aws

Guida per gli sviluppatori

Amazon Rekognition

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

Amazon Rekognition: Guida per gli sviluppatori

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

I marchi e il trade dress di Amazon non possono essere utilizzati in relazione ad alcun prodotto o servizio che non sia di Amazon, in alcun modo che possa causare confusione tra i clienti, né in alcun modo che possa denigrare o screditare Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà delle rispettive aziende, che possono o meno essere associate, collegate o sponsorizzate da Amazon.

Table of Contents

Che cos'è Amazon Rekognition? 1
Funzionalità chiave 1
Casi d'uso 2
Vantaggi 4
Amazon Rekognition e idoneità HIPAA 4
Usi Amazon Rekognition per la prima volta? 5
Come funziona Rekognition 6
Comprendere i tipi di analisi di Rekognition 7
Etichette 9
Custom Labels 10
Rilevamento Face Liveness 10
Rilevamento e analisi facciali 11
Ricerca di volti 11
Percorsi delle persone 12
Dispositivi di protezione individuale 12
Volti celebri 12
Rilevamento del testo 12
Contenuti inappropriati o offensivi 13
Personalizzazione 13
Analisi in blocco 13
Comprendere le operazioni relative alle immagini e ai video di Rekognition 14
Operazioni di Immagini Amazon Rekognition 14
Operazioni di Video Amazon Rekognition 14
Operazioni basate su storage e non basate su storage 15
Utilizzo dell'AWS SDK o di HTTP per richiamare le operazioni API di Amazon Rekognition ... 15
Operazioni API basate su storage e non basate su storage 16
Operazioni non basate su storage 16
Operazioni API basate su storage 18
Comprendere il controllo delle versioni dei modelli 20
Nozioni di base 22
Fase 1: impostazione di un account AWS e creazione di un utente 22
Crea un AWS account e un utente 23
Fase 2: Configurare AWS CLI e AWS SDKs 25
Concessione dell'accesso programmatico 27

Lavorare con AWS SDKs ... 31
Fase 3: Iniziare a utilizzare l' AWS CLI API and AWS SDK ... 32
Formattazione degli esempi AWS CLI ... 32
Approfondimenti ... 33
Fase 4: nozioni di base per iniziare a usare la console ... 33
Configurare le autorizzazioni della console ... 34
Esercizio 1: rilevare oggetti e scene (console) ... 38
Esercizio 2: analizzare i volti (console) ... 44
Esercizio 3: confrontare i volti (console) ... 47
Esercizio 4: visualizzare i parametri aggregati (console) ... 50
Lavorare con immagini e video ... 52
Lavorare con le immagini ... 52
Specifiche dell'immagine ... 53
Analisi di immagini archiviate in un bucket Amazon S3 ... 55
Utilizzo di un file system locale ... 71
Visualizzazione di riquadri di delimitazione ... 87
Ottenere l'orientamento dell'immagine e le coordinate del riquadro di delimitazione ... 100
Utilizzo delle operazioni di analisi video archiviate ... 111
Tipi di analisi ... 112
Panoramica dell'API Video Amazon Rekognition ... 112
Chiamata delle operazioni Video Amazon Rekognition ... 115
Configurazione di Video Amazon Rekognition ... 122
Analisi di un video archiviato (SDK) ... 126
Analisi di un video (AWS CLI) ... 155
Riferimento: Notifica dei risultati dell'analisi video ... 158
Risoluzione dei problemi di Video Amazon Rekognition ... 160
Utilizzo degli eventi video in streaming ... 163
Panoramica delle operazioni del processore di streaming Video Amazon Rekognition ... 163
Aggiunta di tag a un nuovo elaboratore di flussi ... 164
Aggiunta di tag a un nuovo elaboratore di flussi ... 165
Elenca tag in un elaboratore di flussi ... 165
Eliminazione di tag da un elaboratore di flussi ... 166
Gestione degli errori ... 166
Componenti degli errori ... 166
Messaggi e codici di errore ... 167
Gestione degli errori nell'applicazione ... 172

Utilizzo di Amazon Rekognition con FedRAMP 174
Best practice per sensori, immagini di input e video 178
Latenza delle operazioni di Immagini Amazon Rekognition 178
Raccomandazioni per immagini di input di confronto di volti 179
Raccomandazioni generali per le immagini di input per le operazioni relative ai volti 179
Raccomandazioni per la ricerca dei volti in una raccolta 180
Raccomandazioni per le impostazioni della fotocamera (immagini e video) 180
Raccomandazioni per le impostazioni della fotocamera (video archiviati e in streaming) 182
Raccomandazioni per le impostazioni della fotocamera (video in streaming) 183
Raccomandazioni per l'uso di Face Liveness 184
Rilevamento di oggetti e concetti 185
Etichetta oggetti di risposta 186
Riquadri di delimitazione 186
Punteggio di attendibilità 187
Genitori 187
Categories 188
Alias 188
Proprietà immagine 189
Versione del modello 190
Filtri di inclusione ed esclusione 190
Ordinamento e aggregazione di risultati 191
Rilevamento di etichette in un'immagine 191
DetectLabels richiesta di operazione 202
DetectLabels - risposta 203
Trasformare la risposta DetectLabels 207
Rilevamento di etichette in un video 211
StartLabelDetectionRichiesta 212
GetLabelDetection Risposta all'operazione 213
Trasformazione della risposta GetLabelDetection 220
Rilevamento delle etichette negli eventi video in streaming 228
Configurazione delle risorse Video Amazon Rekognition e Amazon Kinesis 229
Chiamare le operazioni di rilevamento delle etichette per eventi video in streaming 234
Rilevamento di Custom Labels 241
Rilevamento e analisi facciale 242
Panoramica del rilevamento e del confronto facciale 243
Linee guida sugli attributi facciali 245

Rilevamento di volti in un'immagine 246
DetectFaces richiesta di operazione 257
DetectFaces risposta operativa 258
Confronto dei volti nelle immagini 265
CompareFaces richiesta di operazione 278
CompareFaces risposta operativa 278
Rilevamento di volti in un video archiviato 281
GetFaceDetection risposta operativa 290
Ricerca di volti in una raccolta 296
Gestione delle raccolte di volti, dei volti e degli utenti 299
Gestire una raccolta 299
Gestione dei volti in una raccolta 300
Gestione degli utenti in una raccolta 300
Utilizzo di soglie di somiglianza per associare e abbinare volti 301
Utilizzo delle soglie di somiglianza per l'associazione dei volti 301
Utilizzo di soglie di somiglianza per la corrispondenza dei volti 301
Linee guida per l'indicizzazione dei volti in scenari comuni 302
Applicazioni critiche o di pubblica sicurezza 302
Condivisione di foto e applicazioni social media 302
Utilizzo generico 303
Ricerca di volti e utenti all'interno di una raccolta 303
Casi d'uso che riguardano la sicurezza pubblica 304
Utilizzo di Amazon Rekognition per favorire la pubblica sicurezza 305
Creazione di una raccolta 306
CreateCollection richiesta di operazione 313
CreateCollection risposta operativa 313
Assegnazione di tag alle raccolte 313
Aggiunta di tag a una nuova raccolta 314
Aggiunta di tag a una raccolta esistente 315
Creazione dell'elenco dei tag di una raccolta 316
Eliminazione dei tag da una raccolta 317
Creazione dell'elenco delle raccolte 318
ListCollections richiesta di operazione 324
ListCollections risposta operativa 325
Descrizione di una raccolta 325
DescribeCollection richiesta di operazione 332

DescribeCollectionrisposta operativa 333
Eliminazione di una raccolta 333
DeleteCollection richiesta di operazione 340
DeleteCollection risposta operativa 340
Aggiunta di volti a una raccolta 340
Filtraggio dei volti 341
IndexFaces richiesta di operazione 351
IndexFaces risposta operativa 351
Creazione dell'elenco dei volti e degli utenti associati in una raccolta 360
ListFaces richiesta di operazione 366
ListFaces risposta all'operazione 366
Eliminazione dei volti da una raccolta 368
DeleteFaces richiesta di operazione 373
DeleteFaces risposta all'operazione 373
Creazione di un utente 374
Eliminazione di un utente 378
Associazione di volti a un utente 380
AssociateFaces risposta all'operazione 384
Dissociazione dei volti da un utente 385
DisassociateFaces risposta all'operazione 388
Creazione dell'elenco degli utenti in una raccolta 389
ListUsers risposta all'operazione 391
Ricerca di un volto (ID volto) 392
SearchFaces richiesta di operazione 399
SearchFaces risposta operativa 399
Ricerca di un volto (immagine) 400
SearchFacesByImage richiesta di operazione 408
SearchFacesByImage risposta operativa 408
Ricerca di utenti (ID volto/ ID utente) 409
SearchUsers richiesta di operazione 413
SearchUsers risposta all'operazione 414
Ricerca di utenti (immagine) 415
SearchUsersByImage richiesta di operazione 419
SearchUsersByImage risposta operativa 419
Ricerca di volti in video archiviati 421
GetFaceSearch risposta operativa 430

Ricerca di volti in una raccolta in streaming video 435
Configurazione delle risorse Video Amazon Rekognition e Amazon Kinesis 436
Ricerca di volti in un video in streaming 439
Streaming tramite un GStreamer plugin 462
Risoluzione dei problemi dello streaming di video 465
Rilevamento dei movimenti delle persone 473
GetPersonTracking risposta all'operazione 482
Rilevamento dei dispositivi di protezione individuale 487
Tipi di DPI 488
Protezione per il volto 488
Guanti 488
Copricapo 488
Affidabilità nel rilevamento dei DPI 489
Riepilogo dei DPI rilevati in un'immagine 489
Tutorial: creazione di una AWS Lambda funzione che rileva le immagini con PPE 490
Comprensione dell'API di rilevamento dei DPI 490
Fornire un'immagine 490
Comprendere la risposta DetectProtectiveEquipment 492
Rilevamento dei DPI in un immagini 497
Esempio: riquadri di delimitazione e protezioni dei volti 509
Riconoscimento delle celebrità 525
Differenze fra Riconoscimento di volti celebri e Ricerca di volti 526
Riconoscimento delle celebrità in un'immagine 526
Chiamata RecognizeCelebrities 527
RecognizeCelebrities richiesta di operazione 537
RecognizeCelebrities risposta operativa 537
Riconoscimento di celebrità in un video archiviato 540
GetCelebrityRecognition risposta all'operazione 555
Come recuperare le informazioni su una celebrità 558
Chiamata GetCelebrityInfo 558
GetCelebrityInfo richiesta di operazione 563
GetCelebrityInfo risposta operativa 563
Moderazione dei contenuti 565
Utilizzo della moderazione di immagini e video APIs 567
Categorie di etichette 567
Tipo di contenuto 579

Confidence 579
Controllo delle versioni 579
Ordinamento e aggregazione 580
Stati dell'adattatore di moderazione personalizzato 580
Test della versione 7 di Content Moderation e trasformazione della risposta dell'API 581
AWS SDK e guida all'uso per la moderazione dei contenuti versione 7 581
Mappature delle etichette per le versioni da 6.1 a 7 582
Rilevamento di immagini inappropriate 587
Rilevamento di contenuti inappropriati in un'immagine 587
..... 587
DetectModerationLabels richiesta di operazione 594
DetectModerationLabels risposta operativa 595
Rilevamento di video archiviati in modo inappropriato 596
GetContentModeration risposta operativa 605
Migliorare la precisione con la moderazione personalizzata 607
Creazione e utilizzo di adattatori 608
Preparazione dei set di dati 612
Gestione degli adattatori con la AWS CLI e SDKs 614
Tutorial sull'adattatore di moderazione personalizzato 621
Valutazione e miglioramento dell'adattatore 639
Formati di file manifest 641
Best practice per l'addestramento degli adattatori 646
Impostazione delle AutoUpdate autorizzazioni 647
AWS Notifica Health Dashboard per Rekognition 650
Revisione di contenuti non appropriati con Amazon A2I 651
Rilevamento del testo 657
Rilevamento di testo in un'immagine 659
DetectText richiesta di operazione 667
DetectText risposta operativa 668
Rilevamento del testo in un video archiviato 674
Filtri 684
GetTextDetection risposta 684
Rilevamento di segmenti video 690
Segnali d'azione tecnici 691
Fotogrammi neri 691
Crediti 691

Barre colore ... 692
Liste ... 692
Loghi Studio ... 692
Contenuti ... 692
Rilevamento delle riprese ... 693
Informazioni sull'API di rilevamento dei segmenti di Video Amazon Rekognition ... 694
Utilizzo dell'API Amazon Rekognition Segment ... 694
Avvio dell'analisi dei segmenti ... 695
Recupero dei risultati dell'analisi dei segmenti ... 696
Esempio: rilevamento di segmenti in un video archiviato ... 701
Rilevamento del riconoscimento facciale ... 714
Requisiti di riconoscimento facciale lato utente ... 716
Architettura e diagrammi di sequenza ... 717
Prerequisiti ... 719
Fase 1: Impostazione di un account AWS ... 719
Passaggio 2: configura Face Liveness AWS SDKs ... 719
Fase 3: Configurare AWS Amplify Resources ... 720
Best practice del rilevamento del riconoscimento facciale ... 720
Programmazione di Amazon Rekognition Face Liveness APIs ... 720
Fase 1: CreateFaceLivenessSession ... 721
Fase 2: StartFaceLivenessSession ... 722
Fase 3: GetFaceLivenessSessionResults ... 722
Fase 4: rispondere ai risultati ... 724
Chiamata Face Liveness APIs ... 724
Configurazione e personalizzazione dell'applicazione ... 730
Configurazione dell'applicazione ... 730
Personalizzare la tua applicazione ... 731
Modello di responsabilità condivisa per il riconoscimento facciale ... 731
Linee guida per l'aggiornamento di Face Liveness ... 735
Controllo delle versioni e tempistiche ... 735
Rilascio delle versioni e matrice di compatibilità ... 736
Comunicazione di nuove versioni ... 737
Domande frequenti su Face Liveness ... 737
Analisi in blocco ... 741
Elaborazione di immagini in blocco ... 741
Per creare un processo di analisi in blocco (CLI) ... 741

Manifesti dei risultati dell'analisi di massa 743
Tipo di contenuto 744
Verifica delle previsioni e adattatori di formazione 744
Tutorial 745
Archiviazione dei dati di Amazon Rekognition con Amazon RDS e DynamoDB 745
Prerequisiti 746
Ottenere etichette per immagini in un bucket Amazon S3 746
Creazione di una tabella Amazon DynamoDB 748
Importazione di dati in DynamoDB 749
Creazione di un database MySQL in Amazon RDS 751
Caricamento di dati su una tabella Amazon RDS MySQL 752
Utilizzo di Amazon Rekognition e Lambda per etichettare gli asset in un bucket Amazon S3 754
Prerequisiti 756
Configurazione del ruolo IAM Lambda 756
Creazione del progetto 757
Scrivi il codice 760
Package del progetto 770
Distribuire la funzione Lambda 771
Test del metodo Lambda 772
Creazione di AWS applicazioni di analisi video 773
Prerequisiti 774
Procedura 774
Creazione di una funzione Lambda Amazon Rekognition 775
Prerequisiti 777
Creazione dell'argomento SNS 777
Creazione della funzione Lambda 777
Configura la funzione Lambda 778
Configurazione del ruolo IAM Lambda 779
Crea il progetto AWS Toolkit for Eclipse Lambda 780
Test della funzione Lambda 784
Utilizzo di Amazon Rekognition per la verifica dell'identità 785
Prerequisiti 786
Creazione di una raccolta 786
Registrazione di un nuovo utente 787
Accesso utente esistente 796
Utilizzo di Amazon Rekognition (REK) per rilevare le etichette per le applicazioni di marketing . 798

Prerequisiti 799
Creazione e compilazione di una tabella DynamoDB 800
Creazione di una funzione Lambda con registrazione e notifica 802
Impostazione delle autorizzazioni e della funzione Lambda 805
Caricamento di immagini su Amazon S3 con metadati 806
CloudWatch Registri dei sondaggi del client 808
Rilevamento di etichette in un'immagine usando Lambda e Python 809
Creazione di una funzione Lambda (console) 810
(Opzionale) Creazione di un livello (console) 811
Aggiungere codice Python (console) 813
Aggiungere codice Python (console) 815
Esempi di codice 819
Nozioni di base 823
Buongiorno Amazon Rekognition 824
Azioni 827
Scenari 986
Creazione di una raccolta e individuazione dei volti al suo interno 987
Creazione di un'applicazione serverless per gestire foto 999
Rilevamento dei DPI nelle immagini 1003
Rilevamento e visualizzazione degli elementi nelle immagini 1004
Rilevamento di volti in un'immagine 1020
Rilevamento delle informazioni nei video 1021
Rilevamento di oggetti nelle immagini 1060
Rilevamento di persone e oggetti in un video 1064
Salvataggio di EXIF e altre informazioni sull'immagine 1065
Documentazione di riferimento API 1067
Sicurezza 1068
Gestione dell'identità e degli accessi 1068
Destinatari 1069
Autenticazione con identità 1069
Gestione dell'accesso con policy 1072
Come funziona Amazon Rekognition con IAM 1075
Policy gestite da AWS 1081
Best practice per le policy 1089
Utilizzo della console 1090
Esempi di utilizzo di esempi di policy basate sull'identità 1090

Esempi di politiche basate sulle risorse ... 1093
Risoluzione dei problemi ... 1094
Protezione dei dati ... 1097
Crittografia dei dati ... 1098
Riservatezza del traffico Internet ... 1100
Utilizzo di Amazon Rekognition con endpoint VPC Amazon ... 1100
Creazione endpoint VPC Amazon per Amazon Rekognition ... 1101
Creazione di una policy di endpoint VPC per Amazon Rekognition ... 1102
Convalida della conformità ... 1103
Resilienza ... 1104
Analisi della configurazione e delle vulnerabilità ... 1104
Prevenzione del problema "confused deputy" tra servizi ... 1104
Sicurezza dell'infrastruttura ... 1107
Monitoraggio ... 1108
Monitoraggio di Rekognition con Amazon CloudWatch ... 1108
Utilizzo delle CloudWatch metriche per Rekognition ... 1109
Accesso ai parametri di Rekognition ... 1110
Creazione di un allarme ... 1111
CloudWatchmetriche per Rekognition ... 1113
Registrazione delle chiamate API Amazon Rekognition con AWS CloudTrail ... 1118
Informazioni su Amazon Rekognition in CloudTrail ... 1118
Informazioni sulle voci dei file di log Amazon Rekognition ... 1119
Linee guida e quote ... 1123
Regioni supportate ... 1123
Quote impostate ... 1123
Immagini Amazon Rekognition ... 1123
Analisi di massa delle immagini di Amazon Rekognition ... 1123
Video archiviato di Video Amazon Rekognition ... 1125
Video Amazon Rekognition video in streaming ... 1125
Quote di default ... 1125
Calcolo della modifica della quota TPS ... 1126
Best practice per le quote TPS ... 1126
Crea un caso per modificare le quote TPS ... 1127
Cronologia dei documenti ... 1130
AWS Glossario ... 1145
... mcxlvi

Che cos'è Amazon Rekognition?

