {
            "Confidence": 100.0,
            "Eyeglasses": {
```

```
                "Confidence": 98.91107940673828,
                "Value": false
            },
            "Sunglasses": {
                "Confidence": 99.7966537475586,
                "Value": false
            },
            "Gender": {
                "Confidence": 99.56611633300781,
                "Value": "Male"
            },
            "Landmarks": [
                {
                    "Y": 0.26721030473709106,
                    "X": 0.6204193830490112,
                    "Type": "eyeLeft"
                },
                {
                    "Y": 0.26831310987472534,
                    "X": 0.6776827573776245,
                    "Type": "eyeRight"
                },
                {
                    "Y": 0.3514654338359833,
                    "X": 0.6241428852081299,
                    "Type": "mouthLeft"
                },
                {
                    "Y": 0.35258132219314575,
                    "X": 0.6713621020317078,
                    "Type": "mouthRight"
                },
                {
                    "Y": 0.3140771687030792,
                    "X": 0.6428444981575012,
                    "Type": "nose"
                },
                {
                    "Y": 0.24662546813488007,
                    "X": 0.6001564860343933,
                    "Type": "leftEyeBrowLeft"
                },
                {
                    "Y": 0.24326619505882263,
```

```
                    "X": 0.6303644776344299,
                    "Type": "leftEyeBrowRight"
                },
                {
                    "Y": 0.23818562924861908,
                    "X": 0.6146903038024902,
                    "Type": "leftEyeBrowUp"
                },
                {
                    "Y": 0.24373626708984375,
                    "X": 0.6640064716339111,
                    "Type": "rightEyeBrowLeft"
                },
                {
                    "Y": 0.24877218902111053,
                    "X": 0.7025929093360901,
                    "Type": "rightEyeBrowRight"
                },
                {
                    "Y": 0.23938551545143127,
                    "X": 0.6823262572288513,
                    "Type": "rightEyeBrowUp"
                },
                {
                    "Y": 0.265746533870697,
                    "X": 0.6112898588180542,
                    "Type": "leftEyeLeft"
                },
                {
                    "Y": 0.2676128149032593,
                    "X": 0.6317071914672852,
                    "Type": "leftEyeRight"
                },
                {
                    "Y": 0.262735515832901,
                    "X": 0.6201658248901367,
                    "Type": "leftEyeUp"
                },
                {
                    "Y": 0.27025148272514343,
                    "X": 0.6206279993057251,
                    "Type": "leftEyeDown"
                },
                {
```

```
                    "Y": 0.268223375082016,
                    "X": 0.6658390760421753,
                    "Type": "rightEyeLeft"
                },
                {
                    "Y": 0.2672517001628876,
                    "X": 0.687832236289978,
                    "Type": "rightEyeRight"
                },
                {
                    "Y": 0.26383838057518005,
                    "X": 0.6769183874130249,
                    "Type": "rightEyeUp"
                },
                {
                    "Y": 0.27138751745224,
                    "X": 0.676596462726593,
                    "Type": "rightEyeDown"
                },
                {
                    "Y": 0.32283174991607666,
                    "X": 0.6350004076957703,
                    "Type": "noseLeft"
                },
                {
                    "Y": 0.3219289481639862,
                    "X": 0.6567046642303467,
                    "Type": "noseRight"
                },
                {
                    "Y": 0.3420318365097046,
                    "X": 0.6450609564781189,
                    "Type": "mouthUp"
                },
                {
                    "Y": 0.3664324879646301,
                    "X": 0.6455618143081665,
                    "Type": "mouthDown"
                },
                {
                    "Y": 0.26721030473709106,
                    "X": 0.6204193830490112,
                    "Type": "leftPupil"
                },
```

```
                {
                    "Y": 0.26831310987472534,
                    "X": 0.6776827573776245,
                    "Type": "rightPupil"
                },
                {
                    "Y": 0.26343393325805664,
                    "X": 0.5946047306060791,
                    "Type": "upperJawlineLeft"
                },
                {
                    "Y": 0.3543180525302887,
                    "X": 0.6044883728027344,
                    "Type": "midJawlineLeft"
                },
                {
                    "Y": 0.4084877669811249,
                    "X": 0.6477024555206299,
                    "Type": "chinBottom"
                },
                {
                    "Y": 0.3562754988670349,
                    "X": 0.707981526851654,
                    "Type": "midJawlineRight"
                },
                {
                    "Y": 0.26580461859703064,
                    "X": 0.7234612107276917,
                    "Type": "upperJawlineRight"
                }
            ],
            "Pose": {
                "Yaw": -3.7351467609405518,
                "Roll": -0.10309021919965744,
                "Pitch": 0.8637830018997192
            },
            "Emotions": [
                {
                    "Confidence": 8.74203109741211,
                    "Type": "SURPRISED"
                },
                {
                    "Confidence": 2.501944065093994,
                    "Type": "ANGRY"
```

```
                },
                {
                    "Confidence": 0.7378743290901184,
                    "Type": "DISGUSTED"
                },
                {
                    "Confidence": 3.5296201705932617,
                    "Type": "HAPPY"
                },
                {
                    "Confidence": 1.7162904739379883,
                    "Type": "SAD"
                },
                {
                    "Confidence": 9.518536567687988,
                    "Type": "CONFUSED"
                },
                {
                    "Confidence": 0.45474427938461304,
                    "Type": "FEAR"
                },
                {
                    "Confidence": 72.79895782470703,
                    "Type": "CALM"
                }
            ],
            "AgeRange": {
                "High": 48,
                "Low": 32
            },
            "EyesOpen": {
                "Confidence": 98.93987274169922,
                "Value": true
            },
            "BoundingBox": {
                "Width": 0.12368916720151901,
                "Top": 0.16007372736930847,
                "Left": 0.5901257991790771,
                "Height": 0.25140416622161865
            },
            "Smile": {
                "Confidence": 93.4493179321289,
                "Value": false
            },
```

```
            "MouthOpen": {
                "Confidence": 90.53053283691406,
                "Value": false
            },
            "Quality": {
                "Sharpness": 95.51618957519531,
                "Brightness": 65.29893493652344
            },
            "Mustache": {
                "Confidence": 89.85221099853516,
                "Value": false
            },
            "Beard": {
                "Confidence": 86.1991195678711,
                "Value": true
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Erkennen von Gesichtern in einem Bild im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie DetectFacesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

import java.util.List;
```

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

            Usage:   <bucketName> <sourceImage>

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

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

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

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

    /**
     * Detects faces in an image stored in an Amazon S3 bucket using the Amazon
 Rekognition service.
     *
     * @param rekClient    The Amazon Rekognition client used to interact with
 the Rekognition service.
```

```
    * @param bucketName   The name of the Amazon S3 bucket where the source
 image is stored.
    * @param sourceImage  The name of the source image file in the Amazon S3
 bucket.
    */
   public static void detectFacesinImage(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
       try {
           S3Object s3ObjectTarget = S3Object.builder()
               .bucket(bucketName)
               .name(sourceImage)
               .build();

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

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

           DetectFacesResponse facesResponse =
 rekClient.detectFaces(facesRequest);
           List<FaceDetail> faceDetails = facesResponse.faceDetails();
           for (FaceDetail face : faceDetails) {
               AgeRange ageRange = face.ageRange();
               System.out.println("The detected face is estimated to be between
 "
                       + ageRange.low().toString() + " and " +
 ageRange.high().toString()
                       + " years old.");

               System.out.println("There is a smile : " +
 face.smile().value().toString());
           }

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

- Einzelheiten zur API finden Sie DetectFacesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between
 ${ageRange?.low} and ${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectFacesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie DetectFacesin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DetectLabels`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectLabels` verwendet wird.

Weitere Informationen finden Sie unter Erkennen von Labels in einem Bild.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect labels within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DetectLabels
{
```

```
        public static async Task Main()
        {
            string photo = "del_river_02092020_01.jpg"; // "input.jpg";
            string bucket = "amzn-s3-demo-bucket"; // "bucket";

            var rekognitionClient = new AmazonRekognitionClient();

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

            try
            {
                DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (Label label in detectLabelsResponse.Labels)
                {
                    Console.WriteLine($"Name: {label.Name} Confidence:
{label.Confidence}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Erkennen Sie Labels in einer Bilddatei, die auf Ihrem Computer gespeichert ist.

```
   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);
            }
        }
    }
```

- Einzelheiten zur API finden Sie DetectLabelsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DetectLabelsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um ein Etikett in einem Bild zu erkennen

Das folgende `detect-labels` Beispiel erkennt Szenen und Objekte in einem Bild, das in einem Amazon S3 S3-Bucket gespeichert ist.

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

Ausgabe:

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

```
            "Name": "Transportation"
        },
        {
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.10616336017847061,
                        "Top": 0.5039216876029968,
                        "Left": 0.0037978808395564556,
                        "Height": 0.18528179824352264
                    },
                    "Confidence": 99.15271759033203
                },
                {
                    "BoundingBox": {
                        "Width": 0.2429988533258438,
                        "Top": 0.5251884460449219,
                        "Left": 0.7309805154800415,
                        "Height": 0.21577216684818268
                    },
                    "Confidence": 99.1286392211914
                },
                {
                    "BoundingBox": {
                        "Width": 0.14233611524105072,
                        "Top": 0.5333095788955688,
                        "Left": 0.6494812965393066,
                        "Height": 0.15528248250484467
                    },
                    "Confidence": 98.48368072509766
                },
                {
                    "BoundingBox": {
                        "Width": 0.11086395382881165,
                        "Top": 0.5354844927787781,
                        "Left": 0.10355594009160995,
                        "Height": 0.10271988064050674
                    },
                    "Confidence": 96.45606231689453
                },
                {
                    "BoundingBox": {
                        "Width": 0.06254628300666809,
                        "Top": 0.5573825240135193,
```

```
                        "Left": 0.46083059906959534,
                        "Height": 0.053911514580249786
                    },
                    "Confidence": 93.65448760986328
                },
                {
                    "BoundingBox": {
                        "Width": 0.10105438530445099,
                        "Top": 0.534368634223938,
                        "Left": 0.5743985772132874,
                        "Height": 0.12226245552301407
                    },
                    "Confidence": 93.06217193603516
                },
                {
                    "BoundingBox": {
                        "Width": 0.056389667093753815,
                        "Top": 0.5235804319381714,
                        "Left": 0.9427769780158997,
                        "Height": 0.17163699865341187
                    },
                    "Confidence": 92.6864013671875
                },
                {
                    "BoundingBox": {
                        "Width": 0.06003860384225845,
                        "Top": 0.5441341400146484,
                        "Left": 0.22409997880458832,
                        "Height": 0.06737709045410156
                    },
                    "Confidence": 90.4227066040039
                },
                {
                    "BoundingBox": {
                        "Width": 0.02848697081208229,
                        "Top": 0.5107086896896362,
                        "Left": 0,
                        "Height": 0.19150497019290924
                    },
                    "Confidence": 86.65286254882812
                },
                {
                    "BoundingBox": {
                        "Width": 0.04067881405353546,
```

```
                        "Top": 0.5566273927688599,
                        "Left": 0.316415935754776,
                        "Height": 0.03428703173995018
                    },
                    "Confidence": 85.36471557617188
                },
                {
                    "BoundingBox": {
                        "Width": 0.043411049991846085,
                        "Top": 0.5394920110702515,
                        "Left": 0.18293385207653046,
                        "Height": 0.0893595889210701
                    },
                    "Confidence": 82.21705627441406
                },
                {
                    "BoundingBox": {
                        "Width": 0.031183116137981415,
                        "Top": 0.5579366683959961,
                        "Left": 0.2853088080883026,
                        "Height": 0.03989990055561066
                    },
                    "Confidence": 81.0157470703125
                },
                {
                    "BoundingBox": {
                        "Width": 0.031113790348172188,
                        "Top": 0.5504819750785828,
                        "Left": 0.2580395042896271,
                        "Height": 0.056484755128622055
                    },
                    "Confidence": 56.13441467285156
                },
                {
                    "BoundingBox": {
                        "Width": 0.08586374670267105,
                        "Top": 0.5438792705535889,
                        "Left": 0.5128012895584106,
                        "Height": 0.08550430089235306
                    },
                    "Confidence": 52.37760925292969
                }
            ],
            "Confidence": 99.15271759033203,
```

```
            "Parents": [
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Car"
        },
        {
            "Instances": [],
            "Confidence": 98.9914321899414,
            "Parents": [],
            "Name": "Human"
        },
        {
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.19360728561878204,
                        "Top": 0.35072067379951477,
                        "Left": 0.43734854459762573,
                        "Height": 0.2742200493812561
                    },
                    "Confidence": 98.9914321899414
                },
                {
                    "BoundingBox": {
                        "Width": 0.03801717236638069,
                        "Top": 0.5010883808135986,
                        "Left": 0.9155802130699158,
                        "Height": 0.06597328186035156
                    },
                    "Confidence": 85.02790832519531
                }
            ],
            "Confidence": 98.9914321899414,
            "Parents": [],
            "Name": "Person"
        },
        {
            "Instances": [],
            "Confidence": 93.24951934814453,
```

```
            "Parents": [],
            "Name": "Machine"
        },
        {
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.03561960905790329,
                        "Top": 0.6468243598937988,
                        "Left": 0.7850857377052307,
                        "Height": 0.08878646790981293
                    },
                    "Confidence": 93.24951934814453
                },
                {
                    "BoundingBox": {
                        "Width": 0.02217046171426773,
                        "Top": 0.6149078607559204,
                        "Left": 0.04757237061858177,
                        "Height": 0.07136218994855881
                    },
                    "Confidence": 91.5025863647461
                },
                {
                    "BoundingBox": {
                        "Width": 0.016197510063648224,
                        "Top": 0.6274210214614868,
                        "Left": 0.6472989320755005,
                        "Height": 0.04955997318029404
                    },
                    "Confidence": 85.14686584472656
                },
                {
                    "BoundingBox": {
                        "Width": 0.020207518711686134,
                        "Top": 0.6348286867141724,
                        "Left": 0.7295016646385193,
                        "Height": 0.07059963047504425
                    },
                    "Confidence": 83.34547424316406
                },
                {
                    "BoundingBox": {
                        "Width": 0.020280985161662102,
```

```
                        "Top": 0.6171894669532776,
                        "Left": 0.08744934946298599,
                        "Height": 0.05297485366463661
                    },
                    "Confidence": 79.9981460571289
                },
                {
                    "BoundingBox": {
                        "Width": 0.018318990245461464,
                        "Top": 0.623889148235321,
                        "Left": 0.6836880445480347,
                        "Height": 0.06730121374130249
                    },
                    "Confidence": 78.87144470214844
                },
                {
                    "BoundingBox": {
                        "Width": 0.021310249343514442,
                        "Top": 0.6167286038398743,
                        "Left": 0.004064912907779217,
                        "Height": 0.08317798376083374
                    },
                    "Confidence": 75.89361572265625
                },
                {
                    "BoundingBox": {
                        "Width": 0.03604431077837944,
                        "Top": 0.7030032277107239,
                        "Left": 0.9254803657531738,
                        "Height": 0.04569442570209503
                    },
                    "Confidence": 64.402587890625
                },
                {
                    "BoundingBox": {
                        "Width": 0.009834849275648594,
                        "Top": 0.5821820497512817,
                        "Left": 0.28094568848609924,
                        "Height": 0.01964157074689865
                    },
                    "Confidence": 62.79907989501953
                },
                {
                    "BoundingBox": {
```

```
                    "Width": 0.01475677452981472,
                    "Top": 0.6137543320655823,
                    "Left": 0.5950819253921509,
                    "Height": 0.039063986390829086
                },
                "Confidence": 59.40483474731445
            }
        ],
        "Confidence": 93.24951934814453,
        "Parents": [
            {
                "Name": "Machine"
            }
        ],
        "Name": "Wheel"
    },
    {
        "Instances": [],
        "Confidence": 92.61514282226562,
        "Parents": [],
        "Name": "Road"
    },
    {
        "Instances": [],
        "Confidence": 92.37877655029297,
        "Parents": [
            {
                "Name": "Person"
            }
        ],
        "Name": "Sport"
    },
    {
        "Instances": [],
        "Confidence": 92.37877655029297,
        "Parents": [
            {
                "Name": "Person"
            }
        ],
        "Name": "Sports"
    },
    {
        "Instances": [
```

```
                {
                    "BoundingBox": {
                        "Width": 0.12326609343290329,
                        "Top": 0.6332163214683533,
                        "Left": 0.44815489649772644,
                        "Height": 0.058117982000112534
                    },
                    "Confidence": 92.37877655029297
                }
            ],
            "Confidence": 92.37877655029297,
            "Parents": [
                {
                    "Name": "Person"
                },
                {
                    "Name": "Sport"
                }
            ],
            "Name": "Skateboard"
        },
        {
            "Instances": [],
            "Confidence": 90.62931060791016,
            "Parents": [
                {
                    "Name": "Person"
                }
            ],
            "Name": "Pedestrian"
        },
        {
            "Instances": [],
            "Confidence": 88.81334686279297,
            "Parents": [],
            "Name": "Asphalt"
        },
        {
            "Instances": [],
            "Confidence": 88.81334686279297,
            "Parents": [],
            "Name": "Tarmac"
        },
        {
```

```
            "Instances": [],
            "Confidence": 88.23201751708984,
            "Parents": [],
            "Name": "Path"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [],
            "Name": "Urban"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
                }
            ],
            "Name": "Town"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [],
            "Name": "Building"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
                }
            ],
            "Name": "City"
        },
        {
```

```
            "Instances": [],
            "Confidence": 78.37934875488281,
            "Parents": [
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Parking Lot"
        },
        {
            "Instances": [],
            "Confidence": 78.37934875488281,
            "Parents": [
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Parking"
        },
        {
            "Instances": [],
            "Confidence": 74.37590026855469,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
                },
                {
                    "Name": "City"
                }
```

```
            ],
            "Name": "Downtown"
        },
        {
            "Instances": [],
            "Confidence": 69.84622955322266,
            "Parents": [
                {
                    "Name": "Road"
                }
            ],
            "Name": "Intersection"
        },
        {
            "Instances": [],
            "Confidence": 57.68518829345703,
            "Parents": [
                {
                    "Name": "Sports Car"
                },
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Coupe"
        },
        {
            "Instances": [],
            "Confidence": 57.68518829345703,
            "Parents": [
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
```

```
                }
            ],
            "Name": "Sports Car"
        },
        {
            "Instances": [],
            "Confidence": 56.59492111206055,
            "Parents": [
                {
                    "Name": "Path"
                }
            ],
            "Name": "Sidewalk"
        },
        {
            "Instances": [],
            "Confidence": 56.59492111206055,
            "Parents": [
                {
                    "Name": "Path"
                }
            ],
            "Name": "Pavement"
        },
        {
            "Instances": [],
            "Confidence": 55.58770751953125,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
                }
            ],
            "Name": "Neighborhood"
        }
    ],
    "LabelModelVersion": "2.0"
}
```


Weitere Informationen finden Sie unter Erkennen von Labels in einem Bild im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie unter DetectLabels AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

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

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

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

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

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

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

    /**
     * Detects the labels in an image stored in an Amazon S3 bucket using the
Amazon Rekognition service.
     *
     * @param rekClient     the Amazon Rekognition client used to make the
detection request
     * @param bucketName    the name of the Amazon S3 bucket where the image is
stored
     * @param sourceImage   the name of the image file to be analyzed
     */
    public static void detectImageLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

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

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

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

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

- Einzelheiten zur API finden Sie DetectLabelsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }
    val request =
        DetectLabelsRequest {
            image = souImage
            maxLabels = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
```

```
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectLabelsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie DetectLabelsin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DetectModerationLabels`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectModerationLabels` verwendet wird.

Weitere Informationen finden Sie unter Erkennen von unangemessenen Bildern.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   using System;
   using System.Threading.Tasks;
   using Amazon.Rekognition;
   using Amazon.Rekognition.Model;

   /// <summary>
   /// Uses the Amazon Rekognition Service to detect unsafe content in a
   /// JPEG or PNG format image.
   /// </summary>
   public class DetectModerationLabels
   {
       public static async Task Main(string[] args)
       {
           string photo = "input.jpg";
           string bucket = "amzn-s3-demo-bucket";

           var rekognitionClient = new AmazonRekognitionClient();

           var detectModerationLabelsRequest = new
DetectModerationLabelsRequest()
           {
               Image = new Image()
               {
                   S3Object = new S3Object()
                   {
                       Name = photo,
                       Bucket = bucket,
                   },
               },
               MinConfidence = 60F,
           };

           try
           {
               var detectModerationLabelsResponse = await
rekognitionClient.DetectModerationLabelsAsync(detectModerationLabelsRequest);
               Console.WriteLine("Detected labels for " + photo);
               foreach (ModerationLabel label in
detectModerationLabelsResponse.ModerationLabels)
               {
                   Console.WriteLine($"Label: {label.Name}");
                   Console.WriteLine($"Confidence: {label.Confidence}");
                   Console.WriteLine($"Parent: {label.ParentName}");
```

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

- Einzelheiten zur API finden Sie DetectModerationLabelsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um unsichere Inhalte in einem Bild zu erkennen

Der folgende `detect-moderation-labels` Befehl erkennt unsichere Inhalte in dem angegebenen Bild, das in einem Amazon S3 S3-Bucket gespeichert ist.

```
aws rekognition detect-moderation-labels \
    --image "S3Object={Bucket=MyImageS3Bucket,Name=gun.jpg}"
```

Ausgabe:

```
{
    "ModerationModelVersion": "3.0",
    "ModerationLabels": [
        {
            "Confidence": 97.29618072509766,
            "ParentName": "Violence",
            "Name": "Weapon Violence"
        },
        {
            "Confidence": 97.29618072509766,
            "ParentName": "",
            "Name": "Violence"
        }
    ]
```

```
}
```

Weitere Informationen finden Sie unter Erkennen unsicherer Bilder im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie unter Befehlsreferenz DetectModerationLabels.AWS CLI

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

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

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

    public static void main(String[] args) {
        final String usage = """
            Usage:  <bucketName>  <sourceImage>

            Where:
```

```
                bucketName - The name of the S3 bucket where the images are
stored.
                sourceImage - The name of the image (for example, pic1.png).\s
            """;

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

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

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

    /**
     * Detects moderation labels in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient      the Amazon Rekognition client to use for the
detection
     * @param bucketName     the name of the Amazon S3 bucket where the image is
stored
     * @param sourceImage    the name of the image file to be analyzed
     *
     * @throws RekognitionException if there is an error during the image
detection process
     */
    public static void detectModLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();
```

```
            DetectModerationLabelsRequest moderationLabelsRequest =
 DetectModerationLabelsRequest.builder()
                    .image(targetImage)
                    .minConfidence(60F)
                    .build();

            DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                    .detectModerationLabels(moderationLabelsRequest);
            List<ModerationLabel> labels =
 moderationLabelsResponse.moderationLabels();
            System.out.println("Detected labels for image");
            for (ModerationLabel label : labels) {
                System.out.println("Label: " + label.name()
                        + "\n Confidence: " + label.confidence().toString() + "%"
                        + "\n Parent:" + label.parentName());
            }

        } catch (RekognitionException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectModerationLabelsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
```

```
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} %
 Parent: ${label.parentName}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectModerationLabelsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie DetectModerationLabelsin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DetectText`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectText` verwendet wird.