Amazon Rekognition è un servizio di analisi di immagini e video basato sul cloud che semplifica l'aggiunta di funzionalità avanzate di visione artificiale alle tue applicazioni. Il servizio è basato su una comprovata tecnologia di deep learning e non richiede competenze di machine learning per essere utilizzato. Amazon Rekognition include un'API easy-to-use semplice in grado di analizzare rapidamente qualsiasi file di immagine o video archiviato in Amazon S3.

Puoi aggiungere funzionalità che rilevano oggetti, testo, contenuti non sicuri, analizzano immagini/video e confrontano volti alla tua applicazione utilizzando Rekognition. APIs Con il riconoscimento facciale di Amazon Rekognition APIs, puoi rilevare, analizzare e confrontare volti per un'ampia varietà di casi d'uso, tra cui la verifica degli utenti, la catalogazione, il conteggio delle persone e la sicurezza pubblica.

Il servizio si basa sulla stessa tecnologia di deep learning collaudata e altamente scalabile sviluppata dagli scienziati di visione artificiale di Amazon, tecnologia in grado di analizzare miliardi di immagini e video ogni giorno. Rekognition impara regolarmente dai nuovi dati e spesso aggiungiamo nuove etichette e funzionalità al servizio.

Per ulteriori informazioni, consulta Amazon Rekognition. FAQs

Funzionalità chiave

Analisi delle immagini:

- Rilevamento di oggetti, scene e concetti: rileva e classifica oggetti, scene, concetti e celebrità nelle immagini.
- Rilevamento del testo: rileva e riconosce il testo stampato e scritto a mano nelle immagini in diverse lingue.
- Contenuti non sicuri: rileva e filtra contenuti e immagini espliciti, inappropriati e violenti. Rileva etichette granulari di contenuto non sicuro.
- Riconoscimento delle celebrità: riconosci nelle tue immagini decine di migliaia di celebrità di diverse categorie, come politici, atleti, attori e musicisti.
- Analisi facciale: rileva, analizza e confronta i volti, oltre agli attributi facciali, come sesso, età ed emozioni. I casi d'uso possono includere la verifica degli utenti, la catalogazione, il conteggio delle persone e la sicurezza pubblica.

- Etichette personalizzate: crea classificatori personalizzati per rilevare oggetti specifici per il tuo caso d'uso, come loghi, prodotti, personaggi.
- Proprietà dell'immagine: analizza le proprietà dell'immagine come qualità, colore, nitidezza e contrasto.

Analisi video:

- Rilevamento di oggetti, scene e concetti: rileva e classifica oggetti, scene, concetti e celebrità nei video.
- Rilevamento del testo: rileva e riconosce il testo stampato e scritto a mano nei video in diverse lingue.
- Percorso delle persone: monitora le persone identificate mentre si spostano tra i fotogrammi del video.
- Analisi facciale: rileva, analizza e confronta i volti in streaming o nei video archiviati.
- Riconoscimento di celebrità: riconosci decine di migliaia di celebrità nei tuoi video archiviati in diverse categorie, come politici, atleti, attori e musicisti.
- Rilevamento di contenuti non sicuri: rileva contenuti espliciti, inappropriati e violenti nei video.
- Segmentazione video: identifica automaticamente segmenti video utili, come cornici nere e titoli di coda.
- Vivacità del viso: rileva se un utente dal vivo è presente durante la verifica facciale.

Casi d'uso

Librerie multimediali ricercabili - Rekognition rileva etichette, oggetti, concetti e scene in immagini e video. Puoi rendere queste etichette ricercabili sulla base di questa analisi del contenuto visivo. Utile per creare librerie di immagini e video ricercabili.

Verifica dell'identità utente basata sui volti: conferma l'identità degli utenti confrontando i volti nelle immagini con le immagini dei volti di riferimento. Utile per la verifica dell'identità nelle applicazioni.

Face Liveness Detection - Rekognition Face Liveness è una funzionalità di machine learning (ML) completamente gestita progettata per aiutare gli sviluppatori a scoraggiare le frodi durante la verifica dell'identità basata sul volto. La funzionalità consente di verificare che un utente sia fisicamente presente davanti alla telecamera e non sia un cattivo attore a falsificare il volto dell'utente. L'uso di Rekognition Face Liveness può aiutarti a rilevare attacchi contraffatti presentati a una fotocamera,

come foto stampate, foto/video digitali o maschere 3D. Inoltre, aiuta a rilevare attacchi falsi che aggirano una telecamera, come video preregistrati o deepfake iniettati direttamente nel sottosistema di acquisizione video.

Ricerca facciale: con Rekognition, puoi cercare immagini, video archiviati e video in streaming per trovare volti che corrispondano a quelli archiviati in un contenitore noto come raccolta di volti. ovvero degli indici di volti di cui hai la proprietà e la gestione. Per cercare le persone in base ai loro volti è necessario indicizzare i volti e poi cercare i volti.

Rilevamento di contenuti non sicuri: rileva e filtra contenuti espliciti, inappropriati e violenti in immagini e video. Utilizza etichette per il filtraggio granulare in base alle esigenze aziendali. L'API Content Moderation restituisce anche un elenco gerarchico di tutte le etichette rilevate (oggetti e concetti), insieme ai punteggi di affidabilità. Questi oggetti/etichette indicano categorie specifiche di contenuti non sicuri e consentono di filtrare in maniera granulare e gestire grandi volumi di contenuti generati dagli utenti (UGC). Puoi personalizzare l'output dell'API Content Moderation con adattatori, che migliorano le prestazioni di immagini come quelle fornite come dati di formazione.

Rilevamento dei dispositivi di protezione individuale: rileva i dispositivi di protezione individuale nelle immagini per monitorare la conformità alla sicurezza in vari settori. È possibile segnalare automaticamente le condizioni non sicure rilevando apparecchiature inadeguate e ricevere avvisi relativi a tali condizioni, il che può migliorare la conformità e la formazione.

Riconoscimento delle celebrità: riconosci le celebrità nelle tue immagini e nei tuoi video in diverse categorie, come politici, atleti, attori e musicisti. Puoi identificare le apparizioni delle celebrità senza dover fornire i nomi.

Rilevamento del testo: rileva ed estrai il testo nelle immagini per la ricerca visiva o l'estrazione di metadati. Funziona con diversi tipi di carattere e stili. Rileva l'orientamento per gestire il testo su cartelli e banner.

Etichette personalizzate: identifica oggetti, concetti e scene personalizzati specifici per casi d'uso aziendali, come il rilevamento del logo. È possibile addestrare classificatori personalizzati a gestire oggetti di nicchia o proprietari, il che migliora la precisione degli oggetti chiave rispetto ai classificatori generali. Per ulteriori informazioni, consulta la pagina relativa a Cos'è Etichette personalizzate Amazon Rekognition? nella Guida per gli sviluppatori di etichette personalizzate Amazon Rekognition.

Vantaggi

Integrazione di potenti analisi di immagini e video nella tua app: aggiungi un'analisi accurata di immagini e video alle app senza esperienza. L'API Amazon Rekognition consente l'analisi tramite deep learning senza richiedere alcuna conoscenza di machine learning. Puoi integrare rapidamente la visione artificiale in app web, mobili e per dispositivi mobili.

Analisi di immagini e video basata sul deep learning: analizza immagini e video utilizzando il deep learning per un'elevata precisione. Amazon Rekognition è in grado di rilevare etichette, oggetti, scene, volti, personaggi famosi. Filtra i risultati per includere/escludere etichette specifiche.

Analisi scalabile delle immagini: analizza milioni di immagini per organizzare enormi set di dati visivi. Scalabilità per gestire librerie di immagini e traffico in crescita. Non è necessario pianificare la capacità e si paga solo per ciò che si utilizza.

Analizza e filtra le immagini in base alle proprietà: analizza e filtra le immagini in base a proprietà, come qualità, colore e contenuto visivo, e rileva nitidezza, luminosità e contrasto delle immagini.

Integrazione con altri AWS servizi: Amazon Rekognition si integra immediatamente con S3 e Lambda. Puoi chiamare Amazon Rekognition APIs da Lambda ed elaborare immagini in Amazon S3 senza spostare i dati. Rekognition offre scalabilità e sicurezza integrate utilizzando IAM. AWS

Basso costo: Pay-as-you-go prezzi, nessun minimo o impegno. Piano gratuito disponibile per iniziare. Risparmia di più man mano che l'utilizzo aumenta grazie alla tariffazione a più livelli. Conveniente rispetto alle soluzioni interne.

Personalizzazione semplice: personalizza la precisione in base al tuo caso d'uso con gli adattatori. Fornisci immagini di esempio per addestrare gli adattatori. Migliora il rilevamento di oggetti ed etichette per determinati domini. Un modo semplice per personalizzare l'analisi senza competenze di machine learning.

Per ulteriori informazioni, consulta Amazon Rekognition. FAQs

Amazon Rekognition e idoneità HIPAA

Questo è un servizio idoneo ai fini HIPAA. Per ulteriori informazioni sull' AWS U.S. Health Insurance Portability and Accountability Act del 1996 (HIPAA) e sull'utilizzo AWS dei servizi per elaborare, archiviare e trasmettere informazioni sanitarie protette (PHI), vedere Panoramica HIPAA.

Usi Amazon Rekognition per la prima volta?

Se è la prima volta che utilizzi Amazon Rekognition, ti consigliamo di leggere le seguenti sezioni in ordine:

1. Come funziona Amazon Rekognition— Questa sezione presenta vari componenti di Amazon Rekognition con cui lavori per creare un'esperienza. end-to-end
2. Nozioni di base su Amazon Rekognition— In questa sezione, configura il tuo account, installa l'SDK che riflette la lingua che preferisci e testa l'API Amazon Rekognition. Per un elenco dei linguaggi di programmazione supportati da Amazon Rekognition, consulta Usare Rekognition con un SDK AWS.
3. Lavorare con le immagini: in questa sezione vengono fornite informazioni sull'utilizzo di Amazon Rekognition con immagini archiviate in bucket Amazon S3; o caricate da un file system locale.
4. Utilizzo delle operazioni di analisi video archiviate: in questa sezione vengono fornite informazioni sull'utilizzo di Amazon Rekognition con video archiviati in bucket Amazon S3.
5. Utilizzo degli eventi video in streaming: in questa sezione vengono fornite informazioni sull'utilizzo di Amazon Rekognition con video in streaming.

Come funziona Amazon Rekognition

Amazon Rekognition fornisce due set di API per l'analisi visiva:

- Amazon Rekognition Image per l'analisi delle immagini
- Amazon Rekognition Video per l'analisi video

Analisi delle immagini

Con Amazon Rekognition Image le tue applicazioni possono:

- Rileva oggetti, scene e concetti nelle immagini
- Riconosci le celebrità
- Rileva il testo in diverse lingue
- Rileva contenuti o immagini espliciti, inappropriati o violenti
- Rileva, analizza e confronta volti e caratteristiche facciali come età ed emozioni
- Rileva la presenza di DPI

I casi d'uso includono il miglioramento delle app fotografiche, la catalogazione di immagini e la moderazione dei contenuti.

Analisi video

Con Amazon Rekognition Video, le tue applicazioni possono:

- Tieni traccia di persone e oggetti attraverso i frame video
- Riconosci gli oggetti
- Riconosci le celebrità
- Cerca video archiviati e in streaming per persone di interesse
- Analizza i volti per caratteristiche come età ed emozioni
- Rileva contenuti o immagini espliciti, inappropriati o violenti
- Aggrega e ordina i risultati dell'analisi per timestamp e segmenti
- Rileva persone, animali domestici e pacchi nei video in streaming

I casi d'uso includono l'analisi video, la catalogazione di video e il filtraggio di contenuti inappropriati.

Caratteristiche principali

- Potente analisi del deep learning
- Rilevamento ad alta precisione di oggetti, scene, volti, testo
- API facile da usare per l'integrazione nelle app
- Modelli personalizzabili adattati ai tuoi dati
- Analisi scalabile delle librerie multimediali

Amazon Rekognition ti consente di migliorare la precisione di determinati modelli di deep learning addestrando un adattatore personalizzato. Ad esempio, con Amazon Rekognition Custom Moderation, puoi adattare il modello di analisi delle immagini di base di Amazon Rekognition addestrando un adattatore personalizzato con le tue immagini. Per ulteriori informazioni, consulta Migliorare la precisione con la moderazione personalizzata.

Le seguenti sezioni illustrano i tipi di analisi forniti da Amazon Rekognition e una panoramica delle operazioni di Amazon Rekognition Image e Amazon Rekognition Video. Viene inoltre trattata la differenza tra le operazioni basate su storage e non basate su storage.

Per una demo di APIs Amazon Rekognition, consulta la Fase 3: Guida introduttiva all'uso dell'interfaccia a riga di comando di AWS e dell'API SDK AWS, che illustra come provare Rekognition nella console. AWS

Argomenti

- Comprendere i tipi di analisi di Rekognition
- Comprendere le operazioni relative alle immagini e ai video di Rekognition
- Comprensione delle operazioni delle API diverse dallo storage e dallo storage
- Comprendere il controllo delle versioni dei modelli

Comprendere i tipi di analisi di Rekognition

Di seguito sono riportati i tipi di analisi che possono eseguire l'API Immagini Amazon Rekognition e l'API Video Amazon Rekognition. Per informazioni su APIs, vedereComprendere le operazioni relative alle immagini e ai video di Rekognition.

La tabella seguente elenca le operazioni da utilizzare in base al tipo di supporto con cui stai lavorando e al tuo caso d'uso:

Caso d'uso	Tipo di media	Operazioni
Moderazione dei contenuti	Immagini	DetectModerationLabels, StartMediaAnalysisJob, GetMediaAnalysisJob, ListMediaAnalysisJobs
	Video archiviato	StartContentModeration, GetContentModeration
Verifica dell'identità	Immagini	CreateCollection, CreateUser, IndexFaces, AssociateFaces, SearchFacesByImage, SearchUsersByImage
	Video archiviato	CreateCollection, IndexFaces, StartFaceSearch, GetFaceSearch
	Video in streaming (Rilevamento del riconoscimento facciale)	CreateFaceLivenessSession, StartFaceLivenessSession, GetFaceLivenessSessionResults,
Analisi facciale	Immagini	DetectFaces, CompareFaces
	Video archiviato	StartFaceDetection, GetFaceDetection
	Video in streaming	CreateStreamProcessor, StartStreamProcessor
Riconoscimento di oggetti e attività	Immagini	DetectLabels
	Video archiviato	StartLabelDetection, GetLabelDetection
Casa connessa	Video in streaming	StartStreamProcessor

Caso d'uso	Tipo di media	Operazioni
Analisi dei media	Video archiviato	StartSegmentDetection, GetSegmentDetection
Sicurezza sul posto di lavoro	Immagini	DetectProtectiveEquipment
Rilevamento del testo	Immagini	DetectText
	Video	StartTextDetection, GetTextDetection
Rilevamento dei movimenti delle persone	Video	StartPersonTracking, GetPersonTracking
Riconoscimento di volti celebri	Immagini	RecognizeCelebrities
	Video	StartCelebrityRecognition, GetCelebrityRecognition
Rilevamento di etichette personalizzate	Immagini	DetectCustomLabels
	Training del modello	Consulta la guida per sviluppatori di Custom Labels

Etichette

Un'etichetta si riferisce a uno dei seguenti elementi: oggetti (ad esempio fiori, alberi o tavolo), eventi (ad esempio un matrimonio, una laurea o una festa di compleanno), concetti (ad esempio un paesaggio, una sera e la natura) o attività (ad esempio correre o giocare a basket). Amazon Rekognition è in grado rilevare volti in immagini e video. Per ulteriori informazioni, consulta Rilevamento di oggetti e concetti.