Weitere Informationen finden Sie unter Erkennen von Text in einem Bild.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect text in an image. The
/// example was created using the AWS SDK for .NET version 3.7 and .NET
/// Core 5.0.
/// </summary>
public class DetectText
{
    public static async Task Main()
    {
        string photo = "Dad_photographer.jpg"; // "input.jpg";
        string bucket = "amzn-s3-demo-bucket"; // "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectTextRequest = new DetectTextRequest()
        {
            Image = new Image()
```

```
                {
                    S3Object = new S3Object()
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },
            };

            try
            {
                DetectTextResponse detectTextResponse = await
rekognitionClient.DetectTextAsync(detectTextRequest);
                Console.WriteLine($"Detected lines and words for {photo}");
                detectTextResponse.TextDetections.ForEach(text =>
                {
                    Console.WriteLine($"Detected: {text.DetectedText}");
                    Console.WriteLine($"Confidence: {text.Confidence}");
                    Console.WriteLine($"Id : {text.Id}");
                    Console.WriteLine($"Parent Id: {text.ParentId}");
                    Console.WriteLine($"Type: {text.Type}");
                });
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
```

- Einzelheiten zur API finden Sie DetectTextin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um Text in einem Bild zu erkennen

Der folgende `detect-text` Befehl erkennt Text im angegebenen Bild.

```
aws rekognition detect-text \
```

```
    --image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"ExamplePicture.jpg"}}'
```

Ausgabe:

```
{
    "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
        }
    ]
}
```


- Einzelheiten zur API finden Sie DetectTextin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:   <bucketName> <sourceImage>\n" +
            "\n" +
            "Where:\n" +
            "   bucketName - The name of the S3 bucket where the image is stored
\n" +
            "   sourceImage - The path to the image that contains text (for
 example, pic1.png). \n";

        if (args.length != 2) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

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

    /**
     * Detects text labels in an image stored in an S3 bucket using Amazon
Rekognition.
     *
     * @param rekClient    an instance of the Amazon Rekognition client
     * @param bucketName   the name of the S3 bucket where the image is stored
     * @param sourceImage  the name of the image file in the S3 bucket
     * @throws RekognitionException if an error occurs while calling the Amazon
Rekognition API
     */
    public static void detectTextLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

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

            DetectTextRequest textRequest = DetectTextRequest.builder()
                    .image(souImage)
                    .build();

            DetectTextResponse textResponse = rekClient.detectText(textRequest);
            List<TextDetection> textCollection = textResponse.textDetections();
            System.out.println("Detected lines and words");
            for (TextDetection text : textCollection) {
```

```
                System.out.println("Detected: " + text.detectedText());
                System.out.println("Confidence: " +
 text.confidence().toString());
                System.out.println("Id : " + text.id());
                System.out.println("Parent Id: " + text.parentId());
                System.out.println("Type: " + text.type());
                System.out.println();
            }

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

- Einzelheiten zur API finden Sie DetectTextin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
```

```
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id:  ${text.parentId}")
            println("Type: ${text.type}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectTextin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie DetectTextin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DisassociateFaces`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DisassociateFaces` verwendet wird.

CLI

AWS CLI

```
aws rekognition disassociate-faces --face-ids list-of-face-ids
  --user-id user-id --collection-id collection-name --region region-name
```

- Einzelheiten zur API finden Sie DisassociateFacesin der AWS CLI Befehlsreferenz.

Python

SDK für 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'Disssociating faces from user: {user_id}, {face_ids}')
    try:
        response = client.disassociate_faces(
            CollectionId=collection_id,
            UserId=user_id,
            FaceIds=face_ids
        )
        print(f'- disassociated {len(response["DisassociatedFaces"])} faces')
    except ClientError:
        logger.exception("Failed to disassociate faces from the given user")
        raise
    else:
        print(response)
        return response

def main():
    face_ids = ["faceId1", "faceId2"]
    collection_id = "collection-id"
    user_id = "user-id"
    disassociate_faces(collection_id, user_id, face_ids)
```

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

- Einzelheiten zur API finden Sie DisassociateFacesin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`GetCelebrityInfo`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetCelebrityInfo` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie GetCelebrityInfoin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um Informationen über eine Berühmtheit zu erhalten

Der folgende `get-celebrity-info` Befehl zeigt Informationen über den angegebenen Star an. Der `id` Parameter stammt aus einem früheren Aufruf von`recognize-celebrities`.

```
aws rekognition get-celebrity-info --id nnnnnnn
```

Ausgabe:

```
{
    "Name": "Celeb A",
    "Urls": [
        "www.imdb.com/name/aaaaaaaaa"
```

```
    ]
}
```

Weitere Informationen finden Sie unter Informationen über Prominente im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie GetCelebrityInfoin der AWS CLI Befehlsreferenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterRekognition mit einem SDK verwenden AWS. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`IndexFaces`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `IndexFaces` verwendet wird.

Weitere Informationen finden Sie unter Hinzufügen von Gesichtern zu einer Sammlung.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces in an image
/// that has been uploaded to an Amazon Simple Storage Service (Amazon S3)
/// bucket and then adds the information to a collection.
/// </summary>
public class AddFaces
{
```

```
        public static async Task Main()
        {
            string collectionId = "MyCollection2";
            string bucket = "amzn-s3-demo-bucket";
            string photo = "input.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            var image = new Image
            {
                S3Object = new S3Object
                {
                    Bucket = bucket,
                    Name = photo,
                },
            };

            var indexFacesRequest = new IndexFacesRequest
            {
                Image = image,
                CollectionId = collectionId,
                ExternalImageId = photo,
                DetectionAttributes = new List<string>() { "ALL" },
            };

            IndexFacesResponse indexFacesResponse = await
rekognitionClient.IndexFacesAsync(indexFacesRequest);

            Console.WriteLine($"{photo} added");
            foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
            {
                Console.WriteLine($"Face detected: Faceid is
{faceRecord.Face.FaceId}");
            }
        }
    }
```

- Einzelheiten zur API finden Sie IndexFacesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um Gesichter zu einer Sammlung hinzuzufügen

Mit dem folgenden `index-faces` Befehl werden die in einem Bild gefundenen Gesichter zur angegebenen Sammlung hinzugefügt.

```
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"
```

Ausgabe:

```
{
    "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"
}
```

Weitere Informationen finden Sie unter Gesichter zu einer Sammlung hinzufügen im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie IndexFacesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AddFacesToCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId> <sourceImage> <bucketName>

            Where:
                collectionName - The name of the collection.
                sourceImage - The name of the image (for example, pic1.png).
                bucketName - The name of the S3 bucket.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        String bucketName = args[2];;
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        addToCollection(rekClient, collectionId, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Adds a face from an image to an Amazon Rekognition collection.
     *
     * @param rekClient     the Amazon Rekognition client
     * @param collectionId  the ID of the collection to add the face to
     * @param bucketName    the name of the Amazon S3 bucket containing the image
     * @param sourceImage   the name of the image file to add to the collection
     * @throws RekognitionException if there is an error while interacting with
 the Amazon Rekognition service
     */
    public static void addToCollection(RekognitionClient rekClient, String
 collectionId, String bucketName, String sourceImage) {
```

```
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

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

            IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                    .collectionId(collectionId)
                    .image(targetImage)
                    .maxFaces(1)
                    .qualityFilter(QualityFilter.AUTO)
                    .detectionAttributes(Attribute.DEFAULT)
                    .build();

            IndexFacesResponse facesResponse =
rekClient.indexFaces(facesRequest);
            System.out.println("Results for the image");
            System.out.println("\n Faces indexed:");
            List<FaceRecord> faceRecords = facesResponse.faceRecords();
            for (FaceRecord faceRecord : faceRecords) {
                System.out.println("  Face ID: " + faceRecord.face().faceId());
                System.out.println("  Location:" +
faceRecord.faceDetail().boundingBox().toString());
            }

            List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
            System.out.println("Faces not indexed:");
            for (UnindexedFace unindexedFace : unindexedFaces) {
                System.out.println("  Location:" +
unindexedFace.faceDetail().boundingBox().toString());
                System.out.println("  Reasons:");
                for (Reason reason : unindexedFace.reasons()) {
                    System.out.println("Reason:  " + reason);
                }
            }

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

```
    }
}
```

- Einzelheiten zur API finden Sie IndexFacesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun addToCollection(
    collectionIdVal: String?,
    sourceImage: String,
) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        IndexFacesRequest {
            collectionId = collectionIdVal
            image = souImage
            maxFaces = 1
            qualityFilter = QualityFilter.Auto
            detectionAttributes = listOf(Attribute.Default)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)

        // Display the results.
        println("Results for the image")
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
            println("Face ID: ${faceRecord.face?.faceId}")
```

```
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
            println("Reasons:")

            unindexedFace.reasons?.forEach { reason ->
                println("Reason:  $reason")
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie IndexFacesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie IndexFacesin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListCollections`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListCollections` verwendet wird.

Weitere Informationen finden Sie unter Sammlungen auflisten.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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);
        }
```

```
    }
```

- Einzelheiten zur API finden Sie ListCollectionsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um die verfügbaren Sammlungen aufzulisten

Der folgende `list-collections` Befehl listet die verfügbaren Sammlungen im AWS Konto auf.

```
aws rekognition list-collections
```

Ausgabe:

```
{
    "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"
    ]
}
```

Weitere Informationen finden Sie unter Sammlungen auflisten im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie ListCollectionsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListCollectionsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
```

```
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- Einzelheiten zur API finden Sie ListCollectionsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie ListCollectionsin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListFaces`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListFaces` verwendet wird.

Weitere Informationen finden Sie unter Gesichter in einer Sammlung auflisten.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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));
        }
    }
```

- Einzelheiten zur API finden Sie ListFacesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um die Gesichter in einer Sammlung aufzulisten

Der folgende `list-faces` Befehl listet die Gesichter in der angegebenen Sammlung auf.

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

Ausgabe:

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

Weitere Informationen finden Sie unter Gesichter in einer Sammlung auflisten im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie ListFacesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListFacesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listFaces(request)
        response.faces?.forEach { face ->
            println("Confidence level there is a face: ${face.confidence}")
            println("The face Id value is ${face.faceId}")
        }
```

```
    }
}
```

- Einzelheiten zur API finden Sie ListFacesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie ListFacesin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`RecognizeCelebrities`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RecognizeCelebrities` verwendet wird.

Weitere Informationen finden Sie unter Erkennen von Prominenten in einem Bild.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

/// <summary>
/// Shows how to use Amazon Rekognition to identify celebrities in a photo.
/// </summary>
public class CelebritiesInImage
{
    public static async Task Main(string[] args)
    {
        string photo = "moviestars.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        var recognizeCelebritiesRequest = new RecognizeCelebritiesRequest();

        var img = new Amazon.Rekognition.Model.Image();
        byte[] data = null;
        try
        {
            using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
            data = new byte[fs.Length];
```

```
                fs.Read(data, 0, (int)fs.Length);
            }
            catch (Exception)
            {
                Console.WriteLine($"Failed to load file {photo}");
                return;
            }

            img.Bytes = new MemoryStream(data);
            recognizeCelebritiesRequest.Image = img;

            Console.WriteLine($"Looking for celebrities in image {photo}\n");

            var recognizeCelebritiesResponse = await
rekognitionClient.RecognizeCelebritiesAsync(recognizeCelebritiesRequest);


Console.WriteLine($"{recognizeCelebritiesResponse.CelebrityFaces.Count}
celebrity(s) were recognized.\n");
            recognizeCelebritiesResponse.CelebrityFaces.ForEach(celeb =>
            {
                Console.WriteLine($"Celebrity recognized: {celeb.Name}");
                Console.WriteLine($"Celebrity ID: {celeb.Id}");
                BoundingBox boundingBox = celeb.Face.BoundingBox;
                Console.WriteLine($"position: {boundingBox.Left}
{boundingBox.Top}");
                Console.WriteLine("Further information (if available):");
                celeb.Urls.ForEach(url =>
                {
                    Console.WriteLine(url);
                });
            });


Console.WriteLine($"{recognizeCelebritiesResponse.UnrecognizedFaces.Count}
face(s) were unrecognized.");
        }
    }
```

- Einzelheiten zur API finden Sie RecognizeCelebritiesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um Prominente auf einem Bild zu erkennen

Der folgende `recognize-celebrities` Befehl erkennt Prominente in dem angegebenen Bild, das in einem Amazon S3 S3-Bucket gespeichert ist. :

```
aws rekognition recognize-celebrities \
    --image "S3Object={Bucket=MyImageS3Bucket,Name=moviestars.jpg}"
```

Ausgabe:

```
{
    "UnrecognizedFaces": [
        {
            "BoundingBox": {
                "Width": 0.14416666328907013,
                "Top": 0.07777778059244156,
                "Left": 0.625,
                "Height": 0.2746031880378723
            },
            "Confidence": 99.9990234375,
            "Pose": {
                "Yaw": 10.80408763885498,
                "Roll": -12.761146545410156,
                "Pitch": 10.96889877319336
            },
            "Quality": {
                "Sharpness": 94.1185531616211,
                "Brightness": 79.18367004394531
            },
            "Landmarks": [
                {
                    "Y": 0.18220913410186768,
                    "X": 0.6702951788902283,
                    "Type": "eyeLeft"
                },
                {
                    "Y": 0.16337193548679352,
                    "X": 0.7188183665275574,
                    "Type": "eyeRight"
```

```
                },
                {
                    "Y": 0.20739148557186127,
                    "X": 0.7055801749229431,
                    "Type": "nose"
                },
                {
                    "Y": 0.2889308035373688,
                    "X": 0.687512218952179,
                    "Type": "mouthLeft"
                },
                {
                    "Y": 0.2706988751888275,
                    "X": 0.7250053286552429,
                    "Type": "mouthRight"
                }
            ]
        }
    ],
    "CelebrityFaces": [
        {
            "MatchConfidence": 100.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.14000000059604645,
                    "Top": 0.1190476194024086,
                    "Left": 0.82833331823349,
                    "Height": 0.2666666805744171
                },
                "Confidence": 99.99359130859375,
                "Pose": {
                    "Yaw": -10.509642601013184,
                    "Roll": -14.51749324798584,
                    "Pitch": 13.799399375915527
                },
                "Quality": {
                    "Sharpness": 78.74752044677734,
                    "Brightness": 42.201324462890625
                },
                "Landmarks": [
                    {
                        "Y": 0.2290833294391632,
                        "X": 0.8709492087364197,
                        "Type": "eyeLeft"
```

```
                    },
                    {
                        "Y": 0.20639978349208832,
                        "X": 0.9153988361358643,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.25417643785476685,
                        "X": 0.8907724022865295,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.32729196548461914,
                        "X": 0.8876466155052185,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.3115464746952057,
                        "X": 0.9238573312759399,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Celeb A",
            "Urls": [
                "www.imdb.com/name/aaaaaaaaa"
            ],
            "Id": "1111111"
        },
        {
            "MatchConfidence": 97.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.13333334028720856,
                    "Top": 0.24920634925365448,
                    "Left": 0.4449999928474426,
                    "Height": 0.2539682686328888
                },
                "Confidence": 99.99979400634766,
                "Pose": {
                    "Yaw": 6.557040691375732,
                    "Roll": -7.316643714904785,
                    "Pitch": 9.272967338562012
                },
```

```
                "Quality": {
                    "Sharpness": 83.23492431640625,
                    "Brightness": 78.83267974853516
                },
                "Landmarks": [
                    {
                        "Y": 0.3625510632991791,
                        "X": 0.48898839950561523,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.35366007685661316,
                        "X": 0.5313721299171448,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.3894785940647125,
                        "X": 0.5173314809799194,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.44889405369758606,
                        "X": 0.5020005702972412,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.4408611059188843,
                        "X": 0.5351271629333496,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Celeb B",
            "Urls": [
                "www.imdb.com/name/bbbbbbbbb"
            ],
            "Id": "2222222"
        },
        {
            "MatchConfidence": 100.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.12416666746139526,
                    "Top": 0.2968254089355469,
```

```
                    "Left": 0.2150000035762787,
                    "Height": 0.23650793731212616
                },
                "Confidence": 99.99958801269531,
                "Pose": {
                    "Yaw": 7.801797866821289,
                    "Roll": -8.326810836791992,
                    "Pitch": 7.844768047332764
                },
                "Quality": {
                    "Sharpness": 86.93206024169922,
                    "Brightness": 79.81291198730469
                },
                "Landmarks": [
                    {
                        "Y": 0.4027804136276245,
                        "X": 0.2575301229953766,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.3934555947780609,
                        "X": 0.2956969439983368,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.4309830069541931,
                        "X": 0.2837020754814148,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.48186683654785156,
                        "X": 0.26812544465065,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.47338807582855225,
                        "X": 0.29905644059181213,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Celeb C",
            "Urls": [
                "www.imdb.com/name/ccccccccc"
```

```
            ],
            "Id": "3333333"
        },
        {
            "MatchConfidence": 97.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.11916666477918625,
                    "Top": 0.3698412775993347,
                    "Left": 0.008333333767950535,
                    "Height": 0.22698412835597992
                },
                "Confidence": 99.99999237060547,
                "Pose": {
                    "Yaw": 16.38478660583496,
                    "Roll": -1.0260354280471802,
                    "Pitch": 5.975185394287109
                },
                "Quality": {
                    "Sharpness": 83.23492431640625,
                    "Brightness": 61.408443450927734
                },
                "Landmarks": [
                    {
                        "Y": 0.4632347822189331,
                        "X": 0.049406956881284714,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.46388113498687744,
                        "X": 0.08722897619009018,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.5020678639411926,
                        "X": 0.0758260041475296,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.544157862663269,
                        "X": 0.054029736667871475,
                        "Type": "mouthLeft"
                    },
                    {
```

```
                        "Y": 0.5463630557060242,
                        "X": 0.08464983850717545,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Celeb D",
            "Urls": [
                "www.imdb.com/name/ddddddddd"
            ],
            "Id": "4444444"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erkennen von Prominenten in einem Bild im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie RecognizeCelebritiesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

import software.amazon.awssdk.services.rekognition.model.*;

/**
 * Before running this Java V2 code example, set up your development
```

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

                Where:
                   bucketName - The name of the S3 bucket where the images are
 stored.
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

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

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

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

    /**
     * Recognizes all celebrities in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient    the Amazon Rekognition client used to perform the
 celebrity recognition operation
     * @param bucketName   the name of the Amazon S3 bucket where the source
 image is stored
```

```
     * @param sourceImage  the name of the source image file stored in the Amazon
 S3 bucket
     */
    public static void recognizeAllCelebrities(RekognitionClient rekClient,
 String bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
                .build();

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

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

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

                System.out.println("Further information (if available):");
                for (String url : celebrity.urls()) {
                    System.out.println(url);
                }
                System.out.println();
            }
            System.out.println(result.unrecognizedFaces().size() + " face(s) were
 unrecognized.");

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

- Einzelheiten zur API finden Sie RecognizeCelebritiesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
        println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
    }
}
```

- Einzelheiten zur API finden Sie RecognizeCelebritiesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie RecognizeCelebritiesin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`SearchFaces`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SearchFaces` verwendet wird.

Weitere Informationen finden Sie unter Nach einem Gesicht suchen (Gesichts-ID).

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to find faces in an image that
    /// match the face Id provided in the method request.
    /// </summary>
    public class SearchFacesMatchingId
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection";
            string faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";

            var rekognitionClient = new AmazonRekognitionClient();

            // Search collection for faces matching the face id.
            var searchFacesRequest = new SearchFacesRequest
            {
                CollectionId = collectionId,
                FaceId = faceId,
                FaceMatchThreshold = 70F,
                MaxFaces = 2,
            };

            SearchFacesResponse searchFacesResponse = await
rekognitionClient.SearchFacesAsync(searchFacesRequest);

            Console.WriteLine("Face matching faceId " + faceId);
```

```
            Console.WriteLine("Matche(s): ");
            searchFacesResponse.FaceMatches.ForEach(face =>
            {
                Console.WriteLine($"FaceId: {face.Face.FaceId} Similarity:
 {face.Similarity}");
            });
        }
    }
```

- Einzelheiten zur API finden Sie SearchFacesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um in einer Sammlung nach Gesichtern zu suchen, die einer Gesichts-ID entsprechen.

Mit dem folgenden `search-faces` Befehl wird in einer Sammlung nach Gesichtern gesucht, die der angegebenen Gesichts-ID entsprechen.

```
aws rekognition search-faces \
    --face-id 8d3cfc70-4ba8-4b36-9644-90fba29c2dac \
    --collection-id MyCollection
```

Ausgabe:

```
{
    "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
        }
    ]
}
```

Weitere Informationen finden Sie unter Suchen nach einem Gesicht anhand seiner Gesichts-ID im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie SearchFacesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

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

                Usage:    <collectionId> <sourceImage>

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

                """;

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

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
```

```
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }

    public static void searchFaceInCollection(RekognitionClient rekClient, String
 collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(new
 File(sourceImage));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            SearchFacesByImageRequest facesByImageRequest =
 SearchFacesByImageRequest.builder()
                    .image(souImage)
                    .maxFaces(10)
                    .faceMatchThreshold(70F)
                    .collectionId(collectionId)
                    .build();

            SearchFacesByImageResponse imageResponse =
 rekClient.searchFacesByImage(facesByImageRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " +
 face.similarity());
                System.out.println();
            }

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

- Einzelheiten zur API finden Sie SearchFacesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie SearchFacesin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`SearchFacesByImage`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SearchFacesByImage` verwendet wird.

Weitere Informationen finden Sie unter Nach einem Gesicht suchen (Bild).

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to search for images matching those
/// in a collection.
/// </summary>
public class SearchFacesMatchingImage
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
```

```
            string bucket = "amzn-s3-demo-bucket";
            string photo = "input.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            // Get an image object from S3 bucket.
            var image = new Image()
            {
                S3Object = new S3Object()
                {
                    Bucket = bucket,
                    Name = photo,
                },
            };

            var searchFacesByImageRequest = new SearchFacesByImageRequest()
            {
                CollectionId = collectionId,
                Image = image,
                FaceMatchThreshold = 70F,
                MaxFaces = 2,
            };

            SearchFacesByImageResponse searchFacesByImageResponse = await
rekognitionClient.SearchFacesByImageAsync(searchFacesByImageRequest);

            Console.WriteLine("Faces matching largest face in image from " +
photo);
            searchFacesByImageResponse.FaceMatches.ForEach(face =>
            {
                Console.WriteLine($"FaceId: {face.Face.FaceId}, Similarity:
{face.Similarity}");
            });
        }
    }
```

- Einzelheiten zur API finden Sie SearchFacesByImagein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um in einer Sammlung nach Gesichtern zu suchen, die dem größten Gesicht in einem Bild entsprechen.

Mit dem folgenden `search-faces-by-image` Befehl wird in einer Sammlung nach Gesichtern gesucht, die dem größten Gesicht im angegebenen Bild entsprechen. :

```
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"
}
```


Weitere Informationen finden Sie unter Mit einem Bild nach einem Gesicht suchen im Amazon Rekognition Developer Guide.

- Einzelheiten zur API finden Sie SearchFacesByImagein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

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

                Usage:    <collectionId> <sourceImage>

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

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

```
        }

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

        System.out.println("Searching for a face in a collections");
        searchFacebyId(rekClient, collectionId, faceId);
        rekClient.close();
    }

    public static void searchFacebyId(RekognitionClient rekClient, String
 collectionId, String faceId) {
        try {
            SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceId(faceId)
                    .faceMatchThreshold(70F)
                    .maxFaces(2)
                    .build();

            SearchFacesResponse imageResponse =
 rekClient.searchFaces(searchFacesRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " +
 face.similarity());
                System.out.println();
            }

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

- Einzelheiten zur API finden Sie SearchFacesByImagein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie SearchFacesByImagein AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Rekognition mit einem SDK verwenden AWS Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Szenarien für die Verwendung von Amazon Rekognition AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon Rekognition mit implementieren. AWS SDKs Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen in Amazon Rekognition oder in Kombination mit anderen aufrufen. AWS-Services Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen Sie eine Amazon Rekognition Rekognition-Sammlung und suchen Sie darin mithilfe eines SDK nach Gesichtern AWS
- Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können

- Ermitteln Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines SDK AWS
- Erkennen und Anzeigen von Elementen in Bildern mit Amazon Rekognition mithilfe eines SDK AWS
- Erkennen Sie Gesichter in einem Bild mithilfe eines AWS SDK
- Informationen in Videos mithilfe von Amazon Rekognition und dem SDK erkennen AWS
- Objekte in Bildern mit Amazon Rekognition mithilfe eines SDK erkennen AWS
- Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines SDK AWS
- Speichern Sie EXIF und andere Bildinformationen mit einem SDK AWS

Erstellen Sie eine Amazon Rekognition Rekognition-Sammlung und suchen Sie darin mithilfe eines SDK nach Gesichtern AWS

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Amazon-Rekognition-Sammlung.
- Fügen Sie der Sammlung Bilder hinzu und erkennen Sie Gesichter darin.
- Durchsuchen Sie die Sammlung nach Gesichtern, die einem Referenzbild entsprechen.
- Löschen einer Sammlung.

Weitere Informationen finden Sie unter Gesichter in einer Sammlung suchen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Klassen, die Amazon-Rekognition-Funktionen wrappen (verpacken).

```
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
```

Verwenden Sie die Wrapper-Klassen, um eine Sammlung von Gesichtern aus einer Reihe von Bildern zu erstellen und dann nach Gesichtern in der Sammlung zu suchen.

```
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)
```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterRekognition mit einem SDK verwenden AWS. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können

Die folgenden Codebeispiele zeigen, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels verwalten können.

.NET

SDK für .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterRekognition mit einem SDK verwenden AWS. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Ermitteln Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines SDK AWS

Das folgende Codebeispiel zeigt, wie Sie eine App erstellen, die Amazon Rekognition verwendet, um persönliche Schutzausrüstung (PSA) in Bildern zu erkennen.

Java

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion erstellt wird, die Bilder mit persönlicher Schutzausrüstung erkennt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterRekognition mit einem SDK verwenden AWS. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erkennen und Anzeigen von Elementen in Bildern mit Amazon Rekognition mithilfe eines SDK AWS

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erkennen von Elementen in Bildern mithilfe von Amazon Rekognition.
- Zeigen Sie Bilder an und zeichnen Sie Begrenzungsrahmen um die erkannten Elemente.

Weitere Informationen finden Sie unter Anzeigen von Begrenzungsrahmen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Klassen, um Amazon-Rekognition-Funktionen zu umschließen.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests

from rekognition_objects import (
    RekognitionFace,
    RekognitionCelebrity,
    RekognitionLabel,
    RekognitionModerationLabel,
    RekognitionText,
    show_bounding_boxes,
    show_polygons,
)

logger = logging.getLogger(__name__)


class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.
```

```
        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    @classmethod
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened
and its
                                bytes are read.
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
                           file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from
the
                 file.
        """
        with open(image_file_name, "rb") as img_file:
            image = {"Bytes": img_file.read()}
        name = image_file_name if image_name is None else image_name
        return cls(image, name, rekognition_client)


    @classmethod
    def from_bucket(cls, s3_object, rekognition_client):
        """
        Creates a RekognitionImage object from an Amazon S3 object.

        :param s3_object: An Amazon S3 object that identifies the image. The
image
                          is not retrieved until needed for a later call.
        :param rekognition_client: A Boto3 Rekognition client.
        :return: The RekognitionImage object, initialized with Amazon S3 object
data.
        """
```

```
        image = {"S3Object": {"Bucket": s3_object.bucket_name, "Name":
s3_object.key}}
        return cls(image, s3_object.key, rekognition_client)


    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
            raise
        else:
            return faces


    def detect_labels(self, max_labels):
        """
        Detects labels in the image. Labels are objects and people.

        :param max_labels: The maximum number of labels to return.
        :return: The list of labels detected in the image.
        """
        try:
            response = self.rekognition_client.detect_labels(
                Image=self.image, MaxLabels=max_labels
            )
            labels = [RekognitionLabel(label) for label in response["Labels"]]
            logger.info("Found %s labels in %s.", len(labels), self.image_name)
        except ClientError:
            logger.info("Couldn't detect labels in %s.", self.image_name)
            raise
        else:
            return labels
```

```
    def recognize_celebrities(self):
        """
        Detects celebrities in the image.

        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
        try:
            response =
self.rekognition_client.recognize_celebrities(Image=self.image)
            celebrities = [
                RekognitionCelebrity(celeb) for celeb in
response["CelebrityFaces"]
            ]
            other_faces = [
                RekognitionFace(face) for face in response["UnrecognizedFaces"]
            ]
            logger.info(
                "Found %s celebrities and %s other faces in %s.",
                len(celebrities),
                len(other_faces),
                self.image_name,
            )
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.",
self.image_name)
            raise
        else:
            return celebrities, other_faces



    def compare_faces(self, target_image, similarity):
        """
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value
greater
                           than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                 reference image. The second element is the list of faces that
have
```

```
                a similarity value below the specified threshold.
        """
        try:
            response = self.rekognition_client.compare_faces(
                SourceImage=self.image,
                TargetImage=target_image.image,
                SimilarityThreshold=similarity,
            )
            matches = [
                RekognitionFace(match["Face"]) for match in
response["FaceMatches"]
            ]
            unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
            logger.info(
                "Found %s matched faces and %s unmatched faces.",
                len(matches),
                len(unmatches),
            )
        except ClientError:
            logger.exception(
                "Couldn't match faces from %s to %s.",
                self.image_name,
                target_image.image_name,
            )
            raise
        else:
            return matches, unmatches


    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify
content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image
            )
            labels = [
                RekognitionModerationLabel(label)
```

```
                for label in response["ModerationLabels"]
            ]
            logger.info(
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels


    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in
 response["TextDetections"]]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

Erstellen Sie Hilfsfunktionen zum Zeichnen von Begrenzungsrahmen und Polygonen.

```
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()
```

Erstellen Sie Klassen, um von Amazon Rekognition zurückgegebene Objekte zu analysieren.

```
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
```

Verwenden Sie die Wrapper-Klassen, um Elemente in Bildern zu erkennen und ihre Begrenzungsrahmen anzuzeigen. Die in diesem Beispiel verwendeten Bilder finden Sie GitHub zusammen mit Anweisungen und weiterem Code unter.

```
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)
```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterRekognition mit einem SDK verwenden AWS. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erkennen Sie Gesichter in einem Bild mithilfe eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Speichern Sie ein Bild in einem Amazon-S3-Bucket.
- Verwenden Sie Amazon Rekognition, um Gesichtsdetails wie Altersgruppe, Geschlecht und Emotionen (z B. Lächeln) zu erkennen.
- Zeigen Sie diese Details an.

Rust

SDK für Rust

Speichern Sie das Bild in einem Amazon-S3-Bucket mit einem uploads-Präfix, verwenden Sie Amazon Rekognition, um Gesichtsdetails wie Altersgruppe, Geschlecht und Emotionen (Lächeln usw.) zu erkennen, und zeigen Sie diese Details an.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterRekognition mit einem SDK verwenden AWS. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Informationen in Videos mithilfe von Amazon Rekognition und dem SDK erkennen AWS

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Starten Sie Amazon-Rekognition-Aufträge, um Elemente wie Personen, Objekte und Text in Videos zu erkennen.
- Überprüfen Sie den Auftragsstatus, bis die Aufträge abgeschlossen sind.
- Gibt die Liste der von jedem Auftrag erkannten Elemente aus.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abrufen von Informationen aus einem Video, das sich in einem Amazon-S3-Bucket befindet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;

/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-
sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class VideoCelebrityDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

       if (args.length != 4) {
           System.out.println(usage);
           System.exit(1);
       }

       String bucket = args[0];
       String video = args[1];
       String topicArn = args[2];
       String roleArn = args[3];
       Region region = Region.US_EAST_1;
       RekognitionClient rekClient = RekognitionClient.builder()
               .region(region)
               .build();

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

       startCelebrityDetection(rekClient, channel, bucket, video);
       getCelebrityDetectionResults(rekClient);
       System.out.println("This example is done!");
       rekClient.close();
   }

   public static void startCelebrityDetection(RekognitionClient rekClient,
           NotificationChannel channel,
           String bucket,
           String video) {
       try {
           S3Object s3Obj = S3Object.builder()
                   .bucket(bucket)
```

```
                    .name(video)
                    .build();

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

            StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
                    .jobTag("Celebrities")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient
                    .startCelebrityRecognition(recognitionRequest);
            startJobId = startCelebrityRecognitionResult.jobId();

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

    public static void getCelebrityDetectionResults(RekognitionClient rekClient)
{
        try {
            String paginationToken = null;
            GetCelebrityRecognitionResponse recognitionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (recognitionResponse != null)
                    paginationToken = recognitionResponse.nextToken();

                GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                        .maxResults(10)
```

```
                        .build();

                // Wait until the job succeeds
                while (!finished) {
                    recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
                    status = recognitionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
recognitionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<CelebrityRecognition> celebs =
recognitionResponse.celebrities();
                for (CelebrityRecognition celeb : celebs) {
                    long seconds = celeb.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    CelebrityDetail details = celeb.celebrity();
                    System.out.println("Name: " + details.name());
                    System.out.println("Id: " + details.id());
                    System.out.println();
                }

            } while (recognitionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
```

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

Erkennen Sie Labels in einem Video mithilfe einer Labelerkennung.

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

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

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

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

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

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

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
```

```
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

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

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

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

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

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

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

                GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
```

```
                        .jobId(startJobId)
                        .maxResults(10)
                        .build();

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

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

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

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

        } catch (RekognitionException | InterruptedException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
                for (Message message : messages) {
                    String notification = message.body();

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

```
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
                        System.out.println("Status : " +
operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            getResultsLabels(rekClient);
                        else
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {
```

```
        int maxResults = 10;
        String paginationToken = null;
        GetLabelDetectionResponse labelDetectionResult = null;

        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData =
labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

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

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
```

```
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
 instance.confidence().toString());
                            System.out.println("        Bounding box: " +
 instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
 ":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult != null &&
 labelDetectionResult.nextToken() != null);

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

Erkennen von Gesichtern in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
```

```
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

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

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of the video (for example, people.mp4).\s
                   queueUrl- The URL of a SQS queue.\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
```

```
                    roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

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

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

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

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

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();
```

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

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

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

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

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

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

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

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

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

        } catch (RekognitionException | InterruptedException e) {
```

```
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
                for (Message message : messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
                        System.out.println("Status : " +
operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            getResultsLabels(rekClient);
                        else
```

```
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

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

        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();

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

```
                VideoMetadata videoMetaData =
labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

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

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
instance.confidence().toString());
                            System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
```

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

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

Erkennen von unangemessenen oder anstößigen Inhalten in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import
 software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

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

```
public class VideoDetectInappropriate {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

       if (args.length != 4) {
           System.out.println(usage);
           System.exit(1);
       }

       String bucket = args[0];
       String video = args[1];
       String topicArn = args[2];
       String roleArn = args[3];
       Region region = Region.US_EAST_1;
       RekognitionClient rekClient = RekognitionClient.builder()
               .region(region)
               .build();

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

       startModerationDetection(rekClient, channel, bucket, video);
       getModResults(rekClient);
       System.out.println("This example is done!");
       rekClient.close();
   }
```

```
    public static void startModerationDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {

        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

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

            StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
                    .jobTag("Moderation")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartContentModerationResponse startModDetectionResult = rekClient
                    .startContentModeration(modDetectionRequest);
            startJobId = startModDetectionResult.jobId();

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

    public static void getModResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetContentModerationResponse modDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (modDetectionResponse != null)
                    paginationToken = modDetectionResponse.nextToken();
```

```
                GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    modDetectionResponse =
rekClient.getContentModeration(modRequest);
                    status = modDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
modDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
                for (ContentModerationDetection mod : mods) {
                    long seconds = mod.timestamp() / 1000;
                    System.out.print("Mod label: " + seconds + " ");
                    System.out.println(mod.moderationLabel().toString());
                    System.out.println();
                }
```

```
            } while (modDetectionResponse != null &&
 modDetectionResponse.nextToken() != null);

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

Erkennen Sie technische Signal-Segmente und Einstellungserkennungssegmente in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
import java.util.List;

/**
```

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

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

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

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

    public static void startSegmentDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

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

            StartShotDetectionFilter cueDetectionFilter =
StartShotDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
StartTechnicalCueDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartSegmentDetectionFilters filters =
StartSegmentDetectionFilters.builder()
                    .shotFilter(cueDetectionFilter)
                    .technicalCueFilter(technicalCueDetectionFilter)
                    .build();
```

```
            StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .segmentTypes(SegmentType.TECHNICAL_CUE, SegmentType.SHOT)
                    .video(vidOb)
                    .filters(filters)
                    .build();

            StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
            startJobId = segDetectionResponse.jobId();

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

    public static void getSegmentResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetSegmentDetectionResponse segDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (segDetectionResponse != null)
                    paginationToken = segDetectionResponse.nextToken();

                GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
                    status = segDetectionResponse.jobStatusAsString();
```

```
                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }
                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
                for (VideoMetadata metaData : videoMetaData) {
                    System.out.println("Format: " + metaData.format());
                    System.out.println("Codec: " + metaData.codec());
                    System.out.println("Duration: " + metaData.durationMillis());
                    System.out.println("FrameRate: " + metaData.frameRate());
                    System.out.println("Job");
                }

                List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
                for (SegmentDetection detectedSegment : detectedSegments) {
                    String type = detectedSegment.type().toString();
                    if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
                        System.out.println("Technical Cue");
                        TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
                        System.out.println("\tType: " + segmentCue.type());
                        System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
                    }

                    if (type.contains(SegmentType.SHOT.toString())) {
                        System.out.println("Shot");
                        ShotSegment segmentShot = detectedSegment.shotSegment();
                        System.out.println("\tIndex " + segmentShot.index());
                        System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
                    }

                    long seconds = detectedSegment.durationMillis();
```

```
                    System.out.println("\tDuration : " + seconds + "
 milliseconds");
                    System.out.println("\tStart time code: " +
 detectedSegment.startTimecodeSMPTE());
                    System.out.println("\tEnd time code: " +
 detectedSegment.endTimecodeSMPTE());
                    System.out.println("\tDuration time code: " +
 detectedSegment.durationSMPTE());
                    System.out.println();
                }

            } while (segDetectionResponse != null &&
 segDetectionResponse.nextToken() != null);

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

Erkennen Sie Text in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetectText {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

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

```
        startTextLabels(rekClient, channel, bucket, video);
        getTextResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startTextLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

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

            StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

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

    public static void getTextResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetTextDetectionResponse textDetectionResponse = null;
            boolean finished = false;
            String status;
```

```
            int yy = 0;

            do {
                if (textDetectionResponse != null)
                    paginationToken = textDetectionResponse.nextToken();

                GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                    status = textDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
textDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<TextDetectionResult> labels =
textDetectionResponse.textDetections();
                for (TextDetectionResult detectedText : labels) {
```

```
                    System.out.println("Confidence: " +
 detectedText.textDetection().confidence().toString());
                    System.out.println("Id : " +
 detectedText.textDetection().id());
                    System.out.println("Parent Id: " +
 detectedText.textDetection().parentId());
                    System.out.println("Type: " +
 detectedText.textDetection().type());
                    System.out.println("Text: " +
 detectedText.textDetection().detectedText());
                    System.out.println();
                }

            } while (textDetectionResponse != null &&
 textDetectionResponse.nextToken() != null);

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

Erkennen Sie Personen in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;
```

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

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
```

```
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startPersonLabels(rekClient, channel, bucket, video);
        getPersonDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startPersonLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

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

            StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vidOb)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

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

    public static void getPersonDetectionResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
```

```
            GetPersonTrackingResponse personTrackingResult = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (personTrackingResult != null)
                    paginationToken = personTrackingResult.nextToken();

                GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds
                while (!finished) {

                    personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                    status = personTrackingResult.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
personTrackingResult.videoMetadata();

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

```
                System.out.println("Job");

                List<PersonDetection> detectedPersons =
 personTrackingResult.persons();
                for (PersonDetection detectedPerson : detectedPersons) {
                    long seconds = detectedPerson.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    System.out.println("Person Identifier: " +
 detectedPerson.person().index());
                    System.out.println();
                }

            } while (personTrackingResult != null &&
 personTrackingResult.nextToken() != null);

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

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection
  - StartTextDetection

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erkennen von Gesichtern in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
suspend fun startFaceDetection(
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vidOb
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val startLabelDetectionResult = rekClient.startFaceDetection(request)
        startJobId = startLabelDetectionResult.jobId.toString()
    }
}

suspend fun getFaceResults() {
    var finished = false
```

```
    var status: String
    var yy = 0
    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        var response: GetFaceDetectionResponse? = null

        val recognitionRequest =
            GetFaceDetectionRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            response = rekClient.getFaceDetection(recognitionRequest)
            status = response.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = response?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        // Show face information.
        response?.faces?.forEach { face ->
            println("Age: ${face.face?.ageRange}")
            println("Face: ${face.face?.beard}")
            println("Eye glasses: ${face?.face?.eyeglasses}")
            println("Mustache: ${face.face?.mustache}")
            println("Smile: ${face.face?.smile}")
        }
    }
}
```

Erkennen von unangemessenen oder anstößigen Inhalten in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
suspend fun startModerationDetection(
    channel: NotificationChannel?,
    bucketVal: String?,
    videoVal: String?,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }
    val request =
        StartContentModerationRequest {
            jobTag = "Moderation"
            notificationChannel = channel
            video = vidOb
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val startModDetectionResult = rekClient.startContentModeration(request)
        startJobId = startModDetectionResult.jobId.toString()
    }
}

suspend fun getModResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var modDetectionResponse: GetContentModerationResponse? = null

        val modRequest =
            GetContentModerationRequest {
                jobId = startJobId
                maxResults = 10
            }
```

```
        // Wait until the job succeeds.
        while (!finished) {
            modDetectionResponse = rekClient.getContentModeration(modRequest)
            status = modDetectionResponse.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = modDetectionResponse?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        modDetectionResponse?.moderationLabels?.forEach { mod ->
            val seconds: Long = mod.timestamp / 1000
            print("Mod label: $seconds ")
            println(mod.moderationLabel)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration

- StartLabelDetection
- StartPersonTracking
- StartSegmentDetection
- StartTextDetection

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterRekognition mit einem SDK verwenden AWS. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Objekte in Bildern mit Amazon Rekognition mithilfe eines SDK erkennen AWS

Die folgenden Code-Beispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

.NET

SDK für .NET

Zeigt, wie Sie die Amazon-Rekognition-.NET-API verwenden, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Bucket von Amazon Simple Storage Service (Amazon S3) befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Amazon Rekognition zusammen mit dem verwendet wird, um eine App AWS SDK für JavaScript zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3) -Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

So funktioniert es:

- Erstellen Sie mit Amazon Cognito einen nicht authentifizierten Benutzer.
- Analysieren Sie mit Amazon Rekognition Bilder für Objekte.
- Verifizieren Sie eine E-Mail-Adresse für Amazon SES.
- Senden Sie eine E-Mail-Benachrichtigung mit Amazon SES.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK für Kotlin

Zeigt, wie man die Amazon-Rekognition-Kotlin-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK für Python (Boto3)

Zeigt Ihnen, wie Sie mit AWS SDK für Python (Boto3) dem eine Webanwendung erstellen, mit der Sie Folgendes tun können:

- Laden Sie Fotos in einen Bucket von Amazon Simple Storage Service (Amazon S3) hoch.
- Verwenden Sie Amazon Rekognition, um die Fotos zu analysieren und zu markieren.
- Verwenden Sie Amazon Simple Email Service (Amazon SES), um E-Mail-Berichte von Bildanalysen zu senden.