Rekognition è in grado di rilevare un ampio elenco di etichette nelle immagini e nei video archiviati. Rekognition è anche in grado di rilevare un numero limitato di etichette nei video in streaming.

Utilizza le seguenti operazioni per rilevare le etichette in base al proprio caso d'uso:

- Per rilevare le etichette nelle immagini: Usa DetectLabels. È possibile identificare le proprietà dell'immagine come i colori dominanti e la qualità dell'immagine. Per ottenere ciò, usa DetectLabelswith `IMAGE_PROPERTIES` come parametro di input.
- Per rilevare le etichette nei video archiviati: Usa StartLabelDetection. Il rilevamento dei colori e della qualità dell'immagine dominanti non è supportato per i video archiviati.
- Per rilevare le etichette nei video in streaming: Usa CreateStreamProcessor. Il rilevamento dei colori e della qualità dell'immagine dominanti non è supportato per i video in streaming.

È possibile specificare i tipi di etichette che si desidera vengano restituiti per il rilevamento delle etichette di immagini e video archiviati utilizzando opzioni di filtro complete ed esclusive.

Custom Labels

Etichette personalizzate Amazon Rekognition può identificare gli oggetti e le scene nelle immagini secondo le tue necessità aziendali addestrando un modello di machine learning. Ad esempio, puoi addestrare un modello per rilevare loghi o parti di macchine su una linea di assemblaggio.

> **Note**
>
> Per informazioni su Etichette personalizzate Amazon Rekognition, consulta la Guida per gli sviluppatori di Etichette personalizzate Amazon Rekognition.

Amazon Rekognition fornisce una console che puoi utilizzare per creare, addestrare, valutare ed eseguire un modello di machine learning. Per ulteriori informazioni, consulta la pagina relativa alle nozioni di base su Etichette personalizzate Amazon Rekognition nella Guida per gli sviluppatori di Etichette personalizzate Amazon Rekognition. Inoltre puoi utilizzare l'API Etichette personalizzate Amazon Rekognition per addestrare ed eseguire un modello. Per ulteriori informazioni, consulta la sezione Guida introduttiva all'SDK Amazon Rekognition Custom Labels nella Amazon Rekognition Developer Guide. CustomLabels

Per analizzare le immagini utilizzando un modello addestrato, usa. DetectCustomLabels

Rilevamento Face Liveness

Amazon Rekognition Face Liveness può aiutarti a verificare che un utente sottoposto a verifica dell'identità basata sul volto sia fisicamente presente davanti alla telecamera e non sia un

malintenzionato che falsifica il volto dell'utente. Rileva lo spoofing rivolto a una telecamera o attacchi che bypassano una telecamera. Un utente può completare un controllo di Face Liveness scattando un breve video selfie e a tale scopo viene restituito un punteggio Liveness. Il riconoscimento facciale viene determinato con un calcolo probabilistico e a seguito del controllo viene fornito un punteggio di confidenza (compreso tra 0 e 100). Più alto è il punteggio, maggiore è la fiducia che la persona che effettua il controllo sia effettivamente presente sul posto.

Per ulteriori informazioni su Face Liveness, consultaRilevamento del riconoscimento facciale.

Rilevamento e analisi facciali

Amazon Rekognition è in grado rilevare volti in immagini e video. Con Amazon Rekognition, puoi ottenere informazioni su:

- Dove vengono rilevati i volti in un'immagine o in un video
- Punti di riferimento facciali, ad esempio la posizione degli occhi
- La presenza di occlusione facciale nelle immagini
- Emozioni rilevate, come felicità o tristezza
- Direzione dello sguardo di una persona nelle immagini

Puoi anche interpretare informazioni demografiche come sesso o età. È possibile confrontare un volto in un'immagine con i volti rilevati in un'altra immagine. Le informazioni sui volti possono anche essere memorizzate per il successivo recupero. Per ulteriori informazioni, consulta Rilevamento e analisi facciale.

Per rilevare i volti nelle immagini, utilizzare DetectFaces. Per rilevare i volti nei video archiviati, utilizzare StartFaceDetection.

Ricerca di volti

Amazon Rekognition può cercare volti. Le informazioni facciali sono indicizzate in un container noto come raccolta. Le informazioni sui volti contenute nella raccolta possono quindi essere abbinate ai volti rilevati nelle immagini, nei video memorizzati e nello streaming video. Per ulteriori informazioni, consultare Ricerca di volti in una raccolta.

Per cercare volti conosciuti nelle immagini, utilizzare DetectFaces. Per cercare volti conosciuti nei video archiviati, utilizzare StartFaceDetection. Per cercare volti conosciuti nei video in streaming, utilizzare CreateStreamProcessor.

Percorsi delle persone

Video Amazon Rekognition può tracciare i movimenti delle persone in un video archiviato. Video Amazon Rekognition fornisce tracciamento, dettagli facciali e informazioni sulla posizione in-frame per le persone rilevate in un video. Per ulteriori informazioni, consulta Rilevamento dei movimenti delle persone.

Per rilevare le persone nei video archiviati, utilizzare StartPersonTracking.

Dispositivi di protezione individuale

Amazon Rekognition è in grado di rilevare i dispositivi di protezione individuale (DPI) indossati dalle persone rilevati in un'immagine. Amazon Rekognition rileva protezioni per volto, mani e testa. Amazon Rekognition prevede se un dispositivo di DPI copre la parte del corpo appropriata. Puoi anche ottenere riquadri di delimitazione per le persone rilevate e gli articoli DPI. Per ulteriori informazioni, consulta Rilevamento dei dispositivi di protezione individuale.

Per rilevare i PPE nelle immagini, utilizzare DetectProtectiveEquipment.

Volti celebri

Amazon Rekognition può identificare migliaia di celebrità nelle immagini e nei video archiviati. È possibile ottenere informazioni su dove si trova il volto di una celebrità su un'immagine, sui punti di riferimento facciali e sulla posa del volto. È possibile ottenere informazioni di monitoraggio sulle celebrità quando appaiono in un video archiviato. Puoi anche ottenere ulteriori informazioni su una celebrità riconosciuta, come l'emozione espressa e la presentazione del genere. Per ulteriori informazioni, consulta Riconoscimento delle celebrità.

Per riconoscere volti celebri in un'immagine, utilizzare RecognizeCelebrities. Per riconoscere volti celebri in un video archiviato, utilizzare StartCelebrityRecognition.

Rilevamento del testo

Amazon Rekognition Text in Image è in grado di rilevare il testo nelle immagini e convertirlo in un formato leggibile dal computer. Per ulteriori informazioni, consulta Rilevamento del testo.

Per rilevare il testo nelle immagini, utilizzare DetectText.

Contenuti inappropriati o offensivi

Amazon Rekognition è in grado di analizzare immagini e video archiviati per contenuti per adulti e violenti. Per ulteriori informazioni, consulta Moderazione dei contenuti.

Per rilevare immagini sconsigliate, utilizzare DetectModerationLabels. Per rilevare i contenuti sconsigliati nei video archiviati, utilizzare StartContentModeration.

Personalizzazione

Alcune analisi delle immagini APIs offerte da Rekognition consentono di migliorare l'accuratezza dei modelli di deep learning creando adattatori personalizzati addestrati sui propri dati. Gli adattatori sono componenti che si collegano al modello di deep learning pre-addestrato di Rekognition, che ne migliorano la precisione grazie alla conoscenza del dominio basata sulle immagini. Addestra un adattatore per soddisfare le tue esigenze fornendo e annotando immagini di esempio.

Dopo aver creato un adattatore, ti viene fornito un. AdapterId Puoi fornirlo AdapterId a un'operazione per specificare che desideri utilizzare l'adattatore che hai creato. Ad esempio, fornisci l'DetectModerationLabelsAPI AdapterId per l'analisi sincrona delle immagini. Forniscili AdapterId come parte della richiesta e Rekognition li utilizzerà automaticamente per migliorare le previsioni per le tue immagini. Ciò consente di sfruttare le funzionalità di Rekognition personalizzandola in base alle proprie esigenze.

Hai anche la possibilità di ottenere previsioni per le immagini in blocco con l'API. StartMediaAnalysisJob Per ulteriori informazioni, consulta Analisi in blocco.

Puoi valutare l'accuratezza delle operazioni di Rekognition caricando immagini sulla console Rekognition ed eseguendo analisi su queste immagini. Rekognition annoterà le tue immagini utilizzando la funzionalità selezionata e potrai quindi rivedere le previsioni, utilizzando le previsioni verificate per determinare quali etichette trarrebbero vantaggio dalla creazione di un adattatore.

Attualmente è possibile utilizzare adattatori con. DetectModerationLabels Per ulteriori informazioni sull'utilizzo dell'adattatore, consulta Migliorare la precisione con la moderazione personalizzata.

Analisi in blocco

Rekognition Bulk Analysis consente di elaborare un'ampia raccolta di immagini in modo asincrono utilizzando un file manifest insieme all'operazione. StartMediaAnalysisJob Per ulteriori informazioni, consulta Analisi in blocco.

Comprendere le operazioni relative alle immagini e ai video di Rekognition

Amazon Rekognition offre due set di API principali per l'analisi di immagini e video:

- Amazon Rekognition Image: questa API è progettata per l'analisi delle immagini.
- Amazon Rekognition Video: questa API si concentra sull'analisi dei video archiviati e in streaming.

Entrambe sono APIs in grado di rilevare varie entità come volti e oggetti. Per una comprensione completa dei tipi di confronto e rilevamento supportati, consulta la sezione dedicataComprendere i tipi di analisi di Rekognition.

Operazioni di Immagini Amazon Rekognition

Le operazioni di Amazon Rekognition Image sono sincrone. L'input e la risposta sono in formato JSON. Le operazioni di Immagini Amazon Rekognition analizzano un'immagine di input in formato .jpg o .png. L'immagine trasmessa a un'operazione di Immagini Amazon Rekognition Image può essere archiviata in un bucket Amazon S3. Se non utilizzi l'AWS CLI, puoi anche passare byte di immagini codificate Base64 direttamente a un'operazione Amazon Rekognition. Per ulteriori informazioni, consulta Lavorare con le immagini.

Operazioni di Video Amazon Rekognition

L'API Amazon Rekognition Video facilita l'analisi dei video archiviati in un bucket Amazon S3 o trasmessi in streaming tramite Amazon Kinesis Video Streams.

Per le operazioni relative ai video archiviati, tieni presente quanto segue:

- Le operazioni sono asincrone.
- L'analisi deve essere avviata con un'operazione «Start» (ad esempio, StartFaceDetectionper il rilevamento dei volti nei video memorizzati).
- Lo stato di completamento dell'analisi viene pubblicato su un argomento di Amazon SNS.
- Per recuperare i risultati di un'analisi, usa l'operazione «Get» corrispondente (ad esempio, GetFaceDetection).
- Per ulteriori informazioni, consultate Utilizzo dell'analisi video memorizzata.

Per l'analisi dei video in streaming:

- Le funzionalità includono la ricerca facciale nelle raccolte Rekognition Video e il rilevamento di etichette (oggetti o concetti).
- I risultati dell'analisi per le etichette vengono inviati come notifiche Amazon SNS e Amazon S3.
- I risultati della ricerca facciale vengono inviati a un flusso di dati Kinesis.
- La gestione dell'analisi dei video in streaming viene effettuata tramite un processore di streaming Amazon Rekognition Video (ad esempio, creando un processore utilizzando). CreateStreamProcessor
- Per ulteriori informazioni, consulta Lavorare con gli eventi video in streaming.

Ogni operazione di analisi video restituisce i metadati relativi al video analizzato, oltre a un ID del lavoro e un tag di lavoro. Operazioni come il rilevamento delle etichette e la moderazione dei contenuti per i video consentono l'ordinamento per timestamp o nome dell'etichetta e l'aggregazione dei risultati per timestamp o per segmento.

Operazioni basate su storage e non basate su storage

Le operazioni di Amazon Rekognition sono raggruppate nelle seguenti categorie.

- Operazioni API non basate su storage – In queste operazioni, Amazon Rekognition non mantiene alcuna informazione. Si forniscono immagini e video di input, l'operazione esegue l'analisi e restituisce i risultati, ma nulla viene salvato da Amazon Rekognition. Per ulteriori informazioni, consulta Operazioni non basate su storage.
- Operazioni API basate su storage – i server Amazon Rekognition possono memorizzare le informazioni sul volto rilevate in container noti come raccolte. Amazon Rekognition fornisce ulteriori operazioni API che è possibile utilizzare per cercare le informazioni sui volti mantenute per le corrispondenze dei volti stessi. Per ulteriori informazioni, consulta Operazioni API basate su storage.

Utilizzo dell'AWS SDK o di HTTP per richiamare le operazioni API di Amazon Rekognition

È possibile chiamare le operazioni API di Amazon Rekognition utilizzando l'AWS SDK o direttamente tramite HTTP. A meno che non si abbia una buona ragione per non farlo, è consigliato sempre utilizzare l'SDK AWS. Gli esempi di Java in questa sezione utilizzano l'SDK AWS. Non viene fornito un file di progetto Java, ma è possibile utilizzare AWS Toolkit for Eclipse per sviluppare applicazioni AWS utilizzando Java.

Negli esempi di .NET riportati in questa sezione viene utilizzato AWS SDK per .NET. È possibile utilizzare AWS Toolkit for Visual Studio per sviluppare applicazioni AWS tramite .NET. Include modelli utili e AWS Explorer per l'implementazione delle applicazioni e la gestione dei servizi.

La documentazione di riferimento API di questa guida riguarda la procedura di chiamata delle operazioni di Amazon Rekognition con HTTP. Per le informazioni di riferimento su Java, consulta AWS SDK per Java.

Gli endpoint del servizio Amazon Rekognition che è possibile utilizzare sono documentati su regioni ed endpoint AWS.

Quando si richiama Amazon Rekognition con HTTP, utilizzare le operazioni POST HTTP.

Comprensione delle operazioni delle API diverse dallo storage e dallo storage

Amazon Rekognition fornisce due tipi di operazioni API. Si tratte delle operazioni non basate su storage in cui non sono memorizzate informazioni da parte di Amazon Rekognition e delle operazioni basate su storage in cui alcune informazioni sui volti sono memorizzate da Amazon Rekognition.

Operazioni non basate su storage

Amazon Rekognition fornisce le seguenti operazioni API non basate su storage per le immagini:

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

Amazon Rekognition fornisce le seguenti operazioni API non basate su storage per i video:

- StartLabelDetection

- StartFaceDetection
- StartPersonTracking
- StartCelebrityRecognition
- StartContentModeration

Queste sono indicate come operazioni API non basate su storage perché quando si chiama l'operazione, Amazon Rekognition non mantiene alcuna informazione individuata riguardante l'immagine di input. Come tutte le altre operazioni API di Amazon Rekognition, nessun byte di immagine di input viene mantenuto da operazioni API non basate su storage.

I seguenti scenari di esempio mostrano dove è possibile integrare le operazioni API non basate su storage nell'applicazione. Questi scenari presuppongono il possesso di un repository locale di immagini.

Example 1: un'applicazione che trova immagini nel repository locale che contengono etichette specifiche

Innanzitutto, è possibile rilevare le etichette (oggetti e concetti) usando l'operazione di Amazon Rekognition `DetectLabels` in ciascuna delle immagini nel repository e costruire un indice lato client, come mostrato di seguito:

```
Label        ImageID

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

Quindi, l'applicazione può cercare all'interno di questo indice per trovare immagini nel repository locale che contengono un'etichetta specifica. Ad esempio, può mostrare le immagini che contengono un albero.

Ogni etichetta che Amazon Rekognition rileva ha un valore di affidabilità associato. Indica il livello di affidabilità che l'immagine di input contenga quella determinata etichetta. È possibile utilizzare questo valore di affidabilità per eseguire facoltativamente un ulteriore filtraggio lato client sulle etichette, a seconda dei requisiti dell'applicazione in base al livello di affidabilità nel rilevamento. Ad esempio, se

si richiedono etichette precise, è possibile filtrare e scegliere solo le etichette con maggiore affidabilità (ad esempio 95% o superiore). Se l'applicazione non richiede un valore di affidabilità superiore, è possibile scegliere di filtrare le etichette con un valore di affidabilità inferiore (vicino al 50%).

Example 2: un'applicazione per visualizzare immagini del volto migliorate

Innanzitutto, è possibile rilevare i volti in ciascuna delle immagini nel repository locale usando l'operazione `DetectFaces` di Amazon Rekognition e costruendo un indice lato client. Per ogni volto, l'operazione restituisce metadata che includono un riquadro di delimitazione, punti di riferimento facciali (ad esempio, la posizione della bocca e dell'orecchio) e attributi facciali (ad esempio, sesso). È possibile archiviare questi metadata in un indice locale lato client, come indicato di seguito:

```
ImageID     FaceID     FaceMetaData

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

In questo indice, la chiave primaria è una combinazione di `ImageID` e `FaceID`.

Pertanto, è possibile utilizzare le informazioni contenute nell'indice per migliorare le immagini quando l'applicazione le visualizza dal repository locale. Ad esempio, è possibile aggiungere un riquadro di delimitazione attorno al volto o evidenziare le caratteristiche facciali.

Operazioni API basate su storage

Amazon Rekognition Image IndexFacessupporta l'operazione, che puoi utilizzare per rilevare i volti in un'immagine e mantenere le informazioni sui tratti del viso rilevati in una raccolta Amazon Rekognition. Di seguito è riportato un esempio di un'operazione API basata su storage, poiché le informazioni vengono mantenute nel server.