Dieses Beispiel enthält zwei Hauptkomponenten: eine eingeschriebene Webseite JavaScript , die mit React erstellt wurde, und einen in Python geschriebenen REST-Dienst, der mit Flask-RESTful erstellt wurde.

Sie können die React-Webseite verwenden, um Folgendes auszuführen:

- Zeigen Sie eine Liste der Bilder an, die in Ihrem S3-Bucket gespeichert sind.
- Laden Sie Bilder von Ihrem Computer in Ihren S3-Bucket hoch.
- Zeigen Sie Bilder und Markierungen an, die Elemente identifizieren, welche im Bild erkannt werden.
- Rufen Sie einen Bericht über alle Bilder in Ihrem S3-Bucket ab und senden Sie eine E-Mail mit dem Bericht.

Die Webseite ruft den REST-Service auf. Der Service sendet Anforderungen an AWS , um die folgenden Aktionen durchzuführen:

- Die Liste der Bilder abrufen und in Ihrem S3-Bucket filtern.
- Fotos in Ihren S3-Bucket hochladen.
- Verwenden Sie Amazon Rekognition, um einzelne Fotos zu analysieren und eine Liste von Markierungen zu erhalten, die die auf dem Foto erkannten Elemente identifizieren.
- Analysieren Sie alle Fotos in Ihrem S3-Bucket und verwenden Sie Amazon SES, um einen Bericht per E-Mail zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterRekognition mit einem SDK verwenden AWS. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines SDK AWS

Die folgenden Code-Beispiele zeigen, wie man Personen und Objekte in einem Video mit Amazon Rekognition erkennt.

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Gesichter und Objekte in Videos erkennt, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK für Python (Boto3)

Verwenden Sie Amazon Rekognition, um Gesichter, Objekte und Personen in Videos zu erkennen, indem Sie asynchrone Erkennungsaufträge starten. In diesem Beispiel wird Amazon Rekognition auch so konfiguriert, dass es ein Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigt, wenn Aufträge abgeschlossen sind, und eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange bei dem Thema abonniert. Wenn die Warteschlange eine Meldung über einen Job erhält, wird der Job abgerufen und die Ergebnisse werden ausgegeben.

Dieses Beispiel lässt sich am besten auf ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition

- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterRekognition mit einem SDK verwenden AWS. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Speichern Sie EXIF und andere Bildinformationen mit einem SDK AWS

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab.
- Laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch.
- Verwenden Sie Amazon Rekognition, um die drei wichtigsten Attribute (Labels) in der Datei zu identifizieren.
- Fügen Sie die EXIF- und Label-Informationen einer Amazon-DynamoDB-Tabelle in der Region hinzu.

Rust

SDK für Rust

Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab, laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch und identifizieren Sie mit Amazon Rekognition die drei wichtigsten Attribute (Labels in Amazon Rekognition) in der Datei. Fügen Sie die EXIF- und Labelinformationen dann einer Amazon-DynamoDB-Tabelle in der Region hinzu.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterRekognition mit einem SDK verwenden AWS. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

API-Referenz

Die Amazon-Rekognition-API-Referenz ist jetzt unter Amazon-Rekognition-API-Referenz verfügbar.

Sicherheit von Amazon Rekognition

Cloud-Sicherheit hat AWS höchste Priorität. Als AWS Kunde profitieren Sie von einem Rechenzentrum und einer Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

In den folgenden Themen wird beschrieben, wie Ihre Amazon-Rekognition-Ressourcen geschützt werden.

Themen

- Identitäts- und Zugriffsverwaltung für Amazon Rekognition
- Datenschutz in Amazon Rekognition
- Verwenden von Amazon Rekognition mit Amazon-VPC-Endpunkten
- Compliance-Validierung für Amazon Rekognition
- Ausfallsicherheit bei Amazon Rekognition
- Konfigurations- und Schwachstellenanalyse in Amazon Rekognition
- Serviceübergreifende Confused-Deputy-Prävention
- Infrastruktursicherheit in Amazon Rekognition

Identitäts- und Zugriffsverwaltung für Amazon Rekognition

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf AWS Ressourcen sicher zu kontrollieren. IAM-Administratoren steuern, wer authentifiziert (angemeldet) und autorisiert (im Besitz von Berechtigungen) ist, Amazon-Rekognition-Ressourcen zu nutzen. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

- Zielgruppe
- Authentifizierung mit Identitäten
- Verwalten des Zugriffs mit Richtlinien
- So funktioniert Amazon Rekognition mit IAM
- AWS verwaltete Richtlinien für Amazon Rekognition
- Bewährte Methoden für Richtlinien

- Verwenden der Amazon-Rekognition-Konsole
- Beispiele für die Verwendung identitätsbasierter Richtlinien von Amazon Rekognition
- Beispiele für ressourcenbasierte Richtlinien von Amazon Rekognition
- Fehlerbehebung für Amazon-Rekognition-Identität und -Zugriff

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, die Sie in Amazon Rekognition ausführen.

Service-Benutzer – Wenn Sie den Amazon–Rekognition-Service zur Ausführung von Aufgaben verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen bereit, die Sie benötigen. Wenn Sie zur Ausführung von Aufgaben weitere Amazon-Rekognition-Features verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Funktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anfordern müssen. Wenn Sie auf ein Feature in Amazon Rekognition nicht zugreifen können, siehe Fehlerbehebung für Amazon-Rekognition-Identität und -Zugriff.

Service-Administrator – Wenn Sie in Ihrem Unternehmen für die Amazon-Rekognition-Ressourcen zuständig sind, haben Sie wahrscheinlich vollen Zugriff auf Amazon Rekognition. Ihre Aufgabe besteht darin, zu bestimmen, auf welche Amazon–Rekognition-Features und -Ressourcen Ihre Service-Benutzer zugreifen sollen. Anschließend müssen Sie Anforderungen an Ihren IAM-Administrator senden, um die Berechtigungen der Servicebenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen dazu, wie Ihr Unternehmen IAM mit Amazon Rekognition verwenden kann, finden Sie unter So funktioniert Amazon Rekognition mit IAM.

IAM-Administrator – Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf Amazon Rekognition verfassen können. Beispiele für identitätsbasierte Amazon-Rekognition-Richtlinien, die Sie in IAM verwenden können, finden Sie unter Beispiele für die Verwendung identitätsbasierter Richtlinien von Amazon Rekognition.

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM-Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAM Identity Center) -Benutzer, die Single Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie über den Verbund darauf zugreifen AWS , übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, um Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode für die Selbstsignierung von Anforderungen finden Sie unter AWS Signature Version 4 für API-Anforderungen im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen bereitstellen. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center - Benutzerhandbuch und AWS Multi-Faktor-Authentifizierung (MFA) in IAM im IAM-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer

gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise eine Gruppe benennen IAMAdminsund dieser Gruppe Berechtigungen zur Verwaltung von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität innerhalb von Ihnen AWS-Konto , die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, jedoch nicht mit einer bestimmten Person verknüpft. Um vorübergehend eine IAM-Rolle in der zu übernehmen AWS Management Console, können Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI oder AWS API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Methoden für die Übernahme einer Rolle im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden

zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

- Serviceübergreifender Zugriff — Einige AWS-Services verwenden Funktionen in anderen AWS-Services. Wenn Sie beispielsweise einen Service aufrufen, ist es üblich, dass dieser Service Anwendungen in Amazon ausführt EC2 oder Objekte in Amazon S3 speichert. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward Access Sessions (FAS) — Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anfrage, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.
  - Servicerolle – Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
  - Dienstbezogene Rolle — Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.
- Auf Amazon ausgeführte Anwendungen EC2 — Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2 Instance ausgeführt werden und AWS API-Anfragen stellen AWS CLI . Dies ist dem Speichern von Zugriffsschlüsseln innerhalb der EC2 Instance vorzuziehen. Um einer EC2 Instanz eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instanzprofil, das an die Instanz angehängt ist. Ein Instanzprofil enthält die Rolle und ermöglicht Programmen, die auf der EC2 Instanz ausgeführt werden, temporäre Anmeldeinformationen abzurufen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Verwenden einer IAM-Rolle, um Berechtigungen für Anwendungen zu gewähren, die auf EC2 Amazon-Instances ausgeführt werden.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Die Berechtigungen in den Richtlinien legen fest, ob eine Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der AWS Management Console AWS CLI, der oder der AWS API abrufen.

Verwenden identitätsbasierter Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können AWS-Konto.

Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer Inline-Richtlinie wählen, finden Sie unter Auswählen zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Verwenden ressourcenbasierter Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Dienste, die Unterstützung ACLs bieten. AWS WAF Weitere Informationen finden Sie unter Übersicht über ACLs die Zugriffskontrollliste (ACL) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der

identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.

- Dienststeuerungsrichtlinien (SCPs) — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie alle Funktionen in einer Organisation aktivieren, können Sie Richtlinien zur Servicesteuerung (SCPs) auf einige oder alle Ihre Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Root-Benutzer des AWS-Kontos Entitäten. Weitere Informationen zu Organizations und SCPs finden Sie unter Richtlinien zur Servicesteuerung im AWS Organizations Benutzerhandbuch.
- Ressourcenkontrollrichtlinien (RCPs) — RCPs sind JSON-Richtlinien, mit denen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten festlegen können, ohne die IAM-Richtlinien aktualisieren zu müssen, die jeder Ressource zugeordnet sind, deren Eigentümer Sie sind. Das RCP schränkt die Berechtigungen für Ressourcen in Mitgliedskonten ein und kann sich auf die effektiven Berechtigungen für Identitäten auswirken, einschließlich der Root-Benutzer des AWS-Kontos, unabhängig davon, ob sie zu Ihrer Organisation gehören. Weitere Informationen zu Organizations RCPs, einschließlich einer Liste AWS-Services dieser Support-Leistungen RCPs, finden Sie unter Resource Control Policies (RCPs) im AWS Organizations Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAM-Benutzerhandbuch unter Bewertungslogik für Richtlinien.

So funktioniert Amazon Rekognition mit IAM

Bevor Sie mit IAM den Zugriff auf Amazon Rekognition verwalten können, sollten Sie sich darüber informieren, welche IAM-Funktionen Sie mit Amazon Rekognition verwenden können. Einen allgemeinen Überblick darüber, wie Amazon Rekognition und andere AWS Services mit IAM zusammenarbeiten, finden Sie unter AWS Services That Work with IAM im IAM-Benutzerhandbuch.

Themen

- Identitätsbasierte Amazon-Rekognition-Richtlinien
- Ressourcenbasierte Amazon-Rekognition-Richtlinien
- Amazon-Rekognition-IAM-Rollen

Identitätsbasierte Amazon-Rekognition-Richtlinien

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Amazon Rekognition unterstützt bestimmte Aktionen, Ressourcen und Bedingungsschlüssel. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Aktionen

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie der zugehörige AWS API-Vorgang. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keinen passenden API-Vorgang gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Richtlinienaktionen in Amazon Rekognition verwenden das folgende Präfix vor der Aktion: `rekognition:`. Wenn Sie beispielsweise einer Person mittels der Amazon-Rekognition-

API-Operation `DetectLabels` die Berechtigung zum Erkennen von Objekten, Szenen oder Konzepten in einem Bild erteilen möchten, muss Ihre Richtlinie die `rekognition:DetectLabels`-Aktion enthalten. Richtlinienanweisungen müssen entweder ein – `Action`oder ein `NotAction`-Element enthalten. Amazon Rekognition definiert eine eigene Gruppe von Aktionen, die Aufgaben beschreiben, die Sie mit diesem Service durchführen können.

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie wie folgt durch Kommata:

```
"Action": [
      "rekognition:action1",
      "rekognition:action2"
```

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Describe` beginnen, einschließlich der folgenden Aktion:

```
"Action": "rekognition:Describe*"
```

Eine Liste der Aktionen für Amazon Rekognition finden Sie unter Von Amazon Rekognition definierte Aktionen im IAM-Benutzerhandbuch.

Ressourcen

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Weitere Informationen zum Format von ARNs finden Sie unter Amazon Resource Names (ARNs) und AWS Service Namespaces.

Wenn Sie beispielsweise die `MyCollection`-Sammlung in Ihrer Anweisung angeben möchten, verwenden Sie den folgenden ARN.

```
"Resource": "arn:aws:rekognition:us-east-1:123456789012:collection/MyCollection"
```

Um alle Instances anzugeben, die zu einem bestimmten Konto gehören, verwenden Sie den Platzhalter (*):

```
"Resource": "arn:aws:rekognition:us-east-1:123456789012:collection/*"
```

Einige Amazon-Rekognition-Aktionen, z. B. das Erstellen von Ressourcen, können für bestimmte Ressourcen nicht ausgeführt werden. In diesen Fällen müssen Sie den Platzhalter (*) verwenden.

```
"Resource": "*"
```

Eine Liste der Amazon Rekognition-Ressourcentypen und ihrer ARNs Eigenschaften finden Sie unter Von Amazon Rekognition definierte Ressourcen im IAM-Benutzerhandbuch. Informationen zu den Aktionen, mit denen Sie den ARN einzelner Ressourcen angeben können, finden Sie unter Von Amazon Rekognition definierte Aktionen.

Bedingungsschlüssel

Amazon Rekognition stellt keine servicespezifischen Bedingungsschlüssel bereit, unterstützt jedoch die Verwendung einiger globaler Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter AWS Global Condition Context Keys im IAM-Benutzerhandbuch.

Ressourcenbasierte Amazon-Rekognition-Richtlinien

Amazon Rekognition unterstützt ressourcenbasierte Richtlinien für Kopiervorgänge des Custom-Labels-Modells. Weitere Informationen finden Sie unter Beispiele für ressourcenbasierte Amazon-Rekognition-Richtlinien.

Andere Services, z. B. Amazon-S3, unterstützen auch ressourcenbasierte Berechtigungsrichtlinien. Beispielsweise können Sie einem S3 Bucket eine Richtlinie zuweisen, um die Zugriffsberechtigungen für diesen Bucket zu verwalten.

Um auf Bilder zuzugreifen, die in einem Amazon-S3-Bucket gespeichert sind, benötigen Sie eine Zugriffsberechtigung für Objekte im S3-Bucket. Mit dieser Berechtigung kann Amazon Rekognition Bilder aus dem S3-Bucket herunterladen. Die folgende Beispielrichtlinie ermöglicht es dem Benutzer, die `s3:GetObject` Aktion auf dem S3-Bucket mit dem Namen amzn-s3-demo-bucket3 auszuführen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket3/*"
            ]
        }
    ]
}
```

Um einen S3-Bucket mit aktivierter Versionsverwaltung zu nutzen, fügen Sie die `s3:GetObjectVersion`-Aktion wie im nachfolgenden Beispiel gezeigt hinzu.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket3/*"
            ]
        }
    ]
```

Amazon-Rekognition-IAM-Rollen

Eine IAM-Rolle ist eine Entität innerhalb Ihres Kontos, die über bestimmte Berechtigungen verfügt. AWS

Verwenden temporärer Anmeldeinformationen mit Amazon Rekognition

Sie können temporäre Anmeldeinformationen verwenden, um sich über einen Verbund anzumelden, eine IAM-Rolle anzunehmen oder eine kontenübergreifende Rolle anzunehmen. Sie erhalten temporäre Sicherheitsanmeldedaten, indem Sie AWS STS API-Operationen wie AssumeRoleoder GetFederationTokenaufrufen.

Amazon Rekognition unterstützt die Verwendung temporärer Anmeldeinformationen.

Service-verknüpfte Rollen

Mit dienstbezogenen Rollen können AWS Dienste auf Ressourcen in anderen Diensten zugreifen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem IAM-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für serviceverknüpfte Rollen anzeigen, aber nicht bearbeiten.

Amazon Rekognition unterstützt keine serviceverknüpften Rollen.

Servicerollen

Dieses Feature ermöglicht einem Service das Annehmen einer Servicerolle in Ihrem Namen. Diese Rolle gewährt dem Service Zugriff auf Ressourcen in anderen Diensten, um eine Aktion in Ihrem Namen auszuführen. Servicerollen werden in Ihrem IAM-Konto angezeigt und gehören zum Konto. Dies bedeutet, dass ein IAM-Administrator die Berechtigungen für diese Rolle ändern kann. Dies kann jedoch die Funktionalität des Dienstes beeinträchtigen.

Amazon Rekognition unterstützt Servicerollen.

Die Verwendung einer Servicerolle kann zu einem Sicherheitsproblem führen, wenn Amazon Rekognition verwendet wird, um einen anderen Service aufzurufen und auf Ressourcen zu reagieren, auf die dieser keinen Zugriff haben sollte. Um die Sicherheit Ihres Kontos zu gewährleisten, sollten Sie den Zugriff von Amazon Rekognition auf die Ressourcen beschränken, die Sie verwenden. Dies kann erreicht werden, indem Sie Ihrer IAM-Servicerolle eine Vertrauensrichtlinie hinzufügen. Weitere Informationen hierzu finden Sie unter Serviceübergreifende Confused-Deputy-Prävention.

Auswählen einer IAM-Rolle in Amazon Rekognition

Wenn Sie für die Analyse gespeicherter Videos Amazon Rekognition konfigurieren, müssen Sie eine Rolle auswählen, mit der Amazon Rekognition für Sie auf Amazon SNS zugreifen kann. Wenn Sie zuvor eine Servicerolle oder serviceverknüpfte Rolle erstellt haben, stellt Ihnen Amazon Rekognition

eine Liste mit Rollen bereit, aus denen Sie wählen können. Weitere Informationen finden Sie unter the section called “Amazon Rekognition Video konfigurieren”.