Immagini Amazon Rekognition fornisce le seguenti operazioni API basate su storage:

- IndexFaces
- ListFaces
- SearchFacesByImage

- SearchFaces
- DeleteFaces
- DescribeCollection
- DeleteCollection
- ListCollections
- CreateCollection

Video Amazon Rekognition fornisce le seguenti operazioni API basate su storage:

- StartFaceSearch
- CreateStreamProcessor

Per archiviare le informazioni sui volti, è necessario creare una raccolta di volti in una delle regioni AWS nell'account. La raccolta di volti viene specificata chiamando l'operazione `IndexFaces`. Dopo aver creato una raccolta di volti e aver archiviato le informazioni sulle caratteristiche facciali per tutti i volti, sarà possibile cercare le corrispondenze di volti al suo interno. Ad esempio, è possibile rilevare il volto più grande di un'immagine e cercare i volti corrispondenti in una raccolta richiamando `searchFacesByImage`.

Le informazioni facciali memorizzate in raccolte da `IndexFaces` sono accessibili alle operazioni di Video Amazon Rekognition. Ad esempio è possibile effettuare una ricerca all'interno di un video delle persone i cui volti corrispondono a quelli in una raccolta esistente richiamando StartFaceSearch.

Per ulteriori informazioni sulla creazione e la gestione di raccolte, consulta Ricerca di volti in una raccolta.

> **Note**
>
> Le raccolte memorizzano i vettori dei volti, che sono rappresentazioni matematiche dei volti. Le raccolte non memorizzano immagini di volti.

Example 1: un'applicazione che autentica l'accesso a un edificio

Per iniziare, si crea una raccolta di volti per memorizzare le immagini dei badge scansionate utilizzando l'operazione `IndexFaces`, che estrae i volti e li memorizza come vettori di immagini con possibilità di ricerca. Quindi, quando un dipendente entra nell'edificio, un'immagine del suo volto

viene acquisita e inviata all'operazione `SearchFacesByImage`. Se la corrispondenza del volto produce un punteggio di somiglianza sufficientemente alto (pari al 99%), è possibile autenticare il collaboratore.

Comprendere il controllo delle versioni dei modelli

Amazon Rekognition usa modelli di deep learning per eseguire il rilevamento del volto e per cercare volti nelle raccolte. Continua a migliorare la precisione dei suoi modelli in base al feedback dei clienti e ai progressi nella ricerca sul deep learning. Questi miglioramenti vengono forniti come aggiornamenti del modello. Ad esempio, con la versione 1.0 del modello, IndexFaces può indicizzare i 15 volti più grandi in un'immagine. Le versioni successive del modello consentono a `IndexFaces` di indicizzare i 100 volti di maggiori dimensioni in un'immagine.

Quando si crea una nuova raccolta, viene associata con la versione più recente del modello. Per migliorare la precisione, il modello viene aggiornato periodicamente.

Quando viene rilasciata una nuova versione del modello, avviene quanto segue:

- Le nuove raccolte create vengono associate all'ultimo modello. I volti che si aggiungono alle nuove raccolte utilizzando IndexFaces vengono rilevati utilizzando il modello più recente.
- Le raccolte esistenti continuano a utilizzare la versione del modello con cui sono state create. I vettori facciali memorizzati in queste raccolte non vengono aggiornati automaticamente all'ultima versione del modello.
- I nuovi volti aggiunti a una raccolta esistente vengono rilevati utilizzando il modello già associato alla raccolta.

Versioni differenti del modello non sono compatibili tra loro. In particolare, se un'immagine viene indicizzata in più raccolte che utilizzano versioni diverse del modello, gli identificatori del volto per gli stessi volti rilevati sono diversi. Se un'immagine viene indicizzata in più raccolte che sono associate allo stesso modello, gli identificatori del volto sono gli stessi.

L'applicazione potrebbe avere problemi di compatibilità se la gestione della raccolta non tiene conto degli aggiornamenti del modello. È possibile determinare la versione del modello utilizzato da una raccolta utilizzando il campo `FaceModelVersion` che viene restituito dalle risposte di un'operazione di raccolta (ad esempio, `CreateCollection`). È possibile ottenere la versione del modello di una raccolta esistente chiamando DescribeCollection. Per ulteriori informazioni, consulta Descrizione di una raccolta.

I vettori del volto esistenti in una raccolta non possono essere aggiornati a una versione successiva del modello. Dal momento che Amazon Rekognition non memorizza i byte dell'immagine di origine, non può reindicizzare automaticamente le immagini utilizzando una versione successiva del modello.

Per utilizzare il modello più recente su volti che sono memorizzati in una raccolta esistente, creare una nuova raccolta (CreateCollection) e reindicizzare le immagini di origine nella nuova raccolta (`Indexfaces`). È necessario aggiornare qualsiasi identificatore del volto archiviato dall'applicazione in quanto gli identificatori del volto nella nuova raccolta sono diversi dagli identificatori del volto nella vecchia raccolta. Se non si ha più bisogno della vecchia raccolta, è possibile eliminarla utilizzando DeleteCollection.

Le operazioni stateless, come ad esempio DetectFaces, utilizzano la versione più recente del modello.

Nozioni di base su Amazon Rekognition

In questa sezione vengono forniti gli argomenti per iniziare a usare Amazon Rekognition. Se non conosci Amazon Rekognition, ti consigliamo di esaminare prima i concetti e la terminologia presentati in Come funziona Amazon Rekognition.

Prima di poter utilizzare Rekognition, devi creare un account AWS e ottenere un ID account AWS. Dovrai anche creare un utente che consenta al sistema Amazon Rekognition di determinare se disponi delle autorizzazioni necessarie per accedere alle sue risorse.

Dopo aver creato i tuoi account, ti consigliamo di installare e configurare AWS CLI and AWS SDKs. Ti AWS CLI consente di interagire con Amazon Rekognition e altri servizi tramite la riga di comando, AWS SDKs mentre ti consente di utilizzare linguaggi di programmazione come Java e Python per interagire con Amazon Rekognition.

Dopo aver configurato AWS CLI e AWS SDKs, puoi dare un'occhiata ad alcuni esempi di come utilizzarli entrambi. Puoi anche visualizzare alcuni esempi di come interagire con Amazon Rekognition utilizzando la console.

La console Amazon Rekognition ti consente di utilizzare i flussi di lavoro di analisi in blocco e moderazione personalizzata. Oltre a queste funzionalità, puoi anche utilizzare la console per dimostrare diverse funzionalità di Amazon Rekognition. Tieni presente che l'uso dell'API è necessario per utilizzare la maggior parte delle funzionalità di Amazon Rekognition.

Argomenti

- Fase 1: impostazione di un account AWS e creazione di un utente
- Passaggio 2: configura AWS CLI e AWS SDKs
- Fase 3: Iniziare a utilizzare l' AWS CLI API and AWS SDK
- Fase 4: nozioni di base per iniziare a usare la console di Amazon Rekognition

Fase 1: impostazione di un account AWS e creazione di un utente

Prima di usare Amazon Rekognition per la prima volta, devi completare le attività seguenti:

1. Registrati per creare un account. AWS
2. Creare un utente.

Questa sezione della guida per sviluppatori spiega perché e come creerai un account AWS e un utente.

Argomenti

- Crea un AWS account e un utente

Crea un AWS account e un utente

AWS Accounts

Quando si effettua la registrazione ad Amazon Web Services (AWS), l'account AWS viene automaticamente registrato per tutti i servizi AWS, tra cui Amazon Rekognition. Ti vengono addebitati solo i servizi che utilizzi.

Con Amazon Rekognition, paghi solo le risorse che utilizzi.

Se sei un nuovo AWS cliente, puoi iniziare a usare Amazon Rekognition gratuitamente. Per ulteriori informazioni, consulta Piano di utilizzo gratuito AWS.

Consulta la prossima Registrati per un Account AWS sezione per le istruzioni per la creazione dell'account.

Se hai già un AWS account, salta la configurazione dell'account e crea un utente amministrativo.

Utenti

Per accedere ai servizi in AWS, come Amazon Rekognition, è necessario fornire le credenziali. In questo modo, il servizio può stabilire se si dispone delle autorizzazioni per accedere alle risorse del servizio stesso.

Puoi creare chiavi di accesso per il tuo AWS account per accedere a AWS CLI o APIs mentre l'utilizzo della console richiede la tua password. Tuttavia, è sconsigliabile accedere ad AWS utilizzando le credenziali dell'account AWS di utente root. Ti consigliamo invece di utilizzare AWS Identity and Access Management (IAM) per creare un utente amministrativo.

Puoi quindi accedere ad AWS utilizzando un URL speciale e le credenziali dell'utente amministratore.

Se hai effettuato la registrazione ad AWS senza aver creato un utente per te, puoi crearne uno mediante la console IAM. Consulta la prossima Crea un utente con accesso amministrativo sezione per le istruzioni su come creare un utente amministratore.

Registrati per un Account AWS

Se non ne hai uno Account AWS, completa i seguenti passaggi per crearne uno.

Per iscriverti a un Account AWS

1. Apri la https://portal.aws.amazon.com/billing/registrazione.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata o un messaggio di testo e ti verrà chiesto di inserire un codice di verifica attraverso la tastiera del telefono.

   Quando ti iscrivi a un Account AWS, Utente root dell'account AWSviene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

AWS ti invia un'email di conferma dopo il completamento della procedura di registrazione. In qualsiasi momento, puoi visualizzare l'attività corrente del tuo account e gestirlo accedendo a https://aws.amazon.com/e scegliendo Il mio account.

Crea un utente con accesso amministrativo

Dopo esserti registrato Account AWS, proteggi Utente root dell'account AWS AWS IAM Identity Center, abilita e crea un utente amministrativo in modo da non utilizzare l'utente root per le attività quotidiane.

Proteggi i tuoi Utente root dell'account AWS

1. Accedi AWS Management Consolecome proprietario dell'account scegliendo Utente root e inserendo il tuo indirizzo Account AWS email. Nella pagina successiva, inserisci la password.

   Per informazioni sull'accesso utilizzando un utente root, consulta la pagina Signing in as the root user della Guida per l'utente di Accedi ad AWS .
2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

   Per istruzioni, consulta Abilitare un dispositivo MFA virtuale per l'utente Account AWS root (console) nella Guida per l'utente IAM.

Crea un utente con accesso amministrativo

1. Abilita Centro identità IAM.

   Per istruzioni, consulta Abilitazione di AWS IAM Identity Center nella Guida per l'utente di AWS IAM Identity Center .

2. In IAM Identity Center, assegna l'accesso amministrativo a un utente.

   Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta Configurare l'accesso utente con l'impostazione predefinita IAM Identity Center directory nella Guida per l'AWS IAM Identity Center utente.

Accesso come utente amministratore

- Per accedere con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

  Per informazioni sull'accesso utilizzando un utente IAM Identity Center, consulta AWS Accedere al portale di accesso nella Guida per l'Accedi ad AWS utente.

Assegna l'accesso a ulteriori utenti

1. In IAM Identity Center, crea un set di autorizzazioni conforme alla best practice dell'applicazione di autorizzazioni con il privilegio minimo.

   Segui le istruzioni riportate nella pagina Creazione di un set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .

2. Assegna al gruppo prima gli utenti e poi l'accesso con autenticazione unica (Single Sign-On).

   Per istruzioni, consulta Aggiungere gruppi nella Guida per l'utente di AWS IAM Identity Center .

Passaggio 2: configura AWS CLI e AWS SDKs

Argomenti

- Concessione dell'accesso programmatico
- Usare Rekognition con un SDK AWS

I passaggi seguenti mostrano come installare AWS Command Line Interface (AWS CLI) e come vengono utilizzati AWS SDKs gli esempi in questa documentazione. Esistono diversi modi per autenticare le chiamate AWS SDK. Gli esempi di questa guida presuppongono che tu stia utilizzando un profilo di credenziali predefinito per chiamare AWS CLI comandi e operazioni dell'API AWS SDK.

Per un elenco delle AWS regioni disponibili, consulta Regioni ed endpoint in. Riferimenti generali di Amazon Web Services

Segui la procedura per scaricare e configurare AWS SDKs.

Per configurare il AWS CLI e il AWS SDKs

1. Scarica e installa AWS CLIil file AWS SDKs che desideri utilizzare. Questa guida fornisce esempi per Java AWS CLI, Python, Ruby, Node.js, PHP, .NET e. JavaScript Per informazioni sull'installazione AWS SDKs, consulta Tools for Amazon Web Services.
2. Creare una chiave di accesso per l'utente creato in Crea un AWS account e un utente.
   a. Accedi AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
   b. Nel pannello di navigazione, seleziona Utenti.
   c. Scegliere il nome dell'utente creato in Crea un AWS account e un utente.
   d. Selezionare la scheda Security Credentials (Credenziali di sicurezza).
   e. Selezionare Create access key (Crea chiave di accesso). Quindi, selezionare Download .csv file (Scarica file .csv) per salvare l'ID chiave di accesso e la chiave di accesso segreta in un file CSV sul computer. Archiviare il file in un posto sicuro. Non sarà possibile accedere nuovamente alla chiave di accesso segreta dopo la chiusura di questa finestra di dialogo. Dopo aver scaricato il file CSV, selezionare Close (Chiudi).
3. Se hai installato il AWS CLI, puoi configurare le credenziali e la regione per la maggior parte AWS SDKs accedendo `aws configure` al prompt dei comandi. In caso contrario, utilizza le seguenti istruzioni.
4. Sul computer, accedere alla directory principale e creare una directory `.aws`. Nei sistemi basati su Unix, ad esempio Linux oppure macOS, si trova nella seguente posizione:

   ```
   ~/.aws
   ```

   In Windows, si trova nella seguente posizione:

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

5. Nella directory `.aws` creare un nuovo file denominato `credentials`.
6. Aprire il file CSV delle credenziali creato nella fase 2 e copiare il contenuto nel file `credentials` utilizzando il seguente formato:

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

   Sostituire l'ID della chiave di accesso e la chiave di accesso segreta in your_access_key_id e in your_secret_access_key.
7. Salvare il file `Credentials` ed eliminare il file CSV.
8. Nella directory `.aws` creare un nuovo file denominato `config`.
9. Aprire il file `config` e inserire la regione nel seguente formato.

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

   Sostituire la regione AWS preferita (ad esempio `us-west-2`) in your_aws_region.

   > **Note**
   >
   > Se non selezioni una regione, la regione us-east-1 sarà quella predefinita.

10. Salvare il file `config`.

Concessione dell'accesso programmatico

Puoi eseguire gli esempi di codice AWS CLI e contenuti in questa guida sul tuo computer locale o in altri AWS ambienti, come un'istanza Amazon Elastic Compute Cloud. Per eseguire gli esempi, devi concedere l'accesso alle operazioni AWS SDK utilizzate dagli esempi.

Argomenti

- Eseguire il codice su un computer locale
- Esecuzione di codice in ambienti AWS

Eseguire il codice su un computer locale

Per eseguire codice su un computer locale, si consiglia di utilizzare credenziali a breve termine per concedere a un utente l'accesso alle operazioni AWS SDK. Per informazioni specifiche sull'esecuzione degli esempi di codice AWS CLI and su un computer locale, consulta. Utilizzo di un profilo su un computer locale

Gli utenti necessitano dell'accesso programmatico se desiderano interagire con l' AWS AWS Management Console esterno di. Il modo per concedere l'accesso programmatico dipende dal tipo di utente che accede. AWS

Per fornire agli utenti l'accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell'accesso programmatico?	Per	Come
Identità della forza lavoro (Utenti gestiti nel centro identità IAM)	Utilizza credenziali temporane e per firmare le richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfa ccia che desideri utilizzare. • Per la AWS CLI, vedere Configurazione dell'uso AWS IAM Identity Center nella AWS CLI Guida per l'utente.AWS Command Line Interface • Per AWS SDKs gli strumenti e AWS APIs, consulta l'autenticazione di IAM Identity Center nella Guida di riferimento AWS SDKs and Tools.
IAM	Utilizza credenziali temporane e per firmare le richieste programmatiche a AWS CLI, AWS SDKs, o. AWS APIs	Seguendo le istruzioni riportate in Utilizzo delle credenziali temporanee con le AWS risorse nella Guida per l'utente IAM.

Quale utente necessita dell'accesso programmatico?	Per	Come
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfaccia che desideri utilizzare. • Per la AWS CLI, consulta Autenticazione tramite credenziali utente IAM nella Guida per l'utente.AWS Command Line Interface • Per gli strumenti AWS SDKs e gli strumenti, consulta Autenticazione tramite credenziali a lungo termine nella Guida di riferimento agli strumenti e agli AWS SDKs strumenti. • Per AWS APIs, consulta la sezione Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM.

Utilizzo di un profilo su un computer locale

Puoi eseguire gli esempi di codice AWS CLI e contenuti in questa guida con le credenziali a breve termine che crei. Eseguire il codice su un computer locale Per ottenere le credenziali e altre informazioni sulle impostazioni, gli esempi utilizzano un profilo denominato `profile-name` Ad esempio:

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

L'utente rappresentato dal profilo deve disporre delle autorizzazioni per chiamare le operazioni di Rekognition SDK e AWS altre operazioni SDK richieste dagli esempi.

Per creare un profilo che funzioni con gli esempi di codice AWS CLI and, scegli una delle seguenti opzioni. Assicurati che il nome del profilo che crei sia `profile-name`.

- Utenti gestiti da IAM — segui le istruzioni in Passaggio a un ruolo IAM (AWS CLI).
- Identità della forza lavoro (utenti gestiti da AWS IAM Identity Center): segui le istruzioni in Configurazione dell'interfaccia a riga di comando di AWS per l'uso. AWS IAM Identity Center Per gli esempi di codice, consigliamo di utilizzare un ambiente di sviluppo integrato (IDE), che supporta AWS Toolkit che abilita l'autenticazione tramite IAM Identity Center. Per gli esempi in Java, consulta Inizia a creare con Java. Per gli esempi in Python, consulta Inizia a creare con Python. Per ulteriori informazioni, consulta Credenziali IAM Identity.

> **Note**
>
> È possibile utilizzare il codice per ottenere credenziali a breve termine. Per ulteriori informazioni, consulta Passaggio a un ruolo IAM (AWS API). Per IAM Identity Center, ottieni le credenziali a breve termine per un ruolo seguendo le istruzioni in Ottenere le credenziali di ruolo IAM per l'accesso alla CLI.

Esecuzione di codice in ambienti AWS

Non è necessario utilizzare le credenziali utente per firmare chiamate AWS SDK in AWS ambienti, ad esempio codice di produzione in esecuzione in una AWS Lambda funzione. Al contrario, devi configurare un ruolo che definisce le autorizzazioni necessarie per il codice. Quindi assegnate il ruolo all'ambiente in cui viene eseguito il codice. Il modo in cui si assegna il ruolo e si rendono disponibili le credenziali temporanee varia a seconda dell'ambiente in cui viene eseguito il codice:

- AWS Lambda funzione: utilizza le credenziali temporanee che Lambda fornisce automaticamente alla funzione quando assume il ruolo di esecuzione della funzione Lambda. Le credenziali sono disponibili nelle variabili di ambiente Lambda. Non è necessario specificare un profilo. Per ulteriori informazioni, consulta Ruolo di esecuzione Lambda.
- Amazon EC2 : utilizza il provider di credenziali endpoint per i metadati delle EC2 istanze Amazon. Il provider genera e aggiorna automaticamente le tue credenziali utilizzando il profilo dell' EC2 istanza Amazon che colleghi all'istanza Amazon EC2 . Per ulteriori informazioni, consulta Utilizzo di un ruolo IAM per concedere autorizzazioni alle applicazioni in esecuzione su istanze Amazon EC2
- Amazon Elastic Container Service — utilizza il provider di credenziali Container. Amazon ECS invia e aggiorna le credenziali a un endpoint di metadati. Un ruolo IAM dell'attività da te specificato

fornisce una strategia per la gestione delle credenziali utilizzate dall'applicazione. Per ulteriori informazioni, consulta la pagina relativa Interagire con i servizi AWS.

Per ulteriori informazioni sui provider di credenziali, consulta Fornitori di credenziali standardizzati.

Usare Rekognition con un SDK AWS

AWS i kit di sviluppo software (SDKs) sono disponibili per molti linguaggi di programmazione più diffusi. Ogni SDK fornisce un'API, esempi di codice, e documentazione che facilitano agli sviluppatori la creazione di applicazioni nel loro linguaggio preferito.

Documentazione sugli SDK	Esempi di codice
AWS SDK per C++	AWS SDK per C++ esempi di codice
AWS CLI	AWS CLI esempi di codice
AWS SDK per Go	AWS SDK per Go esempi di codice
AWS SDK per Java	AWS SDK per Java esempi di codice
AWS SDK per JavaScript	AWS SDK per JavaScript esempi di codice
AWS SDK per Kotlin	AWS SDK per Kotlin esempi di codice
AWS SDK per .NET	AWS SDK per .NET esempi di codice
AWS SDK per PHP	AWS SDK per PHP esempi di codice
AWS Strumenti per PowerShell	AWS Strumenti per PowerShell esempi di codice
AWS SDK per Python (Boto3)	AWS SDK per Python (Boto3) esempi di codice
AWS SDK per Ruby	AWS SDK per Ruby esempi di codice
AWS SDK for Rust	AWS SDK for Rust esempi di codice
SDK AWS per SAP ABAP	SDK AWS per SAP ABAP esempi di codice
SDK AWS per Swift	SDK AWS per Swift esempi di codice

Per esempi specifici relativi a Rekognition, consulta Esempi di codice per l'utilizzo di Amazon Rekognition AWS SDKs.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Provide feedback (Fornisci un feedback) nella parte inferiore di questa pagina.

Fase 3: Iniziare a utilizzare l' AWS CLI API and AWS SDK

Dopo aver configurato la AWS CLI e AWS SDKs che desideri utilizzare, puoi creare applicazioni che utilizzano Amazon Rekognition. La maggior parte delle interazioni con Amazon Rekognition avviene utilizzando le operazioni API, sebbene alcune funzionalità di Amazon Rekognition prevedano flussi di lavoro da console.

I seguenti argomenti mostrano come iniziare a usare Amazon Rekognition Image e Amazon Rekognition Video tramite o il. AWS CLI AWS SDKs

- Lavorare con le immagini— Copre il processo di analisi delle immagini con Amazon Rekognition Image.
- Utilizzo delle operazioni di analisi video archiviate- Copre il processo di analisi dei video archiviati e non in streaming con Amazon Rekognition Video.
- Utilizzo degli eventi video in streaming- Copre il processo di analisi dei video in streaming con Amazon Rekognition Video.

Le sezioni sopra elencate contengono esempi che utilizzano il. AWS CLI Se intendi utilizzare il AWS CLI, consulta la sezione seguente per informazioni su come formattare le chiamate API.

Formattazione degli esempi AWS CLI

Gli AWS CLI esempi in questa guida sono formattati per il sistema operativo Linux. Per usare gli esempi con Microsoft Windows, occorre modificare la formattazione JSON del parametro `--image` e modificare le interruzioni riga da barre rovesciate (\) a caret (^). Per ulteriori informazioni sulla formattazione per JSON, consulta Specifica di valori di parametri per l'interfaccia a riga di comando di AWS.

Di seguito è riportato un AWS CLI comando di esempio formattato per Microsoft Windows (si noti che questi comandi non verranno eseguiti così come sono, sono solo esempi di formattazione):

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

È inoltre possibile fornire una versione abbreviata del parametro JSON che funziona sia su Microsoft Windows che su Linux.

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

Per ulteriori informazioni, consulta Utilizzo della sintassi abbreviata con l'interfaccia a riga di comando di AWS.

Approfondimenti

Fase 4: nozioni di base per iniziare a usare la console di Amazon Rekognition

Fase 4: nozioni di base per iniziare a usare la console di Amazon Rekognition

La console Amazon Rekognition consente di gestire le risorse relative alle etichette personalizzate Rekognition e alla funzione di moderazione personalizzata. La console fornisce solo demo di altre funzionalità di Rekognition.

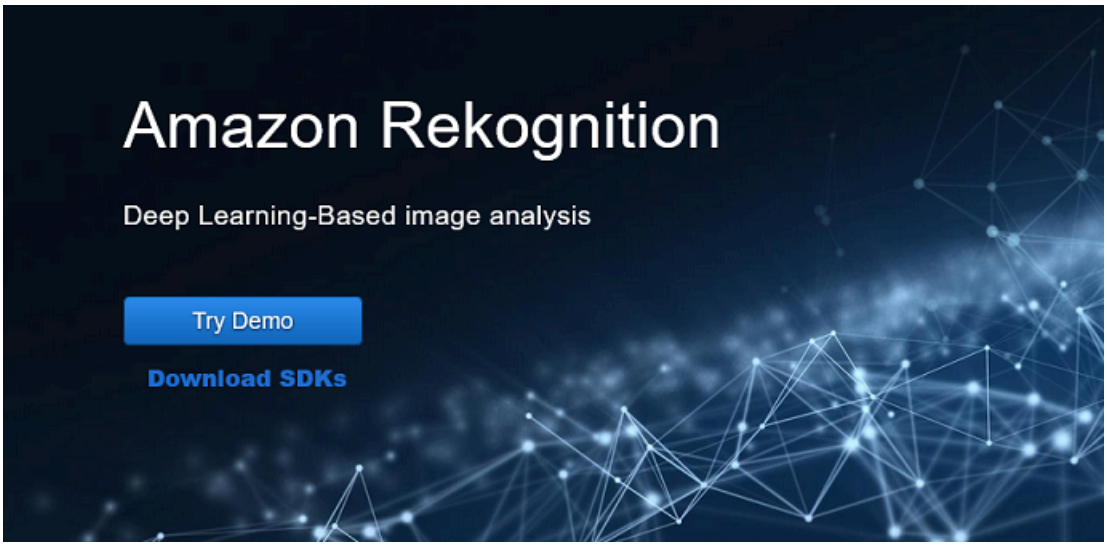
Questa sezione illustra come utilizzare un sottoinsieme di funzionalità di Amazon Rekognition come il rilevamento di oggetti e scene, l'analisi dei volti e il confronto di volti in un gruppo di immagini. Per ulteriori informazioni, consulta Come funziona Amazon Rekognition. Puoi anche utilizzare l' AWS CLI API Amazon Rekognition per rilevare oggetti e scene, rilevare volti, confrontare e cercare volti. Per ulteriori informazioni, consulta Fase 3: Iniziare a utilizzare l' AWS CLI API and AWS SDK.

Questa sezione mostra anche come visualizzare i CloudWatch parametri aggregati di Amazon per Rekognition utilizzando la console Rekognition.

Argomenti

- Configurare le autorizzazioni della console
- Esercizio 1: rilevare oggetti e scene (console)

- Esercizio 2: analizzare i volti in un'immagine (console)
- Esercizio 3: confrontare i volti nelle immagini (console)
- Esercizio 4: visualizzare i parametri aggregati (console)



Configurare le autorizzazioni della console

Per utilizzare la console Rekognition è necessario disporre delle autorizzazioni appropriate per il ruolo o l'account che accede alla console. Per alcune operazioni, Rekognition creerà automaticamente un bucket Amazon S3 per archiviare i file gestiti durante l'operazione. Se desideri archiviare i file di training in un bucket diverso da questo bucket di console, avrai bisogno di autorizzazioni aggiuntive.

Consentire l'accesso alla console

Per utilizzare la console Rekognition, puoi utilizzare una policy IAM come la seguente, che copre Amazon S3 e la console Rekognition. Per informazioni sull'assegnazione delle autorizzazioni, consulta Assegnazione delle autorizzazioni.

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

Accesso a bucket Amazon S3 esterno

Quando apri per la prima volta la console Rekognition in una nuova AWS regione, Rekognition crea un bucket (bucket della console) che viene utilizzato per archiviare i file di progetto. In alternativa, puoi usare il tuo bucket Amazon S3 (bucket esterno) per caricare le immagini o il file manifest sulla console. Per utilizzare un bucket esterno, aggiungi il seguente blocco di policy alla policy precedente.

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

Assegnare le autorizzazioni

Per fornire l'accesso, aggiungi autorizzazioni agli utenti, gruppi o ruoli:

- Utenti e gruppi in Centro identità AWS IAM (successore di AWS Single Sign-On):

  Crea un set di autorizzazioni. Segui le istruzioni in Creare un set di autorizzazioni nella Guida per l'utente del Centro identità AWS IAM (successore di AWS Single Sign-On).

- Utenti gestiti in IAM tramite un provider di identità:

  Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate nella pagina Creating a role for a third-party identity provider (federation) (Creazione di un ruolo per un provider di identità di terze parti [federazione]) nella Guida per l'utente di IAM.

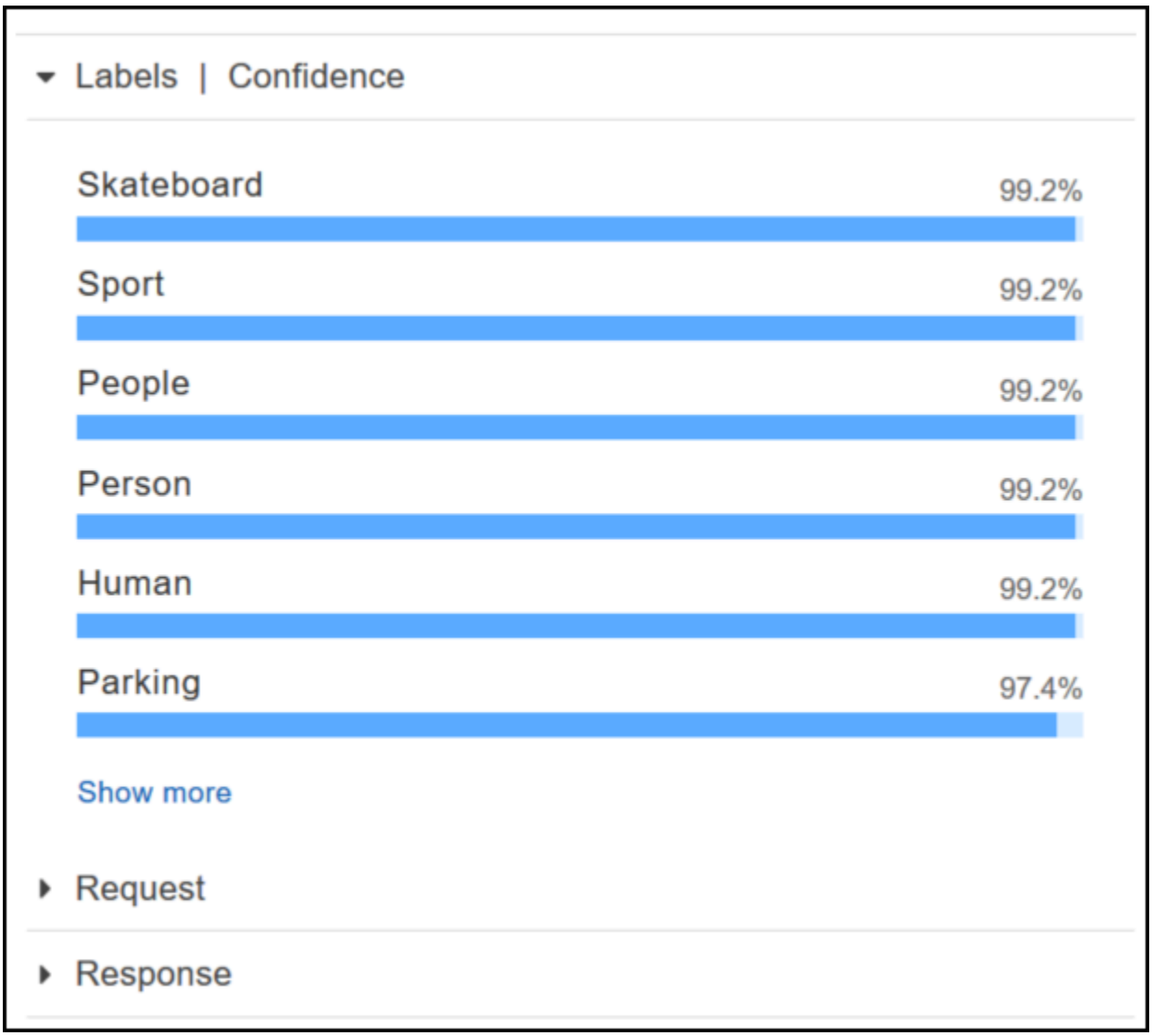
- Utenti IAM:
  - Crea un ruolo che l'utente possa assumere. Per istruzioni, consulta la pagina Creating a role for an IAM user (Creazione di un ruolo per un utente IAM) nella Guida per l'utente di IAM.
  - (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate nella pagina Aggiunta di autorizzazioni a un utente (console) nella Guida per l'utente IAM.

Esercizio 1: rilevare oggetti e scene (console)

Questa sezione mostra a livello generale la funzionalità di rilevamento di oggetti e scene in Amazon Rekognition. Quando si specifica un'immagine come input, il servizio rileva gli oggetti e le scene nell'immagine e li restituisce insieme a una percentuale di affidabilità per ogni oggetto e scena.

Ad esempio, Amazon Rekognition rileva i seguenti oggetti e scene nell'immagine di esempio: skateboard, sport, persona, auto, vettura e veicolo.

Amazon Rekognition restituisce inoltre un punteggio di affidabilità per ogni oggetto rilevato nell'immagine di esempio, come mostrato nella risposta di esempio seguente.



Per visualizzare tutti i punteggi di affidabilità mostrati in questa risposta, scegliere Show more (Mostra altro) nel riquadro Labels | Confidence (Etichette | Affidabilità).

È inoltre possibile esaminare la richiesta all'API e la risposta dall'API come riferimento.

Richiesta

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

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

Risposta

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


Per ulteriori informazioni, consulta Come funziona Amazon Rekognition.

Rilevare oggetti e scene in un'immagine fornita

È possibile caricare un'immagine personale o fornire l'URL di un'immagine come input nella console di Amazon Rekognition. Amazon Rekognition restituisce l'oggetto e le scene, i punteggi di affidabilità per ogni oggetto e la scena rilevata nell'immagine fornita.

> **Note**
>
> L'immagine deve essere inferiore a 5 MB e in formato JPEG o PNG.

Per rilevare gli oggetti e le scene in un'immagine fornita

1. Apri la console Amazon Rekognition all'indirizzo. https://console.aws.amazon.com/rekognition/

2. Scegli Label detection.
3. Esegui una di queste operazioni:
   - Caricare un'immagine - Scegliere Upload (Carica), passare al percorso in cui è stata archiviata l'immagine, quindi selezionarla.
   - Utilizzare un URL - Digitare l'URL nella casella di testo, quindi scegliere Go (Vai).
4. Visualizzare il punteggio di affidabilità di ogni etichetta rilevata nel riquadro Labels | Confidence (Etichette | Affidabilità).