Beispiel: Konfiguration von Amazon Rekognition für den Zugriff auf Bilder in einem Amazon S3 S3-Bucket

Im Folgenden finden Sie ein Beispiel dafür, wie Sie Amazon Rekognition für die Analyse von Bildern in einem Amazon S3 S3-Bucket konfigurieren könnten. Wenn Sie Amazon Rekognition verwenden möchten, um Bilder in einem Amazon S3 S3-Bucket zu analysieren, müssen Sie wie folgt vorgehen:

1. Stellen Sie sicher, dass Ihr IAM-Benutzer/Ihre IAM-Rolle (der Client) berechtigt ist, die entsprechenden Amazon Rekognition Rekognition-API-Operationen (wie usw.) aufzurufen DetectLabels DetectFaces

   Fügen Sie eine identitätsbasierte Richtlinie bei, die die entsprechenden Berechtigungen zum Aufrufen Ihrer gewünschten API-Operationen gewährt. Um Ihrer Rolle beispielsweise die Berechtigung zum Aufrufen von `DetectLabels` und zu erteilen`DetectFaces`, würden Sie Ihrer Rolle eine Richtlinie hinzufügen, die wie folgt aussieht:

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "rekognition:DetectLabels",
                   "rekognition:DetectFaces"
                   // other Rekognition permissions as needed
               ],
               "Resource": "*"
           }
       ]
   }
   ```

2. Der Amazon Rekognition Rekognition-Dienst benötigt eine Genehmigung für den Zugriff auf Ihren Amazon S3 S3-Bucket. Erstellen Sie eine IAM-Servicerolle, die Sie bei API-Aufrufen an Amazon Rekognition übergeben müssen. Stellen Sie sicher, dass die Servicerolle: Vertraut dem Amazon Rekognition Service Principal, hat `s3:GetObject` Berechtigungen für Ihren Bucket.

   Die Vertrauensrichtlinie könnte wie folgt aussehen:

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

Die identitätsbasierte Richtlinie, die der Servicerolle zugeordnet ist, könnte wie folgt aussehen:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

AWS verwaltete Richtlinien für Amazon Rekognition

Um Benutzern, Gruppen und Rollen Berechtigungen hinzuzufügen, ist es einfacher, AWS verwaltete Richtlinien zu verwenden, als Richtlinien selbst zu schreiben. Es erfordert Zeit und Fachwissen, um von Kunden verwaltete IAM-Richtlinien zu erstellen, die Ihrem Team nur die benötigten Berechtigungen bieten. Um schnell loszulegen, können Sie unsere AWS verwalteten Richtlinien verwenden. Diese Richtlinien decken allgemeine Anwendungsfälle ab und sind in Ihrem AWS Konto

verfügbar. Weitere Informationen zu AWS verwalteten Richtlinien finden Sie unter AWS Verwaltete Richtlinien im IAM-Benutzerhandbuch.

AWS Dienste verwalten und aktualisieren AWS verwaltete Richtlinien. Sie können die Berechtigungen in AWS verwalteten Richtlinien nicht ändern. Services fügen einer von AWS verwalteten Richtlinien gelegentlich zusätzliche Berechtigungen hinzu, um neue Features zu unterstützen. Diese Art von Update betrifft alle Identitäten (Benutzer, Gruppen und Rollen), an welche die Richtlinie angehängt ist. Services aktualisieren eine von AWS verwaltete Richtlinie am ehesten, ein neues Feature gestartet wird oder neue Vorgänge verfügbar werden. Dienste entfernen keine Berechtigungen aus einer AWS verwalteten Richtlinie, sodass durch Richtlinienaktualisierungen Ihre bestehenden Berechtigungen nicht beeinträchtigt werden.

AWS Unterstützt außerdem verwaltete Richtlinien für Jobfunktionen, die sich über mehrere Dienste erstrecken. Die ReadOnlyAccess AWS verwaltete Richtlinie bietet beispielsweise schreibgeschützten Zugriff auf alle AWS Dienste und Ressourcen. Wenn ein Dienst eine neue Funktion startet, werden nur Leseberechtigungen für neue Operationen und Ressourcen AWS hinzugefügt. Eine Liste und Beschreibungen der Richtlinien für Auftragsfunktionen finden Sie in Verwaltete AWS -Richtlinien für Auftragsfunktionen im IAM-Leitfaden.

Von AWS verwaltete Richtlinie: AmazonRekognitionFullAccess

`AmazonRekognitionFullAccess` gewährt vollen Zugriff auf die Ressourcen von Amazon Rekognition, einschließlich der Erstellung und Löschung von Sammlungen.

Sie können die `AmazonRekognitionFullAccess`-Richtlinie an Ihre IAM-Identitäten anfügen.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

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

```
            "Resource": "*"
        }
    ]
}
```

Von AWS verwaltete Richtlinie: AmazonRekognitionReadOnlyAccess

`AmazonRekognitionReadOnlyAccess` gewährt reinen Lesezugriff auf Amazon-Rekognition-Ressourcen.

Sie können die `AmazonRekognitionReadOnlyAccess`-Richtlinie an Ihre IAM-Identitäten anfügen.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

```
{
    "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": "*"
        }
    ]
}
```

Von AWS verwaltete Richtlinie: AmazonRekognitionServiceRole

`AmazonRekognitionServiceRole` ermöglicht Amazon Rekognition, Amazon Kinesis Data Streams und Amazon-SNS-Services in Ihrem Namen aufzurufen.

Sie können die `AmazonRekognitionServiceRole`-Richtlinie an Ihre IAM-Identitäten anfügen.

Wenn Sie diese Servicerolle nutzen, sollten Sie Ihr Konto schützen, indem Sie den Zugriff von Amazon Rekognition nur auf die Ressourcen beschränken, die Sie verwenden. Dies kann erreicht werden, indem Sie Ihrer IAM-Servicerolle eine Vertrauensrichtlinie hinzufügen. Weitere Informationen hierzu finden Sie unter Serviceübergreifende Confused-Deputy-Prävention.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

```
{
    "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": "*"
        }
    ]
}
```

Von AWS verwaltete Richtlinie: AmazonRekognitionCustomLabelsFullAccess

Diese Richtlinie gilt für Benutzer von Amazon Rekognition Custom Labels. Verwenden Sie die AmazonRekognitionCustomLabelsFullAccess Richtlinie, um Benutzern vollen Zugriff auf die Amazon Rekognition Custom Labels API und vollen Zugriff auf die Konsolen-Buckets zu gewähren, die von der Amazon Rekognition Custom Labels-Konsole erstellt wurden.

Details zu Berechtigungen

Diese Richtlinie umfasst die folgenden Berechtigungen.

```
{
    "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 aktualisiert verwaltete Richtlinien AWS

Sehen Sie sich Details zu Aktualisierungen der AWS verwalteten Richtlinien für Amazon Rekognition an, seit dieser Service begonnen hat, diese Änderungen zu verfolgen. Um automatische Warnungen über Änderungen an dieser Seite zu erhalten, abonnieren Sie den RSS-Feed auf der Amazon-Rekognition-Dokumentverlauf-Seite.

Änderung	Beschreibung	Datum
Aktionen, die Medienanalyseaufträge beinhalten, wurden der folgenden verwalteten Richtlinie hinzugefügt: • Von AWS verwaltete Richtlinie: AmazonRekognitionReadOnlyAccess	Amazon Rekognition hat die folgenden Aktionen zu der `AmazonRekognitionReadOnlyAccess` -verwalteten Richtlinie hinzugefügt: • `GetMediaAnalysisJob` • `ListMediaAnalysisJob`	31. Oktober 2023
Aktionen zur Verwaltung von Benutzern wurden der folgenden verwalteten Richtlinie hinzugefügt: • Von AWS verwaltete Richtlinie: AmazonRekognitionReadOnlyAccess	Amazon Rekognition hat die folgenden Aktionen zu der `AmazonRekognitionReadOnlyAccess` -verwalteten Richtlinie hinzugefügt: • `ListUsers` • `SearchUsers` • `SearchUsersByImage`	12. Juni 2023
Zu den folgenden verwalteten Richtlinien wurden Aktionen für ProjectPolicy und das Modell „Benutzerdefiniertes Label-Modell kopieren“ hinzugefügt: • Von AWS verwaltete Richtlinie: AmazonRekognitionFullAccess	Amazon Rekognition hat die folgenden Aktionen zu den `AmazonRekognitionCustomLabelsFullAccess` - und `AmazonRekognitionFullAccess` -verwalteten Richtlinien hinzugefügt: • `CopyProjectVersion`	21. Juli 2022

Änderung	Beschreibung	Datum
• Von AWS verwaltete Richtlinie: AmazonRekognitionCustomLabelsFullAccess	• `PutProjectPolicy` • `ListProjectPolicies` • `DeleteProjectPolicy`	
Die folgenden verwaltet en Richtlinien wurden um Aktionen für ProjectPolicy und das Modell „Benutzerdefiniert es Label-Modell kopieren“ erweitert: • Von AWS verwaltete Richtlinie: AmazonRekognitionReadOnlyAccess	Amazon Rekognition hat die folgenden Aktionen zu der AmazonRekognitionR eadOnlyAccess -verwalteten Richtlinie hinzugefügt: • `ListProjectPolicies`	21. Juli 2022

Änderung	Beschreibung	Datum
Update zur Datensatz verwaltung für die folgenden verwalteten Richtlinien: • Von AWS verwaltete Richtlinie: AmazonRekognitionReadOnlyAccess • Von AWS verwaltete Richtlinie: AmazonRekognitionFullAccess • Von AWS verwaltete Richtlinie: AmazonRekognitionCustomLabelsFullAccess	Amazon Rekognition hat die folgenden Aktionen zu den verwalteten AmazonRekognitionReadOnlyAccess Richtlinien AmazonRekognitionFullOnlyAccess, und AmazonRekognitionCustomLabelsFullAccess hinzugefügt • `CreateDataset` • `ListDatasetEntries` • `ListDatasetLabels` • `DescribeDataset` • `UpdateDatasetEntries` • `DistributeDatasetEntries` • `DeleteDataset`	1. November 2021
Tagging-Update für Von AWS verwaltete Richtlinie: AmazonRekognitionReadOnlyAccess und Von AWS verwaltete Richtlinie: AmazonRekognitionFullAccess	Amazon Rekognition hat den Richtlinien und neue Tagging-Aktionen hinzugefügt. AmazonRekognitionFullAccess AmazonRekognitionReadOnlyAccess	2. April 2021
Amazon Rekognition hat mit der Verfolgung von Änderungen begonnen	Amazon Rekognition hat damit begonnen, Änderungen an seinen AWS verwalteten Richtlinien nachzuverfolgen.	2. April 2021

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien können festlegen, ob jemand Amazon-Rekognition-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter AWS -verwaltete Richtlinien oder AWS -verwaltete Richtlinien für Auftrags-Funktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. AWS CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere

und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung mit IAM Access Analyzer im IAM-Benutzerhandbuch.

- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Sicherer API-Zugriff mit MFA im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der Amazon-Rekognition-Konsole

Mit Ausnahme des Features für Amazon Rekognition Custom Labels benötigt Amazon Rekognition keine zusätzlichen Berechtigungen, wenn Sie die Amazon-Rekognition-Konsole verwenden. Informationen zu Amazon Rekognition Custom Labels finden Sie unter Schritt 5: Einrichten von Amazon-Rekognition-Custom-Labels-Konsolenberechtigungen.

Sie müssen Benutzern, die nur die API oder die API aufrufen, keine Mindestberechtigungen für die AWS CLI Konsole gewähren. AWS Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die Sie ausführen möchten.

Beispiele für die Verwendung identitätsbasierter Richtlinien von Amazon Rekognition

Benutzer und Rollen besitzen standardmäßig keine Berechtigungen zum Erstellen oder Ändern von Amazon-Rekognition-Ressourcen. Sie können auch keine Aufgaben mit der AWS Management Console AWS CLI, oder AWS API ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern und Rollen die Berechtigung zum Ausführen bestimmter API-Operationen für die angegebenen Ressourcen gewähren, die diese benötigen. Der Administrator muss diese Richtlinien anschließend den -Benutzern oder -Gruppen anfügen, die diese Berechtigungen benötigen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von Richtlinien auf der JSON-Registerkarte im IAM-Benutzerhandbuch.

Themen

- Beispiel für die Richtlinien von Amazon Rekognition Custom Labels
- Beispiel 1: Gewähren Sie einem Benutzer schreibgeschützten Zugriff auf Ressourcen
- Beispiel 2: Gewähren Sie einem Benutzer kompletten Zugriff auf Ressourcen
- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

Beispiel für die Richtlinien von Amazon Rekognition Custom Labels

Sie können identitätsbasierte Richtlinien für Amazon Rekognition Custom Labels erstellen. Weitere Informationen finden Sie unter Sicherheit.

Beispiel 1: Gewähren Sie einem Benutzer schreibgeschützten Zugriff auf Ressourcen

Im folgenden Beispiel wird schreibgeschützter Zugriff auf alle Amazon-Rekognition-Ressourcen gewährt.

```
{
    "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": "*"
        }
    ]
}
```

Beispiel 2: Gewähren Sie einem Benutzer kompletten Zugriff auf Ressourcen

Im folgenden Beispiel wird vollständiger Zugriff auf alle Amazon-Rekognition-Ressourcen gewährt.

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

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie beinhaltet Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der API AWS CLI oder AWS .

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Beispiele für ressourcenbasierte Richtlinien von Amazon Rekognition

Amazon Rekognition Custom Labels verwendet ressourcenbasierte Richtlinien, sogenannte Projektrichtlinien, um Kopierberechtigungen für eine Modellversion zu verwalten.

Eine Projektrichtlinie erteilt oder verweigert die Erlaubnis, eine Modellversion von einem Quellprojekt in ein Zielprojekt zu kopieren. Sie benötigen eine Projektrichtlinie, wenn sich das Zielprojekt in einem anderen AWS Konto befindet oder wenn Sie den Zugriff innerhalb eines AWS Kontos einschränken möchten. Beispielsweise möchten Sie möglicherweise Kopierberechtigungen für eine bestimmte IAM-Rolle verweigern. Weitere Informationen finden Sie unter Kopieren eines Modells.

Erteilen der Berechtigung zum Kopieren einer Modellversion

Das folgende Beispiel ermöglicht es dem Prinzipal `arn:aws:iam::123456789012:role/Admin`, die Modellversion `arn:aws:rekognition:us-east-1:123456789012:project/my_project/version/test_1/1627045542080` zu kopieren.

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

Fehlerbehebung für Amazon-Rekognition-Identität und -Zugriff

Diagnostizieren und beheben Sie mithilfe der folgenden Informationen gängige Probleme, die bei der Verwendung von Amazon Rekognition und IAM auftreten können.

Themen

- Ich bin nicht autorisiert, eine Aktion in Amazon Rekognition auszuführen
- Ich bin nicht berechtigt, iam auszuführen: PassRole
- Ich bin Administrator und möchte anderen Zugriff auf Amazon Rekognition gewähren.
- Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meine Amazon Rekognition Rekognition-Ressourcen ermöglichen