Per ulteriori opzioni di analisi delle immagini, consultathe section called “Lavorare con le immagini”.

Rilevamento di persone e oggetti in un video

Puoi caricare un video fornito come input nella console Amazon Rekognition. Amazon Rekognition restituisce le persone, gli oggetti e le etichette rilevati nel video.

> **Note**
>
> Il video dimostrativo non deve durare più di un minuto o superare i 30 MB. Deve essere in formato MP4 file e codificato utilizzando il codec H.264.

Per rilevare oggetti e persone in un video fornito

1. Apri la console Amazon Rekognition all'indirizzo. https://console.aws.amazon.com/rekognition/
2. Scegli Stored Video Analysis dalla barra di navigazione.
3. In Scegli un campione o carica il tuo, seleziona Il tuo video dal menu a discesa.
4. Trascina e rilascia il video o selezionalo dalla posizione in cui lo hai archiviato.

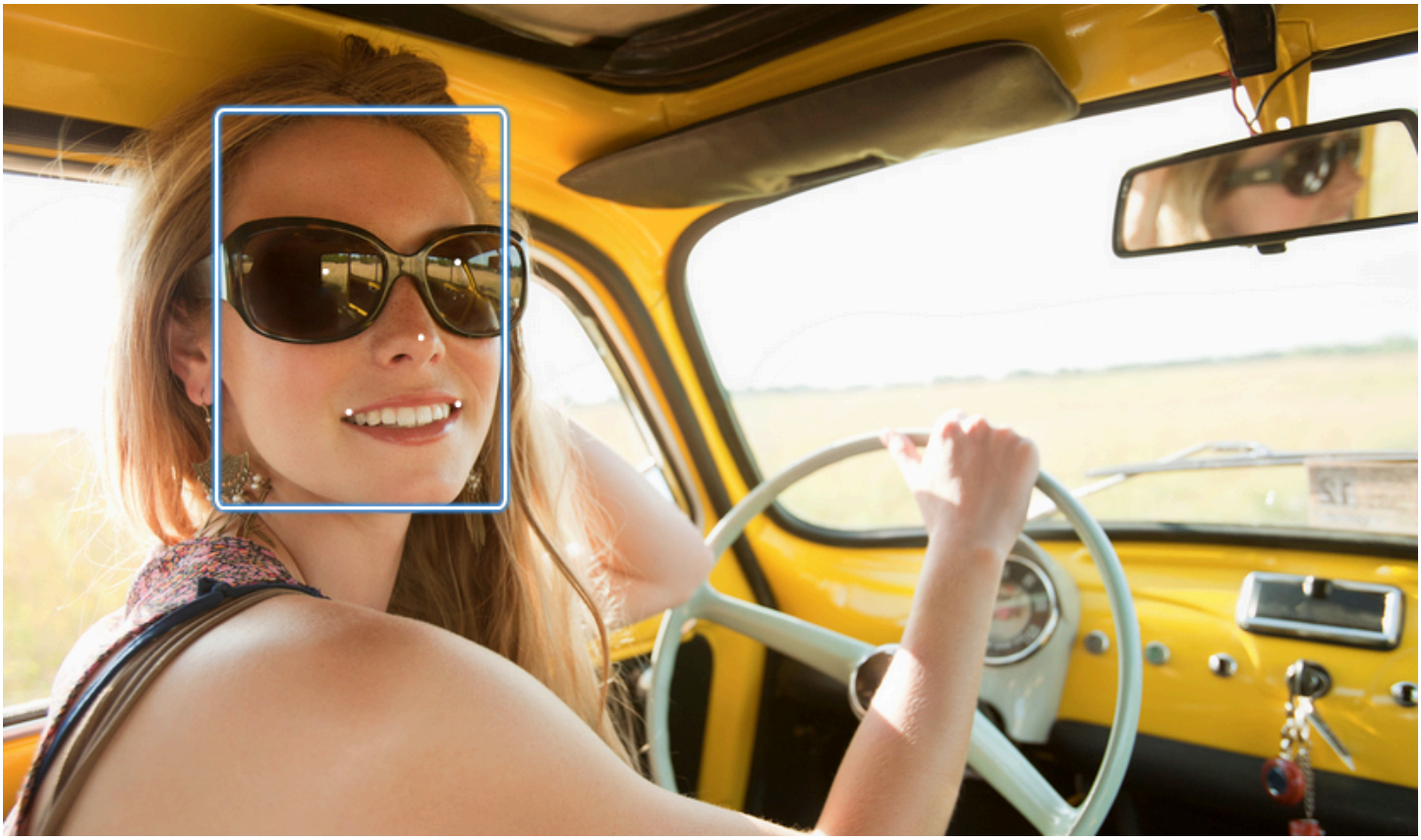
Per ulteriori opzioni di analisi dei video, consulta the section called “Utilizzo delle operazioni di analisi video archiviate” o the section called “Utilizzo degli eventi video in streaming”.

Esercizio 2: analizzare i volti in un'immagine (console)

Questa sezione illustra come utilizzare la console Amazon Rekognition per rilevare i volti e analizzarne gli attributi in un'immagine. In un'immagine che contiene un volto come input, il servizio

rileva il volto nell'immagine, ne analizza gli attributi e restituisce una percentuale di affidabilità per il volto e gli attributi rilevati nell'immagine. Per ulteriori informazioni, consulta Come funziona Amazon Rekognition.

Ad esempio, se si sceglie la seguente immagine di esempio come input, Amazon Rekognition la rileva come volto e restituisce punteggi di affidabilità per il volto e gli attributi rilevati.



Di seguito è illustrata la risposta di esempio.

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

Se nell'immagine di esempio sono presenti più volti, Rekognition rileva fino a 100 volti nell'immagine. Ogni volto rilevato viene contrassegnato con un quadrato. Quando si fa clic sull'area contrassegnata con un quadrato su un volto, Rekognition visualizza il punteggio di affidabilità del volto e dei relativi attributi rilevati nel riquadro Volti | Affidabilità

Analizzare volti in un'immagine fornita

È possibile caricare un'immagine personale o fornire l'URL dell'immagine nella console di Amazon Rekognition.

> **Note**
>
> L'immagine deve essere inferiore a 5 MB e in formato JPEG o PNG.

Per analizzare i volti in un'immagine fornita

1. Apri la console Amazon Rekognition all'indirizzo. https://console.aws.amazon.com/rekognition/
2. Scegliere Facial analysis (Analisi volto).
3. Esegui una di queste operazioni:
   - Caricare un'immagine - Scegliere Upload (Carica), passare al percorso in cui è stata archiviata l'immagine, quindi selezionarla.
   - Utilizzare un URL - Digitare l'URL nella casella di testo, quindi scegliere Go (Vai).
4. Visualizzare il punteggio di affidabilità di uno dei volti rilevati e degli attributi associati nel riquadro Faces | Confidence (Volti | Affidabilità).
5. Se nell'immagine sono presenti più volti, scegliere uno degli altri volti per visualizzarne attributi e punteggi.

Esercizio 3: confrontare i volti nelle immagini (console)

Questa sezione illustra come utilizzare la console Amazon Rekognition per confrontare i volti in un gruppo di immagini con più volti. Quando si specifica un volto di riferimento (origine) e un'immagine di volti di confronto (destinazione), Rekognition confronta il volto più grande nell'immagine di origine (ovvero il volto di riferimento) con un massimo di 100 volti rilevati nell'immagine di destinazione (ovvero, i volti di confronto), quindi rileva il livello di corrispondenza del volto nell'origine con i volti

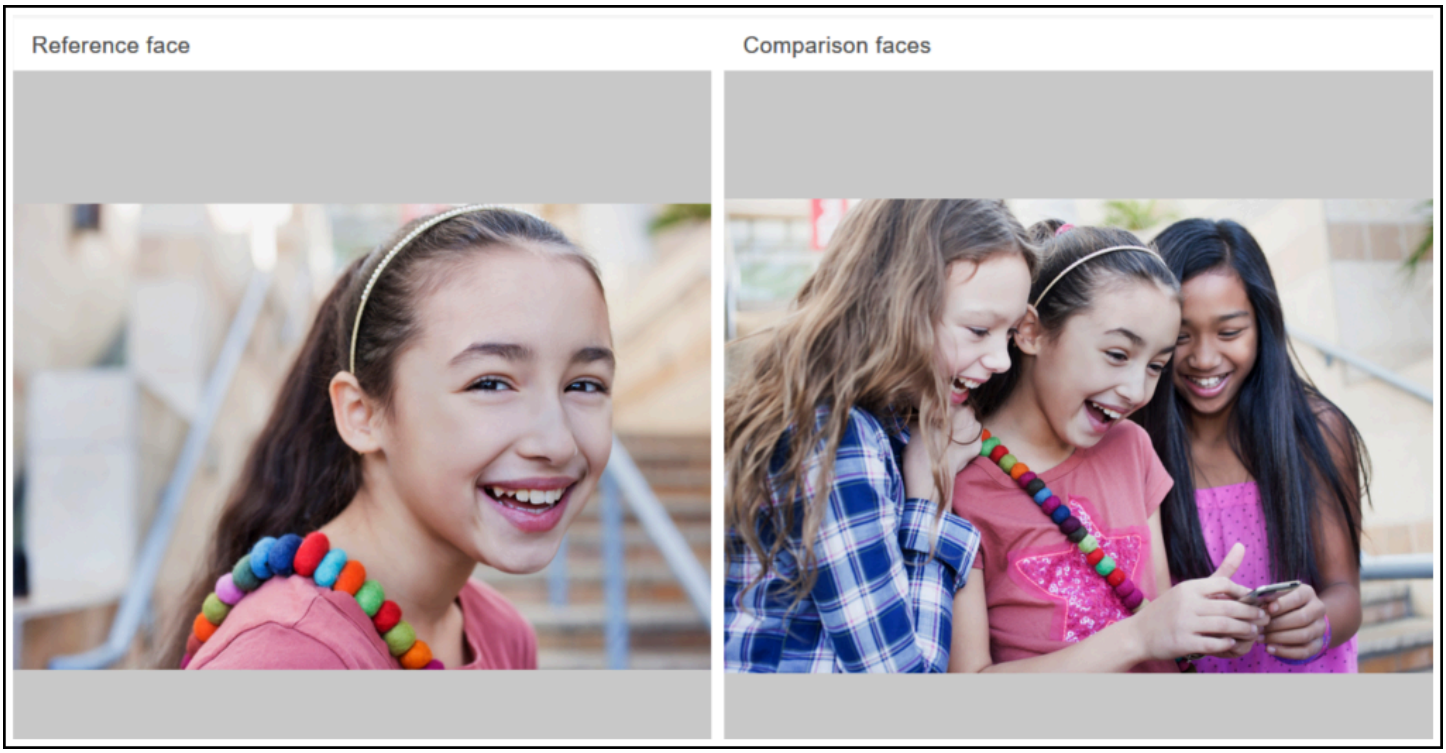
nell'immagine di destinazione. Il punteggio di somiglianza per ogni confronto viene visualizzato nel riquadro Results (Risultati).

Se l'immagine di destinazione contiene più volti, Rekognition effettua la corrispondenza del volto nell'immagine di origine con un massimo di 100 volti rilevati nell'immagine di destinazione e quindi assegna un punteggio di somiglianza a ogni corrispondenza.

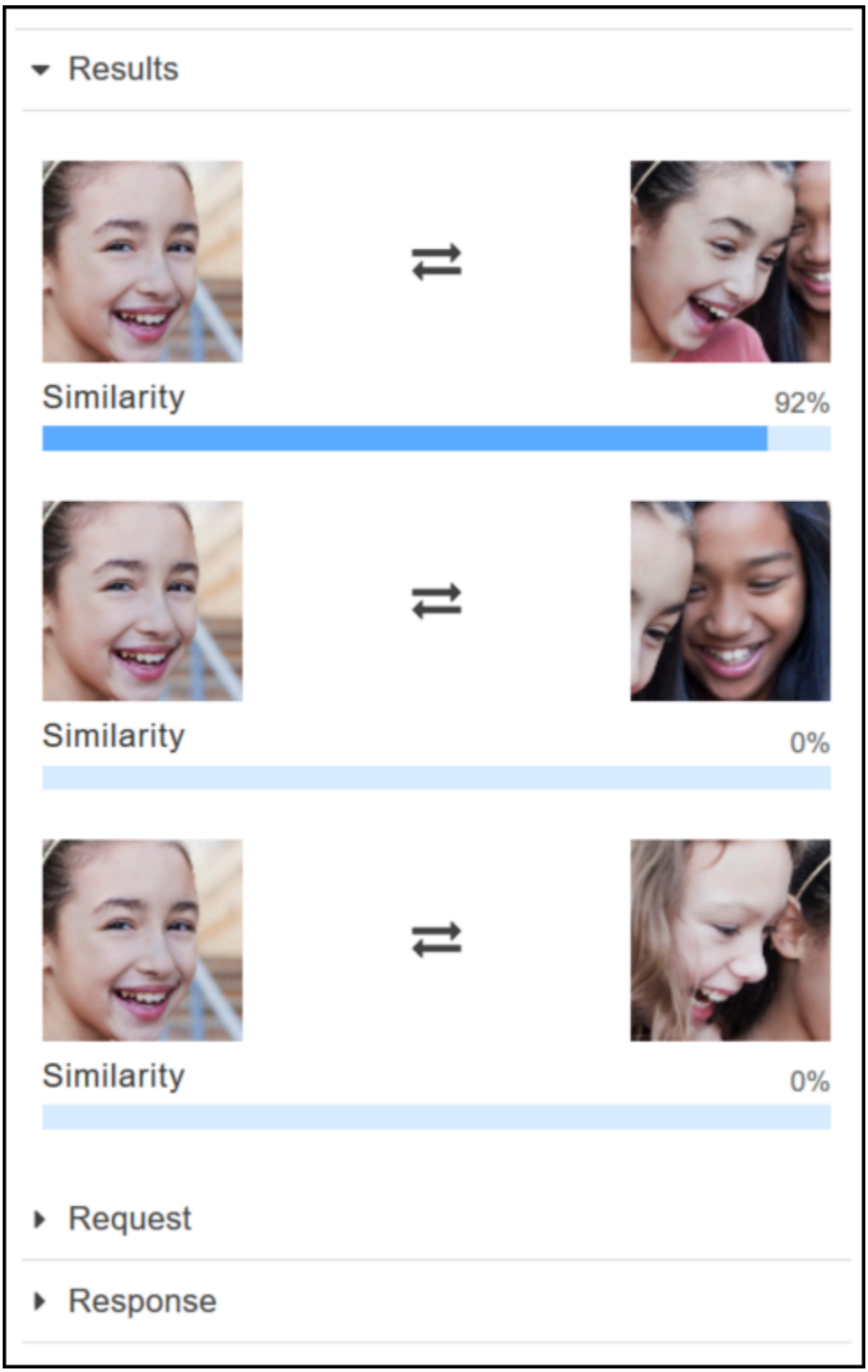
Se l'immagine di origine contiene più volti, il servizio rileva il volto più grande nell'immagine di origine e lo utilizza per confrontarlo con ogni volto rilevato nell'immagine di destinazione.

Per ulteriori informazioni, consulta Confronto dei volti nelle immagini.

Con l'immagine di esempio visualizzata sulla sinistra come immagine di origine e l'immagine di esempio a destra come immagine di destinazione, Rekognition rileva il volto nell'immagine di origine, lo confronta con ogni volto trovato nell'immagine di destinazione e visualizza un punteggio di somiglianza per ogni coppia.



La seguente immagine mostra i volti rilevati nell'immagine di destinazione e il punteggio di somiglianza per ogni volto.



Confrontare volti in un'immagine fornita

È possibile caricare immagini di origine e destinazione per Rekognition per confrontare i volti nelle immagini oppure è possibile specificare un URL per la posizione delle immagini.

> **Note**
>
> L'immagine deve essere inferiore a 5 MB e in formato JPEG o PNG.

Per confrontare i volti nelle immagini

1. Apri la console Amazon Rekognition all'indirizzo. https://console.aws.amazon.com/rekognition/
2. Scegliere Face comparison (Confronto volti).
3. Per l'immagine di origine, procedere in uno dei seguenti modi:
   - Caricare un'immagine - Scegliere Upload (Carica) a sinistra, passare al percorso in cui è stata archiviata l'immagine di origine, quindi selezionarla.
   - Utilizzare un URL - Digitare l'URL dell'immagine di origine nella casella di testo, quindi scegliere Go (Vai).
4. Per l'immagine di destinazione, procedere in uno dei seguenti modi:
   - Caricare un'immagine - Scegliere Upload (Carica) a destra, passare al percorso in cui è stata archiviata l'immagine di origine, quindi selezionarla.
   - Utilizzare un URL - Digitare l'URL dell'immagine di origine nella casella di testo, quindi scegliere Go (Vai).
5. Rekognition esegue la corrispondenza del volto più grande nell'immagine di origine con un massimo di 100 volti nell'immagine di destinazione e quindi visualizza il punteggio di somiglianza per ogni coppia nel riquadro Risultati.

Esercizio 4: visualizzare i parametri aggregati (console)

Il riquadro dei parametri di Amazon Rekognition mostra i grafici delle attività per un'aggregazione di singoli parametri di Rekognition in un periodo di tempo specificato. Ad esempio, il parametro aggregato `SuccessfulRequestCount` mostra il numero totale di richieste riuscite a tutte le operazioni API di Rekognition negli ultimi sette giorni.

La tabella seguente elenca i grafici visualizzati nel riquadro dei parametri di Rekognition e il parametro di Rekognition corrispondente. Per ulteriori informazioni, consulta CloudWatchmetriche per Rekognition.

Grafico	Parametro aggregato
Chiamate andate a buon fine	SuccessfulRequestCount
Errori client	UserErrorCount
Errori server	ServerErrorCount
Con throttle	ThrottledCount
Etichette rilevate	DetectedLabelCount
Volti rilevati	DetectedFaceCount

Ogni grafico mostra i dati dei parametri accumulati raccolti in un periodo di tempo specificato. Viene inoltre visualizzato il numero totale dei dati dei parametri aggregati per il periodo di tempo. Per visualizzare i parametri per singole chiamate API, scegliere il link sotto ogni grafico.

Per consentire agli utenti di accedere al riquadro delle metriche di Rekognition, assicurati che l'utente disponga delle autorizzazioni appropriate e Rekognition. CloudWatch Ad esempio, un utente con autorizzazioni di policy `AmazonRekognitionReadOnlyAccess` e `CloudWatchReadOnlyAccess` gestite può visualizzare il riquadro dei parametri. Se un utente non dispone delle autorizzazioni appropriate, quando apre il riquadro dei parametri non visualizza alcun grafico. Per ulteriori informazioni, consulta Identity and Access Management per Amazon Rekognition.

Per ulteriori informazioni sul monitoraggio di Rekognition, vedere. CloudWatch Monitoraggio di Rekognition con Amazon CloudWatch

Per visualizzare i parametri aggregati (console)

1. Apri la console Amazon Rekognition all'indirizzo. https://console.aws.amazon.com/rekognition/
2. Nel riquadro di navigazione, seleziona Parametri.
3. Nel menu a discesa selezionare il periodo di tempo per cui si desidera visualizzare i parametri.
4. Per aggiornare i grafici, scegliere il pulsante Refresh (Aggiorna).
5. Per visualizzare le CloudWatch metriche dettagliate per una metrica aggregata specifica, scegli Visualizza i dettagli sotto il grafico delle metriche. CloudWatch

Lavorare con immagini e video

Puoi utilizzare le operazioni dell'API Amazon Rekognition con tre diversi tipi di contenuti multimediali: immagini, video archiviati e video in streaming. Questa sezione fornisce informazioni generali sulla scrittura di codice che accede ad Amazon Rekognition per elaborare i diversi tipi di file multimediali. Per indicazioni sulle migliori pratiche e considerazioni, consulta le rispettive sezioni elencate di seguito, a seconda del tipo di supporto che stai elaborando.

Altre sezioni di questa guida forniscono informazioni su tipi specifici di analisi di immagini e video, come il rilevamento dei volti.

Argomenti

- Lavorare con le immagini
- Utilizzo delle operazioni di analisi video archiviate
- Utilizzo degli eventi video in streaming
- Gestione degli errori
- Utilizzo di Amazon Rekognition come servizio autorizzato FedRamp

Lavorare con le immagini

In questa sezione vengono illustrati i tipi di analisi che Immagini Amazon Rekognition può eseguire sulle immagini.

- Rilevamento di oggetti e scene
- Rilevamento e confronto facciale
- Ricerca di volti in una raccolta
- Riconoscimento di volti celebri
- Moderazione di immagini
- Rilevamento di testo nelle immagini

Queste azioni vengono eseguite da operazioni API non basate su storage, in cui Immagini Amazon Rekognition non mantiene le informazioni rilevate dall'operazione. Nessun byte di immagine di input viene mantenuto da parte di operazioni API non basate su storage. Per ulteriori informazioni, consulta Comprensione delle operazioni delle API diverse dallo storage e dallo storage.

Immagini Amazon Rekognition può anche memorizzare metadati facciali in raccolte per il successivo recupero. Per ulteriori informazioni, consulta Ricerca di volti in una raccolta.

In questa sezione si utilizzano le operazioni API di Immagini Amazon Rekognition per analizzare le immagini memorizzate in un bucket Amazon S3 e i byte di immagine caricati dal file system locale. Questa sezione riguarda anche l'ottenimento di informazioni sull'orientamento dell'immagine da un'immagine .jpg.

Rekognition utilizza solo canali RGB per eseguire l'inferenza. AWS consiglia agli utenti di rimuovere il canale alfa prima di utilizzare un display per ispezionare visivamente (manualmente da un umano) il confronto.

Argomenti

- Specifiche dell'immagine
- Analisi di immagini archiviate in un bucket Amazon S3
- Analisi di un'immagine caricata da un file system locale
- Visualizzazione di riquadri di delimitazione
- Ottenere l'orientamento dell'immagine e le coordinate del riquadro di delimitazione

Specifiche dell'immagine

Le operazioni di Immagini Amazon Rekognition analizzano un'immagine di input in formato .jpg o .png.

È possibile passare i byte di immagine a un'operazione di Immagini Amazon Rekognition come parte della chiamata o fare riferimento a un oggetto esistente Amazon S3. Per un esempio di analisi di un'immagine memorizzata in un bucket Amazon S3, consulta Analisi di immagini archiviate in un bucket Amazon S3. Per un esempio di trasferimento dei byte di immagine in un'operazione API Immagini Amazon Rekognition, consulta Analisi di un'immagine caricata da un file system locale.

Se si utilizza HTTP e si trasferiscono i byte di immagine nell'ambito di un'operazione di Immagini Amazon Rekognition, i byte di immagine devono essere una stringa codificata in formato Base64. Se si utilizza l'SDK AWS e si trasmettono i byte di immagine nell'ambito della chiamata all'operazione API, la necessità di codificare i byte di immagine in formato Base64 dipende dalla lingua utilizzata.

Le seguenti immagini comuni codificano AWS SDKs automaticamente in base 64 e non è necessario codificare i byte dell'immagine prima di chiamare un'operazione API di Amazon Rekognition Image.

- Java
- JavaScript
- Python
- PHP

Se si utilizza un altro AWS SDK e compare un errore di formato immagine quando si richiama un'operazione API di Rekognition provare a codificare i byte di immagine in formato Base64 prima di trasmetterli a un'operazione API di Rekognition.

Se utilizzi Amazon Rekognition Image operations AWS CLI per chiamare Amazon Rekognition Image, il trasferimento di byte di immagine come parte della chiamata non è supportato. È necessario prima caricare l'immagine in un bucket Amazon S3, quindi richiamare l'operazione referenziando l'immagine caricata.

> **Note**
>
> Non è necessario che l'immagine sia codificata in formato Base64 se si trasmette un'immagine memorizzata in un `S3Object` invece dei byte di immagine.

Per informazioni su come garantire la latenza più bassa possibile per le operazioni di Immagini Amazon Rekognition, consulta Latenza delle operazioni di Immagini Amazon Rekognition.

Correzione dell'orientamento di un'immagine

In diverse operazioni API di Rekognition viene restituito l'orientamento di un'immagine analizzata. Conoscere l'orientamento dell'immagine è importante in quanto consente di riorientare le immagini per la visualizzazione. Le operazioni API di Rekognition che analizzano i volti restituiscono anche i riquadri di delimitazione relativi alla posizione dei volti all'interno di un'immagine. È possibile utilizzare i riquadri di delimitazione per visualizzare una cornice intorno a un volto su un'immagine. Le coordinate del riquadro restituite sono influenzate dall'orientamento dell'immagine e potrebbe essere necessario convertirle per visualizzare correttamente una cornice intorno a un volto. Per ulteriori informazioni, consulta Ottenere l'orientamento dell'immagine e le coordinate del riquadro di delimitazione.

Ridimensionamento dell'immagine

Durante l'analisi, Amazon Rekognition ridimensiona internamente le immagini utilizzando una serie di intervalli predefiniti che meglio si adattano a un particolare modello o algoritmo. Per questo motivo, Amazon Rekognition potrebbe rilevare un numero diverso di oggetti o fornire risultati diversi, a seconda della risoluzione dell'immagine di input. Supponiamo, ad esempio, che ti vengano mostrate due immagini. La prima immagine ha una risoluzione di 1024x768 pixel. La seconda immagine, una versione ridimensionata della prima, ha una risoluzione di 640x480 pixel. Se invii le immagini a DetectLabels, le risposte delle due chiamate a potrebbero differire `DetectLabels` leggermente.

Analisi di immagini archiviate in un bucket Amazon S3

Immagini Amazon Rekognition è in grado di analizzare le immagini archiviate in un bucket Amazon S3 o le immagini fornite come byte di immagine.

In questo argomento, utilizzi l'operazione DetectLabelsAPI per rilevare oggetti, concetti e scene in un'immagine (JPEG o PNG) archiviata in un bucket Amazon S3. È possibile trasmettere un'immagine a un'operazione API di Immagini Amazon Rekognition utilizzando il parametro di input Image. In `Image`, è necessario specificare la proprietà dell'oggetto S3Object per fare riferimento a un'immagine archiviata in un bucket S3. I byte di immagine per le immagini archiviate in bucket Amazon S3 non devono essere codificati con Base64. Per ulteriori informazioni, consulta Specifiche dell'immagine.

Richiesta di esempio

In questa richiesta JSON di esempio per `DetectLabels`, l'immagine di origine (`input.jpg`) viene caricata da un bucket Amazon S3 denominato `amzn-s3-demo-bucket`. Nota che la regione del bucket S3 contenente l'oggetto S3 deve corrispondere alla regione utilizzata per le operazioni Immagini Amazon Rekognition.

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

Gli esempi seguenti utilizzano various AWS SDKs and the AWS CLI to call. `DetectLabels` Per informazioni sulla risposta dell'operazione `DetectLabels`, consulta DetectLabels - risposta.

Per rilevare le etichette in un'immagine

1. Se non lo hai già fatto:

   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs. Assicurati di aver fornito all'utente che chiama le operazioni API le autorizzazioni appropriate per l'accesso programmatico, consulta Concessione dell'accesso programmatico per le istruzioni su come eseguire questa operazione.

2. Carica nel bucket S3 un'immagine contenente uno o più oggetti, ad esempio alberi, abitazioni e barche. L'immagine deve essere in formato .jpg o .png.

   Per le istruzioni, consulta Caricamento di oggetti in Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

3. Utilizzare i seguenti esempi per richiamare l'operazione `DetectLabels`.

   Java

   Questo esempio mostra un elenco di etichette rilevate nell'immagine di input. Sostituisci i valori `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2.

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

package com.amazonaws.samples;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.DetectLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.Label;
```

```
import com.amazonaws.services.rekognition.model.S3Object;
import java.util.List;

public class DetectLabels {

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

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


      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

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

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

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

AWS CLI

Questo esempio mostra l'output JSON dell'operazione CLI `detect-labels`. Sostituisci i valori `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
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
```

Se usi Windows, potresti dover evitare le virgolette, come mostrato nell'esempio seguente.

```
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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
//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

Questo esempio mostra le etichette rilevate nell'immagine di input. Sostituisci i valori `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

Questo esempio mostra le informazioni sulle etichette che vengono rilevate in un'immagine.

Modifica il valore di `photo` con il percorso e il nome di un file immagine archiviato in locale contenente uno o più volti celebri. Modifica il valore di `bucket` con quello del nome del bucket S3 che contiene il file immagine fornito. Modifica il valore di `REGION` con quello del nome della regione associata al tuo account. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
// 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

Questo esempio mostra un elenco di etichette rilevate nell'immagine di input. Sostituisci i valori `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2.

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

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

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

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

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

        try
        {
            DetectLabelsResponse detectLabelsResponse =
 rekognitionClient.DetectLabels(detectlabelsRequest);
```

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

Ruby

Questo esempio mostra un elenco di etichette rilevate nell'immagine di input. Sostituisci i valori `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2.

```
  # Add to your Gemfile
  # gem 'aws-sdk-rekognition'
  require 'aws-sdk-rekognition'
  credentials = Aws::Credentials.new(
     ENV['AWS_ACCESS_KEY_ID'],
     ENV['AWS_SECRET_ACCESS_KEY']
  )
  bucket = 'bucket' # the bucket name without s3://
  photo  = 'photo' # the name of file
  client   = Aws::Rekognition::Client.new credentials: credentials
  attrs = {
    image: {
      s3_object: {
        bucket: bucket,
        name: photo
      },
    },
    max_labels: 10
  }
 response = client.detect_labels attrs
 puts "Detected labels for: #{photo}"
 response.labels.each do |label|
   puts "Label:      #{label.name}"
   puts "Confidence: #{label.confidence}"
```

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

Example response

La risposta di `DetectLabels` è una matrice di etichette rilevate nell'immagine e il livello di affidabilità con cui sono state rilevate.

Quando esegui l'operazione `DetectLabels` su un'immagine, Amazon Rekognition restituisce un output simile alla risposta di esempio seguente.

La risposta mostra che l'operazione ha rilevato più etichette, tra cui Persona, Veicolo e Auto. A ogni etichetta è associato un livello di affidabilità. Ad esempio, l'algoritmo di rilevamento è sicuro al 98,991432% che l'immagine contenga una persona.

La risposta include inoltre una serie `Parents` con i predecessori dell'etichetta. Ad esempio, l'etichetta Automobile ha due etichette padre denominate Vehicle (Veicolo) e Transportation (Trasporti).

La risposta per le etichette relative a oggetti comuni include informazioni sul riquadro di delimitazione per la posizione dell'etichetta nell'immagine in input. Ad esempio, l'etichetta Persona dispone di una serie di istanze contenenti due riquadri di delimitazione che si riferiscono a due persone rilevate all'interno dell'immagine.

Il campo `LabelModelVersion` contiene il numero di versione del modello di rilevamento utilizzato da `DetectLabels`.

Per ulteriori informazioni sull'utilizzo di questa operazione `DetectLabels`, consulta Rilevamento di oggetti e concetti.


```
{

    {
    "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"
}


}
```

Analisi di un'immagine caricata da un file system locale

Immagini Amazon Rekognition è in grado di analizzare le immagini fornite come byte di immagini o immagini memorizzate in un bucket Amazon S3.

Questi argomenti forniscono esempi relativi all'invio di byte di immagine alle operazioni API di Immagini Amazon Rekognition utilizzando un file caricato da un file system locale. È possibile trasmettere byte di immagine a un'operazione API di Immagini Amazon Rekognition utilizzando il parametro di input Image. In `Image`, è necessario specificare la proprietà `Bytes` per trasmettere i byte di immagine codificati con Base64.

I byte di immagine trasmessi a un'operazione API Amazon Rekognition utilizzando il parametro di input `Bytes` devono essere codificati in formato Base64. AWS utilizza automaticamente SDKs le immagini con codifica in base 64 in questi esempi. Non è necessario codificare i byte di immagine prima di chiamare un'operazione API Amazon Rekognition. Per ulteriori informazioni, consulta Specifiche dell'immagine.

In questa richiesta JSON di esempio per `DetectLabels`, i byte di immagine di origine vengono trasmessi nel parametro di input `Bytes`.

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

I seguenti esempi utilizzano various AWS SDKs and the to call. AWS CLI `DetectLabels` Per informazioni sulla risposta dell'operazione `DetectLabels`, consulta DetectLabels - risposta.

Per un JavaScript esempio sul lato client, vedere. Usando JavaScript

Per rilevare le etichette in un'immagine locale

1. Se non lo hai già fatto:
    a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
    b. Installa e configura il AWS CLI e il. AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Utilizzare i seguenti esempi per richiamare l'operazione `DetectLabels`.

Java

L'esempio Java seguente mostra come caricare un'immagine dal file system locale e rilevare le etichette utilizzando l'operazione detectLabels dell'SDK AWS. Modifica il valore di `photo` con il percorso e il nome di un file immagine in formato .jpg o .png.

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

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

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


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


        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

        DetectLabelsRequest request = new DetectLabelsRequest()
                .withImage(new Image()
```

```
                        .withBytes(imageBytes))
                .withMaxLabels(10)
                .withMinConfidence(77F);

        try {

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

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

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

    }
}
```

Python

Il seguente esempio SDK AWS per Python mostra come caricare un'immagine dal file system locale e richiamare l'operazione detect_labels. Modifica il valore di `photo` con il percorso e il nome di un file immagine in formato .jpg o .png.

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

import boto3

def detect_labels_local_file(photo):


    client=boto3.client('rekognition')

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

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

    return len(response['Labels'])

def main():
    photo='photo'

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


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

.NET

Il seguente esempio di Java mostra come caricare un'immagine dal file system locale e rilevare le etichette utilizzando l'operazione `DetectLabels`. Modifica il valore di `photo` con il percorso e il nome di un file immagine in formato .jpg o .png.