Ich bin nicht autorisiert, eine Aktion in Amazon Rekognition auszuführen

Wenn Ihnen AWS Management Console mitgeteilt wird, dass Sie nicht berechtigt sind, eine Aktion auszuführen, müssen Sie sich an Ihren Administrator wenden, um Unterstützung zu erhalten. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, die Konsole zum Anzeigen von Details zu einem *`widget`* zu verwenden, jedoch nicht über `rekognition:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 rekognition:GetWidget on resource: my-example-widget
```

In diesem Fall bittet Mateo seinen Administrator um die Aktualisierung seiner Richtlinien, um unter Verwendung der Aktion *`my-example-widget`* auf die Ressource `rekognition:`*`GetWidget`* zugreifen zu können.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zur Ausführung der Aktion „`iam:PassRole`“ autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an Amazon Rekognition übergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Amazon Rekognition auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin Administrator und möchte anderen Zugriff auf Amazon Rekognition gewähren.

Um anderen den Zugriff auf Amazon Rekognition zu ermöglichen, müssen Sie den Personen oder Anwendungen, die Zugriff benötigen, die Erlaubnis erteilen. Wenn Sie Personen und Anwendungen verwalten, weisen Sie Benutzern oder Gruppen Berechtigungssätze zu, um deren Zugriffsebene zu

definieren. AWS IAM Identity Center Mit Berechtigungssätzen werden automatisch IAM-Richtlinien erstellt und den IAM-Rollen zugewiesen, die der Person oder Anwendung zugeordnet sind. Weitere Informationen finden Sie im AWS IAM Identity Center Benutzerhandbuch unter Berechtigungssätze.

Wenn Sie IAM Identity Center nicht verwenden, müssen Sie IAM-Entitäten (Benutzer oder Rollen) für die Personen oder Anwendungen erstellen, die Zugriff benötigen. Anschließend müssen Sie der Entität eine Richtlinie anfügen, durch die dieser die korrekten Berechtigungen in Amazon Rekognition gewährt werden. Nachdem die Berechtigungen erteilt wurden, stellen Sie dem Benutzer oder Anwendungsentwickler die Anmeldeinformationen zur Verfügung. Sie werden diese Anmeldeinformationen für den Zugriff verwenden AWS. Weitere Informationen zum Erstellen von IAM-Benutzern, -Gruppen, -Richtlinien und -Berechtigungen finden Sie im IAM-Benutzerhandbuch unter IAM-Identitäten sowie Richtlinien und Berechtigungen in IAM.

Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meine Amazon Rekognition Rekognition-Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen dazu, ob Amazon Rekognition diese Funktionen unterstützt, finden Sie unter So funktioniert Amazon Rekognition mit IAM.
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen in AWS-Konten Ihrem Besitz gewähren können, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen AWS-Konto , dem Sie gehören.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Datenschutz in Amazon Rekognition

Das Modell der AWS gemeinsamen Verantwortung gilt für den Datenschutz in Amazon Rekognition. Wie in diesem Modell beschrieben, AWS ist es verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen. AWS Cloud Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Verwenden Sie SSL/TLS, um mit Ressourcen zu kommunizieren. AWS Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein. AWS CloudTrail Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dazu gehört auch, wenn Sie mit Rekognition oder anderen AWS-Services über die Konsole AWS CLI, API oder arbeiten. AWS SDKs Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen

Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Datenverschlüsselung

In den folgenden Informationen wird erläutert, wo Amazon Rekognition Datenverschlüsselung verwendet, um Ihre Daten zu schützen.

Verschlüsselung im Ruhezustand

Amazon Rekognition Image

Bilder

Bilder, die an Amazon-Rekognition-API-Operationen weitergegeben werden, können gespeichert und zur Verbesserung des Service verwendet werden, sofern Sie sich nicht abgemeldet haben, indem Sie die Seite mit den Abmelderichtlinien für KI-Dienste aufrufen und den dort beschriebenen Prozess befolgen. Die gespeicherten Bilder werden in Ruhe (Amazon S3) mit AWS Key Management Service (SSE-KMS) verschlüsselt.

Sammlungen

Bei Gesichtsvergleichsoperationen, die Informationen in einer Sammlung speichern, erkennt der zugrunde liegende Erkennungsalgorithmus zunächst die Gesichter im Eingabebild, extrahiert einen Vektor für jedes Gesicht und speichert dann die Gesichtsvektoren in der Sammlung. Amazon Rekognition verwendet diese Gesichtsvektoren beim Gesichtsvergleich. Gesichtsvektoren werden als Array von Fließkommazahlen gespeichert und im Ruhezustand verschlüsselt.

Amazon Rekognition Video

Videos

Um ein Video zu analysieren, kopiert Amazon Rekognition Ihre Videos zur Verarbeitung in den Service. Das Video kann gespeichert und zur Verbesserung des Service verwendet werden, sofern Sie sich nicht abgemeldet haben, indem Sie die Seite mit den Abmelderichtlinien für KI-Dienste aufrufen und den dort beschriebenen Prozess befolgen. Die Videos werden in Ruhe (Amazon S3) mit AWS Key Management Service (SSE-KMS) verschlüsselt.

Amazon Rekognition Custom Labels

Amazon Rekognition Custom Labels verschlüsselt Ihre Daten im Ruhezustand.

Bilder

Um Ihr Modell zu trainieren, erstellt Amazon Rekognition Custom Labels eine Kopie Ihrer ursprünglichen Trainings- und Testbilder. Die kopierten Bilder werden im Ruhezustand in Amazon Simple Storage Service (S3) mithilfe einer serverseitigen Verschlüsselung mit einem AWS KMS key von Ihnen bereitgestellten oder einem AWS eigenen KMS-Schlüssel verschlüsselt. Amazon Rekognition Custom Labels unterstützt nur symmetrische KMS-Schlüssel. Ihre Quellbilder sind davon nicht betroffen. Weitere Informationen finden Sie unter Trainieren eines Modells für Amazon Rekognition Custom Labels.

Modelle

Standardmäßig verschlüsselt Amazon Rekognition Custom Labels trainierte Modelle und Manifestdateien, die in Amazon-S3-Buckets mit der serverseitigen Verschlüsselung mit einem AWS-eigener Schlüssel gespeichert werden. Weitere Informationen finden Sie unter Schutz von Daten mittels serverseitiger Verschlüsselung. Die Trainingsergebnisse werden in den Bucket geschrieben, der im `OutputConfig` Eingabeparameter to CreateProjectVersionangegeben ist. Die Trainingsergebnisse werden mit den konfigurierten Verschlüsselungseinstellungen für den Bucket (`OutputConfig`) verschlüsselt.

Konsolen-Bucket

Die Amazon-Rekognition-Custom-Labels-Konsole erstellt einen Amazon-S3-Bucket (Konsolen-Bucket), mit dem Sie Ihre Projekte verwalten können. Der Konsolen-Bucket ist mit der standardmäßigen Amazon-S3-Verschlüsselung verschlüsselt. Weitere Informationen finden Sie unter Amazon-Simple-Storage-Service-Standardverschlüsselung für S3-Buckets. Wenn Sie Ihren eigenen KMS-Schlüssel verwenden, konfigurieren Sie den Konsolen-Bucket, nachdem er erstellt wurde. Weitere Informationen finden Sie unter Schutz von Daten mittels serverseitiger Verschlüsselung. Amazon Rekognition Custom Labels blockiert den öffentlichen Zugriff auf den Konsolen-Bucket.

Rekognition Face Liveness

Alle sitzungsbezogenen Daten, die im Konto des Rekognition-Face-Liveness-Dienstes gespeichert wurden, werden im Ruhezustand vollständig verschlüsselt. Standardmäßig werden Referenz- und Auditbilder mit einem AWS -eigenen Schlüssel im Dienstkonto verschlüsselt. Sie können sich jedoch dafür entscheiden, Ihre eigenen AWS KMS Schlüssel für die Verschlüsselung dieser Bilder bereitzustellen.

Verschlüsselung während der Übertragung

Amazon-Rekognition-API-Endpunkte unterstützen ausschließlich sichere Verbindungen über HTTPS. Die gesamte Kommunikation wird mit Transport Layer Security (TLS) verschlüsselt.

Schlüsselverwaltung

Sie können AWS Key Management Service (KMS) verwenden, um Schlüssel für die Eingabebilder und -videos zu verwalten, die Sie in Amazon-S3-Buckets speichern. Weitere Informationen finden Sie unter AWS-Key-Management-Service-Konzepte.

Vom Kunden verwaltete Schlüsselverschlüsselung für Face Liveness (Echtheit von Gesichtern)

Die CreateFaceLivenessSessionAPI verwendet einen optionalen `KmsKeyId` Parameter. Sie können den `id` des KMS-Schlüssels angeben, den Sie in Ihrem Konto erstellt haben. Dieser Schlüssel wird verwendet, um Referenz- und Audit-Images zu verschlüsseln, die während der StartFaceLivenessSessionAPI abgerufen wurden. Während der GetFaceLivenessSessionResultsAPI werden die Bilder mit diesem Schlüssel entschlüsselt, bevor die Ergebnisse zurückgegeben werden. Wenn die CreateFaceLivenessSession Anfrage eine enthielt OutputConfig, werden die Referenz- und Audit-Images in die angegebenen Amazon S3 S3-Pfade hochgeladen. Wir empfehlen, die serverseitige Verschlüsselung (SSE-S3) in Ihren Amazon-S3-Buckets zu aktivieren, damit die Daten im Ruhezustand weiterhin verschlüsselt bleiben.

Wenn Sie Ihre eigene AWS KMS Schlüssel-ID angeben, erhält der Rekognition Face Liveness Service die Erlaubnis, den vom Kunden verwalteten Schlüssel im Namen des Prinzipals zu verwenden, der den aufruft. APIs Die Principals (Benutzer oder Rollen), mit denen das Backend APIs vom Kunden aufgerufen wurde (APIs`CreateFaceLivenessSession`und`GetFaceLivenessSessionResults`) müssen Zugriff haben, um Folgendes ausführen zu können:

- km: DescribeKey
- km: GenerateDataKey
- kms:Decrypt

Richtlinie für den Datenverkehr zwischen Netzwerken

Ein Amazon Virtual Private Cloud (Amazon VPC)-Endpunkt für Amazon Rekognition ist eine logische Einheit innerhalb einer VPC, die nur Konnektivität mit Amazon Rekognition ermöglicht. Amazon VPC

leitet Anforderungen an Amazon Rekognition weiter und leitet Antworten an die VPC zurück. Weitere Informationen finden Sie unter VPC-Endpunkte im Amazon VPC-Benutzerhandbuch. Informationen zur Verwendung von Amazon-VPC-Endpunkten mit Amazon Rekognition finden Sie unter Verwenden von Amazon Rekognition mit Amazon-VPC-Endpunkten.

Verwenden von Amazon Rekognition mit Amazon-VPC-Endpunkten

Wenn Sie Amazon Virtual Private Cloud (Amazon VPC) zum Hosten Ihrer AWS-Ressourcen verwenden, können Sie eine Verbindung zwischen Ihrer VPC und Amazon Rekognition herstellen. Sie können diese Verbindung verwenden, um Amazon Rekognition die Kommunikation mit Ihren Ressourcen in Ihrer VPC zu ermöglichen, ohne über das öffentliche Internet gehen zu müssen.

Amazon VPC ist ein AWS-Service, den Sie verwenden können, um AWS-Ressourcen in einem von Ihnen definierten virtuellen Netzwerk auszuführen. Mit einer VPC haben Sie die Kontrolle über Ihre Netzwerkeinstellungen, wie IP-Adressbereich, Subnetze, Routing-Tabellen und Netzwerk-Gateways. Mit VPC-Endpunkten übernimmt das AWS-Netzwerk das Routing zwischen der VPC und den AWS-Services.

Um Ihre VPC mit Amazon Rekognition zu verbinden, definieren Sie einen Schnittstellen-VPC-Endpunkt für Amazon Rekognition. Ein Schnittstellenendpunkt ist eine elastic network interface mit einer privaten IP-Adresse, die als Einstiegspunkt für Datenverkehr dient, der für einen unterstützten AWS Dienst bestimmt ist. Der Endpunkt bietet eine zuverlässige, skalierbare Konnektivität zu Amazon Rekognition und benötigt weder ein Internet-Gateway noch eine NAT-Instance (Network Address Translation) oder eine VPN-Verbindung. Weitere Informationen finden Sie unter Was ist Amazon VPC im Benutzerhandbuch zu Amazon VPC.

Schnittstellen-VPC-Endpunkte werden von AWS aktiviert. PrivateLink Diese AWS-Technologie ermöglicht die private Kommunikation zwischen AWS-Services über eine Elastic Network-Schnittstelle mit privaten IP-Adressen.

> **Note**
>
> Alle Amazon Rekognition Federal Information Processing Standard (FIPS) -Endgeräte werden von AWS unterstützt. PrivateLink

Erstellen von Amazon VPC-Endpunkten für Amazon Rekognition

Sie können zwei Arten von Amazon-VPC-Endpunkten zur Verwendung mit Amazon Rekognition erstellen.

- Ein VPC-Endpunkt zur Verwendung mit Amazon-Rekognition-Operationen. Für die meisten Benutzer ist dies der am besten geeignete Typ von VPC-Endpunkt.
- Ein VPC-Endpunkt für Amazon-Rekognition-Operationen mit Endpunkten, die dem US-Regierungsstandard Federal Information Processing Standard (FIPS) Publication 140-2 entsprechen.

Um Amazon Rekognition mit Ihrer VPC zu verwenden, verwenden Sie die Amazon-VPC-Konsole, um einen VPC-Endpunkt für Amazon Rekognition zu erstellen. Anweisungen finden Sie im Verfahren „So erstellen Sie einen Schnittstellenendpunkt zu einem AWS-Service mithilfe der Konsole“ unter Erstellen eines Schnittstellenendpunkts. Beachten Sie die folgenden Verfahrensschritte:

- Schritt 3: Wählen Sie für Servicekategorie die Option AWS-Services.
- Schritt 4: Wählen Sie für Servicename eine der folgenden Optionen aus:
  - com.amazonaws.region.rekognition: Erzeugt einen VPC-Endpunkt für Amazon-Rekognition-Operationen.
  - com.amazonaws.region.rekognition-fips: Erstellt einen VPC-Endpunkt für Amazon-Rekognition-Operationen mit Endpunkten, die dem US-Regierungsstandard Federal Information Processing Standard (FIPS) Publication 140-2 entsprechen.

Weitere Informationen finden Sie unter Erste Schritte im Amazon VPC Benutzerhandbuch.

Eine VPC-Endpunktrichtlinie für Amazon Rekognition erstellen

Sie können eine Richtlinie für Amazon-VPC-Endpunkte für Amazon Rekognition erstellen, in der Sie Folgendes angeben:

- Prinzipal, der die Aktionen ausführen kann.
- Aktionen, die ausgeführt werden können
- Die Ressourcen, für die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter Steuerung des Zugriffs auf Services mit VPC-Endpunkten im Amazon VPC User Guide.

Mit der folgenden Beispielrichtlinie können Benutzer, die über den VPC-Endpunkt eine Verbindung zu Amazon Rekognition herstellen, die `DetectFaces`-API-Operation aufrufen. Über die Richtlinie wird verhindert, dass Benutzer andere Amazon-Rekognition-API-Operationen über den VPC-Endpunkt ausführen können.

Benutzer können weiterhin andere Amazon-Rekognition-API-Operationen von außerhalb der VPC aufrufen. Informationen zum Ablehnen des Zugriffs auf Amazon-Rekognition-API-Operationen außerhalb der VPC finden Sie unter Identitätsbasierte Amazon-Rekognition-Richtlinien.

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

So ändern Sie die VPC-Endpunktrichtlinie für Amazon Rekognition

1. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wenn Sie den Endpunkt für Amazon Rekognition noch nicht erstellt haben, wählen Sie Endpunkt erstellen. Wählen Sie dann com.amazonaws aus. ***Region***.rekognition und wählen Sie Create Endpoint.
3. Wählen Sie im Navigationsbereich Endpunkte aus.
4. Wählen Sie com.amazonaws aus. ***Region***.rekognition-Endpunkt und wählen Sie in der unteren Hälfte des Bildschirms den Tab Richtlinie aus.
5. Wählen Sie Richtlinie bearbeiten und nehmen Sie die Änderungen an der Richtlinie vor.

Compliance-Validierung für Amazon Rekognition

Externe Prüfer bewerten die Sicherheit und Konformität von Amazon Rekognition im Rahmen mehrerer AWS Compliance-Programme. Hierzu zählen unter anderem SOC, PCI, FedRAMP und HIPAA.

Eine Liste der AWS Services im Rahmen bestimmter Compliance-Programme finden Sie unter AWS-Services in Umfang nach Compliance-Programmen. Allgemeine Informationen finden Sie unter AWS -Compliance-Programme.

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Herunterladen von Berichten in AWS Artifact.

Ihre Compliance-Verantwortung bei der Nutzung von Amazon Rekognition richtet sich nach der Sensibilität Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften. AWS bietet die folgenden Ressourcen zur Unterstützung bei der Einhaltung von Vorschriften:

- Schnellstartanleitungen zu Sicherheit und Compliance — In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Implementierung von Umgebungen beschrieben, auf denen auf Sicherheit und Compliance ausgerichtete Basisumgebungen eingerichtet werden. AWS
- Whitepaper „Architecting for HIPAA Security and Compliance“ — In diesem Whitepaper wird beschrieben, wie Unternehmen HIPAA-konforme Anwendungen entwickeln können. AWS
- AWS Ressourcen zur Einhaltung von Vorschriften — Diese Sammlung von Arbeitsmappen und Leitfäden könnte für Ihre Branche und Ihren Standort gelten.
- AWS Config— Dieser AWS Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub— Dieser AWS Service bietet einen umfassenden Überblick über Ihren Sicherheitsstatus und hilft Ihnen AWS , die Einhaltung der Sicherheitsstandards und bewährten Verfahren der Sicherheitsbranche zu überprüfen.

Ausfallsicherheit bei Amazon Rekognition

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger

Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Zusätzlich zur AWS globalen Infrastruktur bietet Amazon Rekognition mehrere Funktionen, um Ihre Anforderungen an Datenstabilität und Datensicherung zu erfüllen.

Konfigurations- und Schwachstellenanalyse in Amazon Rekognition

Konfiguration und IT-Steuerung unterliegen der übergreifenden Verantwortlichkeit von AWS und Ihnen, unserem Kunden. Weitere Informationen finden Sie im Modell der AWS gemeinsamen Verantwortung.

Serviceübergreifende Confused-Deputy-Prävention

In kann es zu einem dienstübergreifenden Identitätswechsel kommen AWS, wenn ein Dienst (der anrufende Dienst) einen anderen Dienst (den aufgerufenen Dienst) aufruft. Der aufrufende Service kann so manipuliert werden, dass er auf die Ressourcen eines anderen Kunden reagiert, obwohl er nicht über die entsprechenden Berechtigungen verfügen sollte, was zu einem Verwirrter-Stellvertreter-Problem führt.

Um dies zu verhindern, werden Tools AWS bereitgestellt, mit denen Sie Ihre Daten für alle Dienste mit Dienstprinzipalen schützen können, denen Zugriff auf Ressourcen in Ihrem Konto gewährt wurde.

Wir empfehlen die Verwendung der globalen Bedingungskontext-Schlüssel `aws:SourceArn` und `aws:SourceAccount` in ressourcenbasierten Richtlinien, um die Berechtigungen, die Amazon Rekognition einem anderen Service erteilt, auf die Ressource zu beschränken.

Wenn der Wert von `aws:SourceArn` die Konto-ID nicht enthält, z. B. einen Amazon-S3-Bucket-ARN, müssen Sie beide Schlüssel verwenden, um Berechtigungen einzuschränken. Wenn Sie beide Schlüssel verwenden und der `aws:SourceArn`-Wert die Konto-ID enthält, müssen der `aws:SourceAccount`-Wert und das Konto im `aws:SourceArn`-Wert dieselbe Konto-ID verwenden, wenn sie in der gleichen Richtlinienanweisung verwendet wird.

Verwenden Sie `aws:SourceArn`, wenn Sie nur eine Ressource mit dem betriebsübergreifenden Zugriff verknüpfen möchten. Verwenden Sie `aws:SourceAccount`, wenn Sie zulassen möchten, dass Ressourcen in diesem Konto mit der betriebsübergreifenden Verwendung verknüpft werden.

Der Wert von `aws:SourceArn` muss der ARN der von Rekognition verwendeten Ressource sein, der im folgenden Format angegeben wird: `arn:aws:rekognition:region:account:resource`.

Der Wert von `arn:User ARN` sollte der ARN des Benutzers sein, der den Videoanalysevorgang aufruft (der Benutzer, der eine Rolle annimmt).

Der effektivste Weg, um sich vor dem Confused-Deputy-Problem zu schützen, ist die Verwendung des globalen `aws:SourceArn`-Bedingungskontextschlüssels mit dem vollständigen ARN der Ressource.

Wenn Sie den vollständigen ARN der Ressource nicht kennen oder wenn Sie mehrere Ressourcen angeben, verwenden Sie den `aws:SourceArn`-Schlüssel mit Platzhalterzeichen (*) für die unbekannten Teile des ARN. Beispiel, `arn:aws:`*`rekognition`*`:*:111122223333:*`.

Um sich vor dem Problem des verwechselten Stellvertreters zu schützen, führen Sie die folgenden Schritte durch:

1. Wählen Sie im Navigationsbereich der IAM-Konsole die Option Rollen aus. In der Konsole werden die Rollen für Ihr aktuelles Konto angezeigt.
2. Wählen Sie den Namen der Rolle, die Sie ändern möchten. Für die Rolle, die Sie ändern, sollte die AmazonRekognitionServiceRoleBerechtigungsrichtlinie gelten. Wählen Sie die Registerkarte Vertrauensbeziehungen.
3. Wählen Sie Vertrauensrichtlinie bearbeiten aus.
4. Ersetzen Sie auf der Seite Vertrauensrichtlinie bearbeiten die Standard-JSON-Richtlinie durch eine Richtlinie, die einen oder beide der globalen `aws:SourceArn`- und `aws:SourceAccount`-Bedingungskontextschlüssel verwendet. Sehen Sie sich die folgenden Beispielrichtlinien an.
5. Wählen Sie Richtlinie aktualisieren.

Die folgenden Beispiele sind Vertrauensrichtlinien, die zeigen, wie Sie die globalen `aws:SourceArn`- und `aws:SourceAccount`-Bedingungskontextschlüssel in Amazon Rekognition verwenden können, um das Problem des verwirrten Stellvertreters zu verhindern.

Wenn Sie mit gespeicherten Videos arbeiten und Videos streamen, könnten Sie in Ihrer IAM-Rolle eine Richtlinie wie die folgende verwenden:

```
{
   "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/*"
            }
         }
      }
   ]
}
```

Wenn Sie ausschließlich mit gespeicherten Videos arbeiten, könnten Sie in Ihrer IAM-Rolle eine Richtlinie wie die folgende verwenden (beachten Sie, dass Sie das `StringLike`-Argument, das den `streamprocessor` angibt, nicht angeben müssen):

```
{
   "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"
            }
         }
```

```
        }
    ]
}
```

Infrastruktursicherheit in Amazon Rekognition

Als verwalteter Service ist Amazon Rekognition durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf Amazon Rekognition zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Überwachen von Amazon Rekognition

Überwachung ist wichtig, um die Zuverlässigkeit, Verfügbarkeit und Leistung von Amazon Rekognition und Ihren anderen AWS-Lösungen aufrechtzuerhalten. AWS bietet die folgenden Überwachungstools zur Beobachtung von Rekognition, für Berichte, falls etwas nicht funktioniert, und zur automatischen Ergreifung von Gegenmaßnahmen bei Bedarf:

- Amazon CloudWatch überwacht Ihre AWS Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit. Sie können Kennzahlen erfassen und verfolgen, benutzerdefinierte Dashboards erstellen und Alarme festlegen, die Sie benachrichtigen oder Maßnahmen ergreifen, wenn eine bestimmte Metrik einen von Ihnen festgelegten Schwellenwert erreicht. Sie können beispielsweise die CPU-Auslastung oder andere Kennzahlen Ihrer EC2 Amazon-Instances CloudWatch verfolgen und bei Bedarf automatisch neue Instances starten. Weitere Informationen finden Sie im CloudWatch Amazon-Benutzerhandbuch.
- Mit Amazon CloudWatch Logs können Sie Ihre Protokolldateien von EC2 Amazon-Instances und anderen Quellen überwachen CloudTrail, speichern und darauf zugreifen. CloudWatch Logs kann Informationen in den Protokolldateien überwachen und Sie benachrichtigen, wenn bestimmte Schwellenwerte erreicht werden. Sie können Ihre Protokolldaten auch in einem sehr robusten Speicher archivieren. Weitere Informationen finden Sie im Amazon CloudWatch Logs-Benutzerhandbuch.
- Amazon EventBridge kann verwendet werden, um Ihre AWS Services zu automatisieren und automatisch auf Systemereignisse wie Probleme mit der Anwendungsverfügbarkeit oder Ressourcenänderungen zu reagieren. Ereignisse aus AWS Services werden nahezu EventBridge in Echtzeit zugestellt. Sie können einfache Regeln schreiben, um anzugeben, welche Ereignisse für Sie interessant sind und welche automatisierten Aktionen ausgeführt werden sollen, wenn ein Ereignis mit einer Regel übereinstimmt. Weitere Informationen finden Sie im EventBridge Amazon-Benutzerhandbuch.
- AWS CloudTrailerfasst API-Aufrufe und zugehörige Ereignisse, die von oder im Namen Ihres AWS Kontos getätigt wurden, und übermittelt die Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket. Sie können feststellen, welche Benutzer und Konten angerufen wurden AWS, von welcher Quell-IP-Adresse aus die Anrufe getätigt wurden und wann die Aufrufe erfolgten. Weitere Informationen finden Sie im AWS CloudTrail -Benutzerhandbuch.

Überwachung von Rekognition mit Amazon CloudWatch

Mit CloudWatch können Sie Metriken für einzelne Rekognition-Operationen oder globale Rekognition-Metriken für Ihr Konto abrufen. Sie können Metriken verwenden, um den Zustand Ihrer auf Rekognition basierenden Lösung zu verfolgen, und Alarme einrichten, die Sie benachrichtigen, wenn eine oder mehrere Metriken einen definierten Schwellenwert überschreiten. Beispielsweise können Sie Metriken für die Anzahl aufgetretener Serverfehler oder Metriken für die Anzahl erkannter Gesichter anzeigen. Außerdem können Sie Metriken für die Anzahl von erfolgreichen spezifischen Rekognition-Operationen anzeigen. Um Metriken zu sehen, können Sie Amazon CloudWatch AWS Command Line Interface, Amazon oder die CloudWatch API verwenden.

Sie können auch Gesamtmetriken für einen ausgewählten Zeitraum anzeigen, indem Sie die Rekognition-Konsole nutzen. Weitere Informationen finden Sie unter Übung 4: Anzeigen von Gesamtmetriken (Konsole).

CloudWatch Metriken für Rekognition verwenden

Um Metriken zu verwenden, müssen Sie die folgenden Informationen angeben:

- Die Metrikdimension oder keine Dimension. Eine Dimension ist ein Name-Wert-Paar, mit dem Sie eine Metrik eindeutig identifizieren. Rekognition verfügt über eine Dimension namens Operation. Es stellt Metriken für eine bestimmte Operation zur Verfügung. Wenn Sie keine Dimension angeben, gilt die Metrik für alle Rekognition-Operationen innerhalb Ihres Kontos.
- Der Metrikname, beispielsweise `UserErrorCount`.

Sie können Überwachungsdaten für Rekognition mithilfe der AWS Management Console, der oder der AWS CLI CloudWatch API abrufen. Sie können die CloudWatch API auch über eines der Amazon AWS Software Development Kits (SDKs) oder die CloudWatch API-Tools verwenden. Die Konsole zeigt eine Reihe von Diagrammen an, die auf den Rohdaten der CloudWatch API basieren. Je nach Anforderungen können Sie entweder die in der Konsole angezeigten oder die mit der API aufgerufenen Graphen verwenden.

In der folgenden Liste finden Sie einige häufige Verwendungszwecke für die Metriken. Es handelt sich dabei um Vorschläge für den Einstieg und nicht um eine umfassende Liste.

Wie gehe ich vor?	Relevante Metriken
Wie kann ich ermitteln, wie viele Gesichter erkannt wurden?	Überwachen Sie die Sum-Statistik der `DetectedFaceCount` -Metrik.
Wie kann ich erkennen, ob meine Anwendung die maximale Anzahl an Anforderungen pro Sekunde erreicht hat?	Überwachen Sie die Sum-Statistik der `ThrottledCount` -Metrik.
Wie überwache ich die Anforderungsfehler?	Verwenden Sie die Sum-Statistik der `UserErrorCount` -Metrik.
Wie finde ich die Gesamtanzahl der Anforderungen?	Verwenden Sie die `ResponseTime` - und `Data Samples`-Statistik der `ResponseTime` -Metrik. Dies umfasst jegliche Anforderung, die zu einem Fehler geführt hat. Um nur erfolgreiche Operationsaufrufe anzuzeigen, verwenden Sie die`SuccessfulRequestCount` -Metrik.
Wie überwache ich die Latenz der `Rekognition` -Operationsaufrufe?	Verwenden Sie die `ResponseTime` -Metrik.
Wie überwache ich, wie oft `IndexFaces` erfolgreich Gesichter zu den Rekognition-Sammlungen hinzugefügt hat?	Überwachen Sie die Sum-Statistik der `SuccessfulRequestCount` -Metrik und die `IndexFaces` -Operation. Verwenden Sie die `Operation` -Dimension, um die Operation und die Metrik auszuwählen.

Sie müssen über die entsprechenden CloudWatch Berechtigungen verfügen, um Rekognition zu überwachen. CloudWatch Weitere Informationen finden Sie unter Identity and Access Management for Amazon CloudWatch.

Zugreifen auf Rekognition-Metriken

Die folgenden Beispiele zeigen, wie Sie mit der CloudWatch Konsole, der und der API auf Rekognition-Metriken zugreifen können. AWS CLI CloudWatch

So zeigen Sie Metriken an (Konsole)

1. Öffnen Sie die CloudWatch Konsole unter. https://console.aws.amazon.com/cloudwatch/
2. Wählen Sie Metriken, dann die Registerkarte Alle Metriken und dann Rekognition.
3. Wählen Sie Metriken ohne Dimensionen, und dann eine Metrik.

   Wählen Sie beispielsweise die DetectedFace-Metrik aus, um zu prüfen, wie viele Gesichter erkannt wurden.
4. Wählen Sie einen Wert für den Datumsbereich aus. Die Metrikanzahl, die im Graph angezeigt wird.

Um Metriken anzuzeigen, müssen über einen bestimmten Zeitraum regelmäßig erfolgreiche Aufrufe an die **`DetectFaces`**-Operation gemacht worden sein (CLI).

- Öffnen Sie den AWS CLI und geben Sie den folgenden Befehl ein:

  ```
  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
  ```

  Dieses Beispiel zeigt die erfolgreichen Aufrufe der `DetectFaces`-Operation für einen bestimmten Zeitraum an. Weitere Informationen finden Sie unter get-metric-statistics.

Um auf Metriken (CloudWatch API) zuzugreifen

- Rufen Sie die folgende Seite auf `GetMetricStatistics`. Weitere Informationen finden Sie in der Amazon CloudWatch API-Referenz.

Alarm erstellen

Sie können einen CloudWatch Alarm erstellen, der eine Amazon Simple Notification Service (Amazon SNS) -Nachricht sendet, wenn sich der Status des Alarms ändert. Ein Alarm überwacht eine Metrik über einen bestimmten, von Ihnen definierten Zeitraum und führt eine oder mehrere Aktionen durch, die vom Wert der Metrik im Vergleich zu einem festgelegten Schwellenwert in einer Reihe von

Zeiträumen abhängt. Die Aktion ist eine Benachrichtigung, die an ein Amazon-SNS-Thema oder eine Auto-Scaling-Richtlinie gesendet wird.

Alarme lösen nur Aktionen für anhaltende Statusänderungen aus. CloudWatch Alarme lösen keine Aktionen aus, nur weil sie sich in einem bestimmten Zustand befinden. Der Status muss sich geändert haben und für eine festgelegte Anzahl an Zeiträumen aufrechterhalten worden sein.

So richten Sie einen Alarm ein (Konsole)

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie Alarm erstellen aus. Dadurch wird der Assistent zum Erstellen von Alarmen gestartet.
3. Wählen Sie aus der Liste der Metriken Metriken ohne Dimensionen den Eintrag Rekognition-Metriken aus, und wählen Sie dann eine Metrik.

   Wählen Sie beispielsweise DetectedFaceCount aus, um einen Alarm für eine maximale Anzahl an erkannten Gesichtern einzurichten.
4. Wählen Sie im Bereich Zeitbereich einen Datumsbereich-Wert aus, der die Operationen zur Gesichtserkennung beinhaltet, die Sie aufgerufen haben. Wählen Sie Weiter
5. Geben Sie Name und Description an. Für Wann immer wählen Sie >= aus und geben einen maximalen Wert Ihrer Wahl an.
6. Wenn Sie Ihnen eine E-Mail senden CloudWatch möchten, wenn der Alarmstatus erreicht ist, wählen Sie für Wann immer dieser Alarm: die Option Status ist ALARM. Um Alarme für ein bestehendes Amazon-SNS-Thema zu senden, wählen Sie für Benachrichtigung senden an: ein bestehendes SNS-Thema aus. Um den Namen und die E-Mail-Adressen für eine neue E-Mail-Abonnementliste festzulegen, wählen Sie Thema erstellen, CloudWatch speichert die Liste und zeigt sie im Feld an, sodass Sie sie für future Alarme verwenden können.

   > **Note**
   >
   > Wenn Sie Thema erstellen nutzen, um ein neues Amazon-SNS-Thema einzurichten, müssen die E-Mail-Adressen überprüft werden, bevor die gewünschten Empfänger Benachrichtigungen erhalten. Amazon SNS sendet nur dann eine E-Mail, wenn der Alarm einen Alarmzustand auslöst. Wenn es zu dieser Änderung des Alarmzustands kommt, bevor die E-Mail-Adressen überprüft wurden, erhalten die vorgesehenen Empfänger keine Benachrichtigung.

7. Nutzen Sie die Alarmvorschau im Bereich Alarm Preview. Wählen Sie Alarm erstellen aus.

So richten Sie einen Alarm ein (AWS CLI)

- Öffnen Sie den AWS CLI und geben Sie den folgenden Befehl ein. Ändern Sie den Wert des `alarm-actions`-Parameters als Verweis auf ein Amazon-SNS-Thema, das Sie zuvor erstellt haben.

  ```
  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
  ```

  Dieses Beispiel zeigt, wie Sie einen Alarm für den Fall erstellen, dass innerhalb von 5 Minuten mehr als 10 Benutzerfehler auftreten. Weitere Informationen finden Sie unter put-metric-alarm.

Um einen Alarm einzustellen (CloudWatch API)

- Rufen Sie die folgende Seite auf `PutMetricAlarm`. Weitere Informationen finden Sie unter Amazon CloudWatch API-Referenz.

CloudWatchMetriken für Rekognition

Dieser Abschnitt enthält Informationen zu den CloudWatch Amazon-Metriken und der Operation-Dimension, die für Amazon Rekognition verfügbar sind.

Sie können auch eine Gesamtansicht der Rekognition-Metriken aus der Amazon-Rekognition-Konsole sehen. Weitere Informationen finden Sie unter Übung 4: Anzeigen von Gesamtmetriken (Konsole).

CloudWatch Metriken für Rekognition

In der folgenden Tabelle finden Sie eine Übersicht über die Rekognition-Metriken.

Metrik	Beschreibung
SuccessfulRequestCount	die Anzahl erfolgreicher Anforderungen. Der Antwortcode-Bereich für eine erfolgreiche Anforderung ist 200 bis 299. Einheit: Anzahl Gültige Statistiken: `Sum,Average`
ThrottledCount	Die Anzahl der gedrosselten Anforderungen. Rekognition drosselt eine Anforderung, wenn mehr Anforderungen eingehen, als das Transakti onslimit pro Sekunde für das Konto erlaubt. Wenn der Grenzwert für Ihr Konto häufig überschritten wird, können Sie eine Erweiterung des Limits beantragen. Informationen zum Anfordern einer Erweiterung finden Sie unter AWS Service Limits. Einheit: Anzahl Gültige Statistiken: `Sum,Average`
ResponseTime	Die Zeit in Millisekunden, in der Rekognition die Antwort berechnet. Einheiten: 1. Anzahl für `Data Samples`-Statistiken 2. Millisekunden für die `Average` Statistik Gültige Statistiken: `Data Samples,Average` **Note** Die `ResponseTime` -Metrik ist nicht im Rekognition-Metrik bereich enthalten.
DetectedFaceCount	Die Anzahl von Gesichtern, die mithilfe der `IndexFaces` - oder der `DetectFaces` -Operation erkannt wurden. Einheit: Anzahl

Metrik	Beschreibung
	Gültige Statistiken: `Sum,Average`
DetectedLabelCount	Die Anzahl von Labels, die mithilfe der `DetectLabels` -Operation erkannt wurden. Einheit: Anzahl Gültige Statistiken: `Sum,Average`
ServerErrorCount	Die Anzahl von Server-Fehlern. Der Antwortcode für einen Serverfehler liegt zwischen 500 und 599. Einheit: Anzahl Gültige Statistiken: `Sum, Average`
UserErrorCount	Die Anzahl der Benutzerfehler (ungültige Parameter, ungültiges Bild, keine Berechtigung, usw.). Der Antwortcode-Bereich für einen Benutzerf ehler ist 400 bis 499. Einheit: Anzahl Gültige Statistiken: `Sum,Average`
MinInferenceUnits	Die Mindestanzahl von Inferenzeinheiten, die während der `StartProj ectVersion` -Anforderung angegeben wurde. Einheit: Anzahl Gültige Statistiken: `Average`
MaxInferenceUnits	Die Maximalanzahl von Inferenzeinheiten, die während der `StartProj ectVersion` -Anforderung angegeben wurde. Einheit: Anzahl Gültige Statistiken: `Average`

Metrik	Beschreibung
DesiredInferenceUnits	Die Anzahl der Inferenzeinheiten, auf die Rekognition nach oben oder unten skaliert. Einheit: Anzahl Gültige Statistiken: `Average`
InServiceInferenceUnits	Die Anzahl der Inferenzeinheiten, die das Modell verwendet. Einheit: Anzahl Gültige Statistiken: `Average` Es wird empfohlen, die Durchschnittsstatistik zu verwenden, um den 1-Minuten-Durchschnitt der verwendeten Instances zu ermitteln.

CloudWatch Metriken für Rekognition Streaming

Rekognition hat auch einen zweiten Namespace, der für Streaming-Operationen verwendet wird, „Rekognition-Streaming“. In der folgenden Tabelle finden Sie eine Zusammenfassung der Rekognition-Streaming-Metriken.

Metrik	Beschreibung
SuccessfulRequestCount	die Anzahl erfolgreicher Anforderungen. Der Antwortcode-Bereich für eine erfolgreiche Anforderung ist 200 bis 299. Einheit: Anzahl Gültige Statistiken: `Sum,Average`
CallCount	Die Anzahl der angegebenen Operationen, die in Ihrem Konto ausgeführt werden. Gültige Statistiken: `Sum,Average`
ThrottledCount	Die Anzahl der gedrosselten Anforderungen. Rekognition drosselt eine Anforderung, wenn mehr Anforderungen eingehen, als das Transakti

Metrik	Beschreibung
	onslimit pro Sekunde für das Konto erlaubt. Wenn der Grenzwert für Ihr Konto häufig überschritten wird, können Sie eine Erweiterung des Limits beantragen. Informationen zum Anfordern einer Erweiterung finden Sie unter AWS Service Limits. Einheit: Anzahl Gültige Statistiken: `Sum,Average`
ServerErrorCount	Die Anzahl von Server-Fehlern. Der Antwortcode für einen Serverfehler liegt zwischen 500 und 599. Einheit: Anzahl Gültige Statistiken: `Sum, Average`
UserErrorCount	Die Anzahl der Benutzerfehler (ungültige Parameter, ungültiges Bild, keine Berechtigung, usw.). Der Antwortcode-Bereich für einen Benutzerf ehler ist 400 bis 499. Einheit: Anzahl Gültige Statistiken: `Sum,Average`
ConcurrentSessions	Die Anzahl der gleichzeitigen Sitzungen, die Ihr Konto derzeit verwendet. Einheit: Anzahl Gültige Statistiken: `Max,Average`

CloudWatch Dimension für Rekognition

Um operationsspezifische Metriken aufzurufen, verwenden Sie den `Rekognition`-Namespace und geben Sie eine Operationsdimension an.

Weitere Informationen zu Abmessungen finden Sie unter Abmessungen im CloudWatch Amazon-Benutzerhandbuch.

CloudWatch Dimension für Rekognition Custom Labels

In der folgenden Tabelle sind die CloudWatch Abmessungen aufgeführt, die für die Verwendung mit Rekognition Custom Labels verfügbar sind:

Dimension	Beschreibung
ProjectName	Der Name des Rekognition-Custom-Labels-Projekts, mit dem Sie `CreateProject` erstellt haben.
VersionName	Der Name der Rekognition-Custom-Labels-Projektversion, mit der Sie `CreateProjectVersion` erstellt haben.

Weitere Informationen zu Abmessungen finden Sie unter Abmessungen im CloudWatch Amazon-Benutzerhandbuch.

Protokollieren Amazon Rekognition Rekognition-API-Aufrufen mit AWS CloudTrail

Amazon Rekognition ist in einen Service integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen eines Benutzers, einer Rolle oder eines AWS Dienstes in Amazon Rekognition bereitstellt. CloudTrail erfasst alle API-Aufrufe für Amazon Rekognition als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der Amazon-Rekognition-Konsole und Code-Aufrufe an die Amazon-Rekognition-API-Vorgänge. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Übermittlung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für Amazon Rekognition. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf einsehen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an Amazon Rekognition gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen CloudTrail dazu finden Sie im AWS CloudTrail Benutzerhandbuch.

Informationen zu Amazon Rekognition in CloudTrail

CloudTrail ist für Ihr AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn in Amazon Rekognition eine Aktivität auftritt, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der CloudTrail Ereignishistorie in einem Ereignis aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem AWS Konto ansehen, suchen und herunterladen. Weitere Informationen finden Sie unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung von Ereignissen in Ihrem AWS Konto, einschließlich Ereignissen für Amazon Rekognition, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Pfad in der Konsole anlegen, gilt dieser für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- CloudTrail Unterstützte Dienste und Integrationen
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Alle Amazon Rekognition-Aktionen werden von der Amazon Rekognition API-Referenz protokolliert CloudTrail und sind in dieser dokumentiert. Beispielsweise generieren Aufrufe von `CreateStreamProcessor` und `DetectCustomLabels` Aktionen Einträge in den `CreateCollection` Protokolldateien. CloudTrail

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root- oder AWS Identity and Access Management (IAM-) Benutzeranmeldedaten gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie unter CloudTrail userIdentity-Element.

Erläuterungen zu Amazon-Rekognition-Protokolldateieinträgen

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag mit Aktionen für die folgende API: `StartLabelDetection` und`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"
        }
    ]
}
```

Richtlinien und Kontingente in Amazon Rekognition

Die folgenden Abschnitte enthalten Richtlinien und Kontingente bei der Verwendung von Amazon Rekognition. Es gibt zwei Arten von Kontingenten. Festgelegte Kontingente, wie die maximale Bildgröße, können nicht geändert werden. Die auf der Seite AWS Service Quotas aufgeführten Standardkontingente können geändert werden, indem Sie das im Abschnitt Standardkontingente beschriebene Verfahren befolgen.

Themen

- Unterstützte -Regionen
- Festgelegte Kontingente
- Standardkontingente

Unterstützte -Regionen

Eine Liste der AWS Regionen, in denen Amazon Rekognition verfügbar ist, finden Sie unter AWS-Regionen und Endpunkte in der Amazon Web Services General Reference.

Festgelegte Kontingente

Im Folgenden finden Sie eine Liste von Limits in Amazon Rekognition, die nicht geändert werden können. Informationen über Limits, die Sie ändern können, wie z. B. Limits für Transaktionen pro Sekunde (TPS), finden Sie unter Standardkontingente.

Die Beschränkungen für Amazon Rekognition Custom Labels finden Sie unter Richtlinien und Kontingente in Amazon Rekognition Custom Labels.

Amazon Rekognition Image

- Die Bildgröße für die Speicherung als Amazon-S3-Objekt ist auf 15 MB begrenzt.
- Die maximale Bildgröße für `DetectModerationLabels` beträgt 10.000 Pixel sowohl für die Breite als auch für die Höhe.
- Die maximale Bildgröße für `DetectLabels` beträgt 10.000 Pixel sowohl für die Breite als auch für die Höhe.

- Um erkannt zu werden, darf ein Gesicht nicht kleiner sein als 40 x 40 Pixel in einem Bild mit 1920 x 1080 Pixeln. Bilder mit Maßen über 1920 x 1080 Pixeln benötigen proportional eine höhere Mindestgröße für das Gesicht.
- Die Mindestbildgröße für Höhe und Breite ist 80 Pixel. Die Mindestbildgröße für `DetectProtectiveEquipment` für Höhe und Breite ist 64 Pixel.
- Die maximale Bildgröße für `DetectProtectiveEquipment` beträgt 4.096 Pixel sowohl für die Breite als auch für die Höhe.
- Um von `DetectProtectiveEquipment` erkannt zu werden, darf eine Person nicht kleiner als 100x100 Pixel in einem Bild mit 800x1300 sein. Bilder mit Maßen über 800x1300 Pixeln benötigen proportional eine höhere Mindestgröße für die Person.
- Die maximale Bildgröße für unformatierte Bytes, die als Parameter an eine API übermittelt werden, ist 5 MB. Das Limit für die `DetectProtectiveEquipment`-API beträgt 4 MB.
- Amazon Rekognition unterstützt die Bildformate PNG und JPEG. Genauer gesagt: Die Bilder, die Sie als Eingabe für verschiedene API-Operationen wie `DetectLabels` und `IndexFaces` bereitstellen, müssen in einem der unterstützten Formate vorliegen.
- Die maximale Anzahl von Gesichtsvektoren, die Sie in einer einzelnen Gesichtersammlung speichern können, ist 20 Millionen.
- Die maximale Anzahl von Benutzervektoren, die Sie in einer einzelnen Gesichtssammlung speichern können, beträgt standardmäßig 10 Millionen.
- Die maximale Anzahl an übereinstimmenden Gesichtsvektoren, die die Such-API zurückgibt, ist 4.096.
- Die maximale Anzahl an übereinstimmenden Benutzervektoren, die die Such-API zurückgibt, ist 4.096.
- `DetectText` kann bis zu 100 Wörter in ein Bild erkennen.
- `DetectProtectiveEquipment` kann persönliche Schutzausrüstung von bis zu 15 Personen erkennen.

Informationen zu bewährten Methoden für Bilder und den Gesichtsvergleich finden Sie unter Bewährte Methoden für Sensoren, Eingabebilder und Videos.

Amazon Rekognition Bild-Massenanalyse

- Amazon Rekognition Image Bulk Analysis kann Bildstapel mit einer Größe von bis zu 10.000 Bildern analysieren.

- Amazon Rekognition Image Bulk Analysis unterstützt Eingabemanifeste mit einer Größe von bis zu 50 MB.

Gespeichertes Video von Amazon Rekognition Video

- Amazon Rekognition Video kann gespeicherte Videos mit einer Größe von bis zu 10 GB analysieren.
- Amazon Rekognition Video kann gespeicherte Videos von bis zu 6 Stunden Länge analysieren.
- Amazon Rekognition Video unterstützt maximal 20 gleichzeitige Aufträge pro Konto.
- Gespeicherte Videos müssen mit dem H.264-Codec codiert sein. Die unterstützten Dateiformate sind MPEG-4 und MOV.
- Jede Amazon-Rekognition-Video-API, die Audiodaten analysiert, unterstützt nur AAC-Audiocodecs.
- Die Gültigkeitsdauer (Time to Live, TTL) für Paginierungs-Token beträgt 24 Stunden. Paginierungs-Token befinden sich im `NextToken`-Feld, das von Get-Operationen wie `GetLabeldetection` zurückgegeben wird.

Streaming-Video von Amazon Rekognition Video

- Ein Kinesis-Video-Eingabestrom kann maximal einem Amazon-Rekognition-Video-Stromprozessor zugeordnet werden.
- Ein Kinesis-Data-Ausgabestrom kann maximal einem Amazon-Rekognition-Video-Stromprozessor zugeordnet werden.
- Der Kinesis-Video-Eingabestrom und der Kinesis-Data-Ausgabestrom können einem Amazon-Rekognition-Video-Stromprozessor zugeordnet werden, jedoch nicht zwischen mehreren Prozessoren geteilt werden.
- Jede Amazon-Rekognition-Video-API, die Audiodaten analysiert, unterstützt nur ACC-Audiocodecs.

Standardkontingente

Eine Liste der Standardkontingente finden Sie unter AWS Service Quotas. Diese Standardlimits können geändert werden. Um eine Erhöhung des Limits zu beantragen, erstellen Sie einen Fall. Ihre aktuellen Kontingentlimits (angewandte Kontingentwerte) finden Sie unter Amazon Rekognition Service Quotas. Um Ihren TPS-Nutzungsverlauf für Amazon Rekognition Image einzusehen APIs,

besuchen Sie die Seite Amazon Rekognition Service Quotas und wählen Sie einen bestimmten API-Vorgang aus, um den Verlauf für diesen Vorgang zu sehen.

Themen

- Berechnen der TPS-Kontingentänderung
- Bewährte Methoden für TPS-Kontingente
- Erstellen Sie einen Fall, um TPS-Kontingente zu ändern

Berechnen der TPS-Kontingentänderung

Was ist das neue Limit, das Sie beantragen? Transaktionen pro Sekunde (TPS) sind auf dem Höhepunkt einer zu erwartenden Workload am relevantesten. Es ist wichtig, die maximale Anzahl gleichzeitiger API-Aufrufe zu Spitzenzeiten einer Workload und die Antwortzeit (5–15 Sekunden) zu kennen. Bitte beachten Sie, dass mindestens 5 Sekunden erforderlich sind. Nachfolgend finden Sie zwei Beispiele:

- Beispiel 1: Die maximale Anzahl gleichzeitiger Face Authentication (CompareFaces API) - Benutzer, die ich zu Beginn meiner geschäftigsten Stunde erwarte, liegt bei 1000. Diese Antworten werden über einen Zeitraum von 10 Sekunden verteilt. Daher sind für die CompareFaces API in meiner jeweiligen Region 100 (1000/10) TPS erforderlich.
- Beispiel 2: Die maximale Anzahl gleichzeitiger DetectLabels API-Aufrufe (Object Detection), die zu Beginn meiner geschäftigsten Zeit erwartet werden, beträgt 250. Diese Antworten werden über einen Zeitraum von 5 Sekunden verteilt. Daher ist für die DetectLabels API in meiner jeweiligen Region ein TPS-Wert von 50 (250/5) erforderlich.

Bewährte Methoden für TPS-Kontingente

Zu den empfohlenen bewährten Methoden für Transaktionen pro Sekunde (TPS) gehören die Glättung von hohem Datenverkehr, die Konfiguration von Wiederholungsversuchen und die Konfiguration von exponentiellem Backoff und Jitter.

1. Reibungsloser hoher Datenverkehr. Hoher Datenverkehr beeinträchtigt den Durchsatz. Um den maximalen Durchsatz für die zugewiesenen Transaktionen pro Sekunde (TPS) zu erreichen, verwenden Sie eine serverlose Warteschlangenarchitektur oder einen anderen Mechanismus, um den Datenverkehr zu „glätten“, sodass er konsistenter ist. Codebeispiele und Referenzen für die

serverlose groß angelegte Bild- und Videoverarbeitung mit Rekognition finden Sie unter Bild- und Videoverarbeitung im großen Maßstab mit Amazon Rekognition.

2. Konfigurieren Sie Wiederholungsversuche. Folgen Sie den Richtlinien unter the section called "Fehlerbehandlung", um Wiederholungsversuche für die Fehler zu konfigurieren, die sie zulassen.
3. Konfigurieren Sie exponentielles Backoff und Jitter. Durch die Konfiguration von exponentiellem Backoff und Jitter bei der Konfiguration von Wiederholungsversuchen können Sie den erreichbaren Durchsatz verbessern. Weitere Informationen finden Sie unter Wiederholungen bei Fehlern und exponentieller Backoff in. AWS

Erstellen Sie einen Fall, um TPS-Kontingente zu ändern

Um einen Fall zu erstellen, gehen Sie zu Fall erstellen und beantworten Sie die folgenden Fragen:

- Haben Sie die the section called "Bewährte Methoden für TPS-Kontingente" implementiert, um Ihre Datenverkehrsspitzen zu glätten und Wiederholungsversuche, exponentiellen Backoff und Jitter zu konfigurieren?
- Haben Sie die Änderung des TPS-Kontingents berechnet, die Sie benötigen? Falls nicht, siehe the section called "Berechnen der TPS-Kontingentänderung".
- Haben Sie Ihre TPS-Nutzungshistorie überprüft, um Ihren künftigen Bedarf genauer vorhersagen zu können? Ihren TPS-Nutzungsverlauf finden Sie auf der Seite Amazon Rekognition Service Quotas.
- Was ist Ihr Anwendungsfall?
- Was planen APIs Sie zu verwenden?
- In welchen Regionen planen Sie, diese APIs zu verwenden?
- Können Sie die Last auf mehrere Regionen verteilen?
- Wie viele Bilder verarbeiten Sie täglich?
- Wie lange wird dieses Volumen voraussichtlich aufrechterhalten (handelt es sich um einen einmaligen Anstieg oder um einen andauernden Anstieg)?
- Wie werden Sie durch das Standardlimit blockiert? Sehen Sie sich die folgende Ausnahmetabelle an, um zu überprüfen, auf welches Szenario Sie stoßen.

Fehlercode	Exception	Fehlermeldung	Was bedeutet das?	Kann es erneut getestet werden?
HTTP-Statuscode 400	ProvisionedThroughputExceededException	Provisioned Rate exceeded.	Zeigt Drosselung an. Sie können eine Anforderung zur Erhöhung des Limits erneut versuchen oder auswerten.	Ja
HTTP-Statuscode 400	ThrottlingException	Verlangsamung; plötzlicher Anstieg der Anforderungsgeschwindigkeit.	Möglicherweise senden Sie spitzen Datenverkehr und stoßen auf Drosselung. Sie sollten den Datenverkehr flüssiger und konsistenter gestalten. Konfigurieren Sie dann zusätzliche Wiederholungen. Siehe bewährte Methoden.	Ja

Fehlercode	Exception	Fehlermeldung	Was bedeutet das?	Kann es erneut getestet werden?
HTTP-Statuscode 5xx	ThrottlingException (HTTP 500)	Service nicht verfügbar	Zeigt an, dass das Backend hochskaliert wird, um die Aktion zu unterstützen. Sie sollten die Anforderung wiederholen.	Ja

Ein detailliertes Verständnis der Fehlercodes finden Sie unter the section called “Fehlerbehandlung”.

> **Note**
>
> Diese Grenzwerte hängen von der Region ab, in der Sie sich befinden. Wenn Sie einen Fall zum Ändern eines Limits erstellen, wirkt sich die von Ihnen angeforderte API-Operation in der Region aus, in der Sie sie anfordern. Andere API-Operationen und Regionen sind nicht betroffen.

Dokumentverlauf für Amazon Rekognition

In der folgenden Tabelle sind die wichtigen Änderungen in jeder Version des Amazon-Rekognition-Entwicklerleitfadens beschrieben. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie einen RSS-Feed abonnieren.

- Letzte Aktualisierung der Dokumentation: 15. Juni 2023

Änderung	Beschreibung	Datum
Amazon Rekognition unterstützt jetzt neue Moderationsbezeichnungen und verbesserte Genauigkeit bei der Inhaltsmoderation von Bildern	Die Inhaltsmoderationsfunktion von Amazon Rekognition wurde verbessert, um die Genauigkeit zu verbessern, neue Labels zu erkennen und animierte und/oder illustrierte Inhalte zu identifizieren.	1. Februar 2024
Amazon Rekognition unterstützt jetzt die Massenbildanalyse	Amazon Rekognition unterstützt jetzt die asynchrone Verarbeitung einer großen Sammlung von Bildern mithilfe einer Manifestdatei für den Vorgang. StartMediaAnalysisJob	23. Oktober 2023
Amazon Rekognition unterstützt jetzt benutzerdefinierte Inhaltsmoderation mit Adaptern	Amazon Rekognition unterstützt jetzt eine verbesserte Genauigkeit der DetectModerationLabels API durch die Verwendung von Adaptern, die die Funktionen vorhandener Rekognition-Deep-Learning-Modelle erweitern.	12. Oktober 2023

Rekognition unterstützt jetzt Benutzervektoren mit Sammlungen	Rekognition-Gesichtssammlungen unterstützen jetzt die Erstellung von Benutzervektoren. Benutzervektoren aggregieren mehrere Gesichtsvektoren desselben Benutzers und verbessern so die Genauigkeit durch robustere Darstellungen eines Benutzers.	12. Juni 2023
Zu den folgenden verwalteten Richtlinien wurden Aktionen hinzugefügt, mit denen verwaltete Benutzer einbezogen werden können: AmazonRekognitionReadOnlyAccess	Amazon Rekognition hat die folgenden Aktionen zu den `AmazonRekognitionReadOnlyAccess` - verwalteten Richtlinien hinzugefügt: `ListUsers` , `SearchUsers` , `SearchUsersByImage`	12. Juni 2023
Amazon Rekognition Image kann jetzt die Blickrichtung ableiten	Die Gesichtserkennungsfunktionen von Amazon Rekognition Image wurden verbessert, sodass nun die Blickrichtung in ein erkanntes Gesicht abgeleitet werden kann.	31. Mai 2023

Die API zur Moderation von Rekognition-Inhalten wurde verbessert	Rekognition verbesserte das Modell der Inhaltsmoderation für die Bild- und Videomoderation. Durch die Verbesserung wurde die Erkennung von expliziten, gewalttätigen und suggestiven Inhalten erheblich erweitert. Kunden können nun explizite und gewalttätige Inhalte mit höherer Genauigkeit erkennen, um das Benutzererlebnis zu verbessern, ihre Markenidentität zu schützen und sicherzustellen, dass alle Inhalte ihren Branchenvorschriften und -richtlinien entsprechen.	09. Mai 2023
Amazon Rekognition Image kann jetzt verdeckte Gesichter erkennen	Amazon Rekognition Image kann jetzt die Okklusion von Gesichtern erkennen. Ein neues FaceOccluded Attribut wird von Amazon Rekognition Images zurückgegeben DetectFaces und gibt an IndexFaces APIs, ob das Gesicht in einem Bild aufgrund von überlappenden Objekten, Kleidung und Körperteilen teilweise oder nicht vollständig sichtbar ist.	5. Mai 2023

Rekognition kann jetzt die Echtheit von Gesichtern erkennen	Amazon Rekognition Video kann jetzt verwendet werden, um die Echtheit eines Videos zu erkennen und zu überprüfen, ob ein Benutzer vor einer Kamera physisch anwesend ist. Der Face-Liveness-Detektor erkennt auch gefälschte Angriffe, die auf eine Kamera gerichtet sind oder versuchen, eine Kamera zu umgehen.	11. April 2023
Aktualisieren Sie auf Amazon Rekognition Video.	Amazon Rekognition Video kann jetzt mehr Labels erkennen und mehr Informationen zu den Attributen von Bildern und Labels zurückgeben. Die GetLabelDetection API gibt jetzt Informationen über Aliase und Kategorien zurück. Zurückgegebene Labelinformationen können mit inklusiven und exklusiven Filteroptionen gefiltert werden. Die Ergebnisse können nach Zeitstempeln oder Videosegmenten aggregiert werden.	7. Dezember 2022

Aktualisieren Sie auf Amazon Rekognition Image.	Amazon Rekognition Image kann jetzt mehr Labels erkennen und gibt jetzt mehr Informationen zu den Attributen von Bildern und Labels zurück. Die DetectLabels API gibt jetzt Informationen über Aliase, Kategorien und Bildeigenschaften wie dominante Farben zurück. Zurückgegebene Labelinformationen können mit inklusiven und exklusiven Filteroptionen gefiltert werden.	11. November 2022
Die folgenden verwalteten Richtlinien wurden um Aktionen für ProjectPolicy und das Kopieren von Modellen mit benutzerdefinierten Labels erweitert: AmazonRekognitionReadOnlyAccess	Amazon Rekognition hat die folgenden Aktionen zu der `AmazonRekognitionReadOnlyAccess` -verwalteten Richtlinie hinzugefügt:`ListProjectPolicies`	21. Juli 2022
Die folgenden verwalteten Richtlinien wurden um Aktionen für ProjectPolicy und das Modell „Benutzerdefiniertes Labels“ -Modell erweitert: AmazonRekognitionFullAccessAmazonRekognitionCustomLabelsFullAccess	Rekognition hat die folgenden Aktionen zu den `AmazonRekognitionCustomLabelsFullAccess` - und `AmazonRekognitionFullAccess` -verwalteten Richtlinien hinzugefügt: `CopyProjectVersion` , `PutProjectPolicy` , `ListProjectPolicies` , `DeleteProjectPolicy`	21. Juli 2022

Amazon Rekognition Video kann jetzt Labels in Streaming-Videos erkennen	Amazon Rekognition Video kann Label wie Haustiere und Pakete in Streaming-Videos erkennen. Dies erfolgt mithilfe von `ConnectedHome` -Einstellungen auf Stromprozessoren, die im Rahmen der `CreateStreamProcessor` -Operation erstellt wurden.	28. April 2022
Die API-Referenz wurde aus dem Amazon-Rekognition-Entwicklerhandbuch entfernt	Die Amazon-Rekognition-API-Referenz ist jetzt unter Amazon-Rekognition-API-Referenz verfügbar.	24. Februar 2022
Update zur Datensatzverwaltung für die folgenden verwalteten Richtlinien: Von AWS verwaltete Richtlinie: AmazonRekognitionReadOnlyAccess,Von AWS verwaltete Richtlinie: AmazonRekognitionFullAccess,Von AWS verwaltete Richtlinie: AmazonRekognitionCustomLabelsFullAccess	Amazon Rekognition hat den`AmazonRekognitionReadOnlyAccess` ,, `AmazonRekognitionFullOnlyAccess` und den `AmazonRekognitionCustomLabelsFullAccess` verwalteten Richtlinien die folgenden Aktionen hinzugefügt:`CreateDataset` ,,`ListDatasetEntries` ,`ListDatasetLabels` ,`DescribeDataset` , `UpdateDatasetEntries` `DistributeDatasetEntries` DeleteDataset	1. November 2021

Ein neuer Knoten im Inhaltsverzeichnis zeigt Amazon Rekognition Rekognition-Beispiele, die gehostet werden auf GitHub	Aktualisierte Codebeispiele aus dem AWS Code Examples Repository werden jetzt in einem separaten Knoten im Amazon Rekognition Rekognition-Entwicklerhandbuch angezeigt, um den Zugriff zu erleichtern.	22. Oktober 2021
Amazon Rekognition kann schwarze Frames und Hauptprogramminhalte in Videosegmenten erkennen	Amazon Rekognition kann mithilfe der `StartSegmentDetection` - und `GetSegmentDetection` -Operationen schwarze Frames, Farbbalken, Vorspann, Abspann, Studiologos und Hauptprogramminhalte als technische Hinweise in einem Video identifizieren.	7. Juni 2021
Update zur Datensatzverwaltung für die folgenden verwalteten Richtlinien:	Sie können die `DetectText` -Operation von Amazon Rekognition verwenden, um bis zu 100 Wörter in einem Bild zu erkennen.	21. Mai 2021
Tagging-Update für und AmazonRekognitionReadOnlyAccessAmazonRekognitionFullAccess	Rekognition hat den `AmazonRekognitionFullAccess` - und `AmazonRekognitionReadOnlyAccess` -Richtlinien und neue Tagging-Aktionen hinzugefügt.	2. April 2021

Amazon Rekognition unterstützt jetzt Tagging	Sie können jetzt Tags verwenden, um Amazon-Rekognition-Sammlungen, Stromprozessoren und Custom-Labels-Modelle zu identifizieren, zu organisieren, zu suchen und zu filtern.	25. März 2021
Amazon Rekognition kann jetzt persönliche Schutzausrüstung erkennen	Amazon Rekognition kann jetzt Hand-, Gesichts- und Kopfbedeckungen von Personen auf einem Bild erkennen.	15. Oktober 2020
Amazon Rekognition hat neue Kategorien für die Moderation von Inhalten	Die Inhaltsmoderationskategorien von Amazon Rekognition umfassen jetzt 6 neue Kategorien: Drogen, Tabak, Alkohol, Glücksspiel, unhöfliche Gesten und Hasssymbole.	12. Oktober 2020
Neues Tutorial zur lokalen Anzeige von Amazon-Rekognition-Video-Ergebnissen aus Kinesis Video Streams	Sie können die Ausgabe von Amazon Rekognition Video aus einem Streaming-Video in Kinesis Video Streams in einem lokalen Video-Feed anzeigen.	20. Juli 2020
Neues Amazon-Rekognition-Tutorial zur Verwendung von GStreamer	Mit GStreamer können Sie ein Livestrom-Video von einer Gerätekameraquelle über Kinesis Video Streams in Amazon Rekognition Video aufnehmen.	17. Juli 2020

Amazon Rekognition unterstützt jetzt die Segmentierung gespeicherter Videos	Mit der asynchronen Amazon-Rekognition-Video-Segmentierungs-API können Sie schwarze Frames, Farbbalken, Abspanne und Aufnahmen in gespeicherten Videos erkennen.	22. Juni 2020
Amazon Rekognition unterstützt jetzt Amazon-VPC-Endpunktrichtlinien	Durch die Angabe einer Richtlinie können Sie den Zugriff auf einen Amazon-Rekognition-Amazon-VPC-Endpunkt einschränken.	3. März 2020
Amazon Rekognition unterstützt jetzt die Erkennung von Text in gespeicherten Videos	Sie können die Amazon-Rekognition-Video-API verwenden, um Text in einem gespeicherten Video asynchron zu erkennen.	17. Februar 2020
Amazon Rekognition unterstützt jetzt Augmented AI (Preview) und Amazon Rekognition Customs Labels	Mit Amazon Rekognition Custom Labels können Sie spezielle Objekte, Szenen und Konzepte in Bildern erkennen, indem Sie Ihr eigenes Modell für maschinelles Lernen erstellen. DetectModerationLabels unterstützt jetzt Amazon Augmented AI (Preview).	3. Dezember 2019
Amazon Rekognition unterstützt jetzt AWS PrivateLink	Mit AWS können PrivateLink Sie eine private Verbindung zwischen Ihrer VPC und Amazon Rekognition herstellen.	12. September 2019

Gesichtsfilterung von Amazon Rekognition	Amazon Rekognition erweitert den IndexFaces API-Betrieb um eine erweiterte Unterstützung für Gesichtsfilterung und führt Gesichtsfilterung für die CompareFaces und SearchFacesByImage API-Operationen ein.	12. September 2019
Beispiele für Amazon Rekognition Video aktualisiert	Beispielcode von Amazon Rekognition Video wurde aktualisiert, um das Amazon-SNS-Thema und die Amazon-SQS-Warteschlange zu erstellen und zu konfigurieren.	5. September 2019
Ruby- und Node.js-Beispiele hinzugefügt	Amazon-Rekognition-Image-Ruby- und Node.js-Beispiele für die synchrone Label- und Gesichtserkennung hinzugefügt.	19. August 2019
Erkennung unsicherer Inhalte aktualisiert	Die Amazon-Rekognition-Erkennung unsicherer Inhalte kann jetzt gewalttätige Inhalte erkennen.	9. August 2019
GetContentModeration Der Vorgang wurde aktualisiert	GetContentModeration gibt jetzt die Version des Moderationserkennungsmodells zurück, das zur Erkennung unsicherer Inhalte verwendet wurde.	13. Februar 2019

GetLabelDetection und die Operationen wurden aktualisiert DetectModerationLabels	GetLabelDetection gibt jetzt Bounding-Box-Informationen für häufig verwendete Objekte und eine hierarchische Taxonomie der erkannten Labels zurück. Die Version des Modells, das für die Labelerkennung verwendet wurde, wird jetzt zurückgegeben. DetectModerationLabels gibt jetzt die Version des Modells zurück, das für die Erkennung unsicherer Inhalte verwendet wurde.	17. Januar 2019
DetectFaces und der IndexFaces Betrieb wurde aktualisiert	In dieser Version wird der DetectFaces IndexFaces AND-Betrieb aktualisiert. Wenn der Eingabeparameter Attribute auf ALL gesetzt ist, enthalten die Landmarks für die Gesichtsposition 5 neue Landmarken: upperJawlineLeft, midJawlineLeft, ChinBottom, midJawlineRight, upperJawlineRight.	19. November 2018
DetectLabels Vorgang wurde aktualisiert	Nun werden Begrenzungsrahmen für bestimmte Objekte zurückgegeben. Jetzt ist eine hierarchische Taxonomie für Labels verfügbar. Sie können jetzt die Version des für die Erkennung verwendeten Erkennungsmodells anfordern.	1. November 2018

IndexFaces Operation aktualisiert	Mit können IndexFaces Sie jetzt den QualityFilter Eingabeparameter verwenden, um Gesichter herauszufiltern, die mit geringer Qualität erkannt wurden. Sie können den MaxFaces Eingabeparameter auch verwenden, um die Anzahl der zurückgegebenen Gesichter auf der Grundlage der Qualität der Gesichtserkennung und der Größe des erkannten Gesichts zu reduzieren.	18. September 2018
DescribeCollection Vorgang hinzugefügt	Sie können jetzt Informationen über eine bestehende Sammlung abrufen, indem Sie die DescribeCollection Operation aufrufen.	22. August 2018
Neue Python-Beispiele	Den Inhalten zu Amazon Rekognition Video wurden Python-Beispiele hinzugefügt und die Inhalte wurden umstrukturiert.	26. Juni 2018
Aktualisiertes Inhaltslayout	Die Inhalte für Amazon Rekognition Image wurden neu angeordnet und es wurden neue Python- und C#-Beispiele hinzugefügt.	29. Mai 2018

Amazon Rekognition unterstützt AWS CloudTrail	Amazon Rekognition ist in einen Service integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen eines Benutzers, einer Rolle oder eines AWS Dienstes in Amazon Rekognition bereitstellt. Weitere Informationen finden Sie unter Protokollieren von Amazon Rekognition Rekognition-API-Aufrufen mit AWS. CloudTrail	6. April 2018
Analysieren von gespeicherten und gestromten Videos. Neues Inhaltsverzeichnis	Weitere Informationen zum Analysieren von gespeicherten Videos finden Sie unter Arbeiten mit gespeicherten Videos. Weitere Informationen zum Analysieren von Streaming-Videos finden Sie unter Arbeiten mit Streaming-Videos. Das Inhaltsverzeichnis für die Amazon-Rekognition -Dokumentation wurde neu geordnet und enthält nun die Bild- und Videooperationen.	29. November 2017

Text in Bild- und Gesichtserkennungsmodellen	Amazon Rekognition kann jetzt Text in Bildern erkennen. Weitere Informationen finden Sie unter Erkennen von Text. Amazon Rekognition führt die Versionsverwaltung für das Deep-Learning-Modell bei der Gesichtserkennung ein. Weitere Informationen finden Sie unter Modellversionsverwaltung.	21. November 2017
Prominentenerkennung	Amazon Rekognition kann jetzt Bilder im Hinblick auf Prominente analysieren. Weitere Informationen finden Sie unter Erkennen von Prominenten.	8. Juni 2017
Überwachen von Bildern	Amazon Rekognition kann jetzt bestimmen, ob ein Bild explizite oder anzügliche, nicht jugendfreie Inhalte enthält. Weitere Informationen finden Sie unter Unsichere Inhalte erkennen.	19. April 2017

Altersbereich für erkannte Gesichter. Metrikbereich für eine Gesamtheit von Ergebnissen	Amazon Rekognition gibt jetzt den geschätzten Altersbereich in Jahren für Gesichter an, die von der Amazon-Rekognition -API erkannt wurden. Weitere Informationen finden Sie unter AgeRange. Die Rekognition-Konsole verfügt jetzt über einen Bereich mit Kennzahlen, in dem Aktivitätsdiagramme für eine Zusammenfassung von CloudWatch Amazon-Metriken für Rekognition über einen bestimmten Zeitraum angezeigt werden. Weitere Informationen finden Sie unter Übung 4: Anzeigen von Gesamtmetriken (Konsole).	9. Februar 2017
Neuer Dienst mit dazugehörigem Handbuch	Dies ist die erste Version des Bildanalysedienstes Amazon Rekognition mit dem Amazon-Rekognition-Entwicklerhandbuch.	30. November 2016

AWS Glossar

Die neueste AWS Terminologie finden Sie im AWS Glossar in der AWS-Glossar Referenz.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.