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

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

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

        Amazon.Rekognition.Model.Image image = new
 Amazon.Rekognition.Model.Image();
```

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

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

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

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

PHP

Il seguente esempio di AWS SDK for PHP mostra come caricare un'immagine dal file system locale e DetectFacesrichiamare l'operazione dell'API. Modifica il valore di `photo` con il percorso e il nome di un file immagine in formato .jpg o .png.

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

    require 'vendor/autoload.php';

    use Aws\Rekognition\RekognitionClient;

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

    $rekognition = new RekognitionClient($options);

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


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

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

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

Ruby

Questo esempio mostra un elenco di etichette rilevate nell'immagine di input. Modifica il valore di `photo` con il percorso e il nome di un file immagine in formato .jpg o .png.

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

    # gem 'aws-sdk-rekognition'
    require 'aws-sdk-rekognition'
    credentials = Aws::Credentials.new(
       ENV['AWS_ACCESS_KEY_ID'],
       ENV['AWS_SECRET_ACCESS_KEY']
    )
    client   = Aws::Rekognition::Client.new credentials: credentials
    photo = 'photo.jpg'
    path = File.expand_path(photo) # expand path relative to the current
 directory
    file = File.read(path)
    attrs = {
      image: {
        bytes: file
      },
      max_labels: 10
    }
    response = client.detect_labels attrs
    puts "Detected labels for: #{photo}"
    response.labels.each do |label|
      puts "Label:      #{label.name}"
      puts "Confidence: #{label.confidence}"
```

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

Java V2

Questo codice è tratto dal repository degli esempi di AWS Documentation SDK. GitHub Guarda l'esempio completo qui.

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

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

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

```
   public static void main(String[] args) {
       final String usage = """
           Usage: <bucketName> <sourceImage>

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

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

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

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

   /**
    * Detects the labels in an image stored in an Amazon S3 bucket using the
Amazon Rekognition service.
    *
    * @param rekClient     the Amazon Rekognition client used to make the
detection request
    * @param bucketName    the name of the Amazon S3 bucket where the image is
stored
    * @param sourceImage   the name of the image file to be analyzed
    */
   public static void detectImageLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
       try {
           S3Object s3ObjectTarget = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceImage)
                   .build();
```

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

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

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

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

Usando JavaScript

Il seguente esempio di JavaScript pagina Web consente a un utente di scegliere un'immagine e visualizzare le età stimate dei volti rilevati nell'immagine. Le età stimate vengono restituite da una chiamata a DetectFaces.

L'immagine scelta viene caricata utilizzando la JavaScript `FileReader.readAsDataURL` funzione che codifica l'immagine in base64. Questo è utile per visualizzare l'immagine su un canvas HTML. Significa però che i byte dell'immagine non devono essere codificati prima di essere passati a un'operazione Immagini Amazon Rekognition. Questo esempio illustra come non codificare i byte dell'immagine caricata. Se i byte dell'immagine codificata non sono utili, utilizza `FileReader.readAsArrayBuffer` perché l'immagine caricata non è codificata. Ciò significa che le operazioni Immagini Amazon Rekognition possono essere chiamate senza prima decodificare i byte dell'immagine. Per vedere un esempio, consulta Utilizzo di readAsArray Buffer.

Per eseguire l'esempio JavaScript

1. Carica il codice sorgente dell'esempio in un editor.
2. Scarica l'identificatore Amazon Cognito del pool di identità. Per ulteriori informazioni, consulta Scarica l'identificatore Amazon Cognito del pool di identità.
3. Nella funzione `AnonLog` del codice di esempio, modifica `IdentityPoolIdToUse` e `RegionToUse` nei valori indicati nella fase 9 di Scarica l'identificatore Amazon Cognito del pool di identità.
4. Nella funzione `DetectFaces`, modifica `RegionToUse` con il valore utilizzato nella fase precedente.
5. Salva il codice sorgente di esempio come un file `.html`.
6. Carica il file nel tuo browser.
7. Seleziona il pulsante Cerca…, quindi scegli un'immagine che contenga uno o più volti. Viene visualizzata una tabella che contiene le età stimate per ogni faccia rilevata nell'immagine.

> **Note**
>
> L'esempio di codice seguente utilizza due script che non fanno più parte di Amazon Cognito. Per ottenere questi file, segui i collegamenti per _aws-cognito-sdk.min.js e _amazon-cognito-identity.min.js, quindi salva il testo di ciascuno di essi come file separati. `.js`

JavaScript codice di esempio

Il seguente esempio di codice utilizza JavaScript V2. Per un esempio in JavaScript V3, vedi l'esempio nel repository degli esempi di AWS Documentation SDK. GitHub

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

```
  <meta charset="UTF-8">
  <title>Rekognition</title>
</head>

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

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

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

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

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

Utilizzo di readAsArray Buffer

Il seguente frammento di codice è un'implementazione alternativa della `ProcessImage` funzione nel codice di esempio, utilizzando V2. JavaScript Viene utilizzato `readAsArrayBuffer` per caricare un'immagine ed effettuare la chiamata a `DetectFaces`. Poiché `readAsArrayBuffer` non utilizza la codifica Base64 per il file caricato, non è necessario decodificare i byte dell'immagine prima di chiamare un'operazione Immagini Amazon Rekognition.

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

Scarica l'identificatore Amazon Cognito del pool di identità

Per semplicità, l'esempio utilizza un pool di identità Amazon Cognito anonimo per fornire un accesso non autenticato all'API Immagini Amazon Rekognition. Questo potrebbe essere idoneo per le tue esigenze. Ad esempio, puoi utilizzare l'accesso non autenticato per fornire il periodo di prova o l'accesso gratuito a un sito Web, prima che gli utenti effettuino la registrazione. Per fornire l'accesso autenticato, utilizza un pool di utenti Amazon Cognito. Per ulteriori informazioni, consulta il pool di utenti Amazon Cognito.

La procedura seguente mostra come creare un pool di identità che consente l'accesso a identità non autenticate e ottenere l'identificatore del pool di identità necessario per il codice di esempio.

Per scaricare l'identificatore del pool di identità

1. Apri la console Amazon Cognito.
2. Scegli Create new identity pool (Crea un nuovo pool di identità).
3. Digita un nome per il tuo pool di identità in Identity pool name* (Nome pool di identità).
4. In Unauthenticated identites (Identità non autenticate), seleziona Enable access to unauthenticated identities (Abilita l'accesso a identità non autenticate).
5. Seleziona Create Pool (Crea pool).

6. Scegli View Details (Visualizza dettagli) e annota il nome del ruolo per le identità non autenticate.
7. Scegli Permetti.
8. In Piattaforma, scegli. JavaScript
9. In Ottieni credenziali AWS, prendere nota dei valori di `AWS.config.region` e `IdentityPooldId` che sono visualizzati nello snippet di codice.
10. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
11. Nel riquadro di navigazione, seleziona Ruoli.
12. Scegli il nome del ruolo annotato nella fase 6.
13. Nella scheda Permissions (Autorizzazioni), scegliere Attach policies (Collega policy).
14. Scegli AmazonRekognitionReadOnlyAccess.
15. Scegli Attach Policy (Collega policy).

Visualizzazione di riquadri di delimitazione

Le operazioni di Immagini Amazon Rekognition possono restituire le coordinate dei riquadri di delimitazione per gli elementi rilevati nelle immagini. Ad esempio, l'operazione DetectFaces restituisce un riquadro (BoundingBox) per ogni volto rilevato in un'immagine. Puoi utilizzare le coordinate per visualizzare un riquadro intorno agli elementi rilevati. Ad esempio, l'immagine seguente mostra un volto circondato da un riquadro.

A `BoundingBox` possiede le seguenti proprietà:

- Altezza: L'altezza del riquadro di delimitazione come rapporto rispetto all'altezza complessiva dell'immagine.
- Sinistra: La coordinata di sinistra del riquadro di delimitazione come rapporto rispetto alla larghezza complessiva dell'immagine.
- Alto: La coordinata superiore del riquadro di delimitazione come rapporto rispetto all'altezza complessiva dell'immagine.

- Larghezza: La larghezza del riquadro di delimitazione come rapporto rispetto alla larghezza complessiva dell'immagine.

Ogni BoundingBox proprietà ha un valore compreso tra 0 e 1. Il valore di ciascuna proprietà è un rapporto tra la larghezza (`Left` e `Width`) o l'altezza (`Height` e `Top`) complessiva dell'immagine. Ad esempio, se l'immagine di input è di 700 x 200 pixel e la coordinata superiore sinistra del riquadro di delimitazione è di 350 x 50 pixel, l'API restituisce un valore `Left` di 0,5 (350/700) e un valore `Top` di 0,25 (50/200).

Il seguente diagramma mostra la porzione di immagine coperta da ciascuna proprietà del riquadro.

Per visualizzare il riquadro di delimitazione con la posizione e le dimensioni corrette, è necessario moltiplicare BoundingBox i valori per la larghezza o l'altezza dell'immagine (a seconda del valore desiderato) per ottenere i valori dei pixel. I valori in pixel servono a visualizzare il riquadro di delimitazione. Ad esempio, le dimensioni in pixel dell'immagine precedente sono 608 x 588 (larghezza x altezza). I valori del riquadro di delimitazione del volto sono:

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

La posizione in pixel del riquadro di delimitazione del volto viene calcolata come segue:

```
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
```

Puoi utilizzare questi valori per visualizzare un riquadro attorno al volto.

> **Note**
>
> Un'immagine può essere orientata in diversi modi. Per visualizzare un'immagine con l'orientamento corretto, potrebbe essere necessario eseguire una rotazione. Le coordinate del riquadro di delimitazione cambiano in base all'orientamento dell'immagine. Per visualizzare un riquadro nella posizione corretta, potrebbe essere necessario traslare le coordinate. Per

ulteriori informazioni, consulta Ottenere l'orientamento dell'immagine e le coordinate del riquadro di delimitazione.

I seguenti esempi mostrano come visualizzare un riquadro di delimitazione intorno ai volti rilevati, richiamando DetectFaces. Negli esempi si presume che le immagini siano orientate a 0 gradi. Gli esempi indicano anche come scaricare l'immagine da un bucket Amazon S3.

Per visualizzare un riquadro di delimitazione

1. Se non lo hai già fatto:
   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
   b. Installa e configura il AWS CLI e il. AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Utilizzare i seguenti esempi per richiamare l'operazione `DetectFaces`.

   Java

   Modificare il valore `bucket` con quello del bucket Amazon S3 che contiene il file immagine. Modificare il valore `photo` con il nome di un file immagine (formato .jpg o .png).

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

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

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

import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.DetectFacesRequest;
```

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

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

    private static final long serialVersionUID = 1L;

    BufferedImage image;
    static int scale;
    DetectFacesResult result;

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

        result = facesResult;
        image = bufImage;


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

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

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

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

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

        }
    }


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

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

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

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

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

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

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

            BoundingBox box = face.getBoundingBox();
            float left = width * box.getLeft();
            float top = height * box.getTop();
```

```
            System.out.println("Face:");

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

        }

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

    }
}
```

Python

Modificare il valore `bucket` con quello del bucket Amazon S3 che contiene il file immagine. Modificare il valore `photo` con il nome di un file immagine (formato .jpg o .png). Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

Nota che `s3` si riferisce al client AWS SDK Amazon Amazon S3 `rekClient` e al client AWS SDK Amazon Rekognition.

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

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

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

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

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

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

```
}
```

Ottenere l'orientamento dell'immagine e le coordinate del riquadro di delimitazione

Le applicazioni che utilizzano Immagini Amazon Rekognition solitamente devono visualizzare le immagini che vengono rilevate dalle operazioni di Immagini Amazon Rekognition e i riquadri intorno ai volti rilevati. Per visualizzare correttamente un'immagine nella propria applicazione, è necessario conoscere l'orientamento dell'immagine. Potrebbe essere necessario correggere questo orientamento. Per alcuni file .jpg, l'orientamento dell'immagine è contenuto nei metadata Exif (Exchangeable image file format) dell'immagine.

Per visualizzare un riquadro intorno a un volto, è necessario disporre delle coordinate del riquadro di delimitazione del volto. Se il riquadro non è orientato correttamente, potrebbe essere necessario modificare tali coordinate. Le operazioni di riconoscimento facciale di Immagini Amazon Rekognition restituiscono le coordinate del bounding box per ogni volto rilevato, ma non stimano le coordinate per i file.jpg senza metadati Exif.

Il seguente esempio mostra come ottenere le coordinate del riquadro di delimitazione per i volti rilevati in un'immagine.

Utilizzare le informazioni in questo esempio per verificare che le immagini siano orientate correttamente e che i riquadri di delimitazione siano visualizzati in posizione corretta all'interno dell'applicazione.

Poiché il codice utilizzato per ruotare e visualizzare le immagini e i riquadri di delimitazione dipende dalla lingua e dall'ambiente utilizzati, non spiegheremo come visualizzare le immagini e i riquadri nel codice o come ottenere informazioni di orientamento dai metadati Exif.

Determinazione dell'orientamento di un'immagine

Per visualizzare correttamente un'immagine nella propria applicazione, potrebbe essere necessario ruotarla. La seguente immagine è orientata a 0 gradi ed è visualizzata correttamente.

Tuttavia, la seguente immagine è ruotata di 90 gradi in senso antiorario. Per visualizzarla correttamente, è necessario determinare l'orientamento dell'immagine e utilizzare tali informazioni nel codice per ruotare l'immagine a 0 gradi.

Alcune immagini in formato .jpg contengono le informazioni di orientamento nei metadata Exif. Se disponibili, i metadati Exif dell'immagine contengono l'orientamento. Nei metadata Exif è possibile trovare l'orientamento dell'immagine nel campo `orientation`. Anche se Immagini Amazon Rekognition identifica la presenza di informazioni sull'orientamento dell'immagine nei metadati Exif, non fornisce accesso a tali informazioni. Per accedere ai metadata Exif in un'immagine, utilizzare una libreria di terze parti o scrivere il proprio codice. Per ulteriori informazioni, consulta Exif versione 2.32.

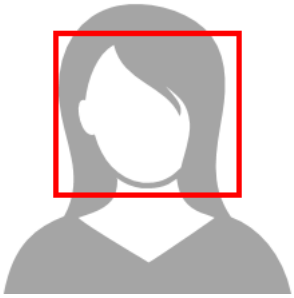
Quando si conosce l'orientamento di un'immagine, è possibile scrivere il codice per ruotarla e visualizzarla correttamente.

Visualizzazione di riquadri di delimitazione

Le operazioni di Immagini Amazon Rekognition che analizzano i volti in un'immagine restituiscono anche le coordinate dei riquadri di delimitazione che circondano i volti. Per ulteriori informazioni, consulta BoundingBox.

Per visualizzare un riquadro intorno a un volto simile a quello mostrato nell'immagine seguente all'interno dell'applicazione, utilizzare le coordinate del riquadro nel proprio codice. Le coordinate

del riquadro di delimitazione restituite da un'operazione indicano l'orientamento dell'immagine. Se è necessario ruotare l'immagine per visualizzarla correttamente, potrebbe essere necessario convertire le coordinate del riquadro di delimitazione.



Visualizzazione dei riquadri di delimitazione quando le informazioni di orientamento sono presenti nei metadata Exif

Se l'orientamento di un'immagine è inclusa nei metadata Exif, le operazioni di Immagini Amazon Rekognition eseguiranno quanto segue:

- Restituiranno un valore nullo nel campo di correzione dell'orientamento nella risposta dell'operazione. Per ruotare l'immagine, utilizzare l'orientamento fornito nei metadata Exif del proprio codice.
- Restituiranno le coordinate del riquadro di delimitazione già orientate a 0 gradi. Per visualizzare il riquadro di delimitazione nella posizione corretta, utilizzare le coordinate restituite. Non è necessario convertirle.

Esempio: ottenimento dell'orientamento dell'immagine e delle coordinate del riquadro per un'immagine

L'esempio seguente mostra come utilizzare l'AWS SDK per ottenere i dati di orientamento dell'immagine Exif e le coordinate del riquadri di delimitazione per le celebrità rilevate dall'operazione `RecognizeCelebrities`.

> **Note**
>
> Il supporto per la stima dell'orientamento dell'immagine utilizzando il campo `OrientationCorrection` è cessato ad agosto 2021. Tutti i valori restituiti per questo campo inclusi in una risposta API saranno sempre NULL.

Java

In questo esempio viene caricata un'immagine dal file system locale, richiamata l'operazione `RecognizeCelebrities`, determinata l'altezza e la larghezza dell'immagine e vengono inoltre calcolate le coordinate del riquadro di delimitazione del volto per l'immagine ruotata. L'esempio non mostra come elaborare le informazioni di orientamento memorizzate nei metadata Exif.

Nella funzione `main` sostituire il valore di `photo` con il nome e il percorso di un'immagine memorizzata localmente in formato .png o .jpg.

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

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

public class RotateImage {

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

  String photo = "photo.png";

  //Get Rekognition client
```

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


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

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

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

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

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

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

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

 //Call GetCelebrities

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


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

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

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

            System.out.println();
       }

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

}


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

  float left = 0;
  float top = 0;

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

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

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

  }
}
```

Python

In questo esempio viene utilizzata la libreria di immagini PIL/Pillow per ottenere la larghezza e l'altezza dell'immagine. Per ulteriori informazioni, consulta Pillow. In questo esempio vengono conservati i metadati exif che potrebbe essere necessari in altri punti dell'applicazione.

Nella funzione `main` sostituire il valore di `photo` con il nome e il percorso di un'immagine memorizzata localmente in formato .png o .jpg.

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

import boto3
import io
from PIL import Image
```

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

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


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

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

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

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

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

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

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

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

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

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


def main():
    photo = 'photo'

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


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

Java V2

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

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

Utilizzo delle operazioni di analisi video archiviate

Video Amazon Rekognition è un'API che puoi utilizzare per analizzare i video. Con Video Amazon Rekognition, puoi rilevare etichette, volti, persone, celebrità e contenuti per adulti (espliciti e spinti) nei video archiviati in un bucket Amazon Simple Storage Service (Amazon S3). Puoi utilizzare Video Amazon Rekognition in categorie quali media/intrattenimento e pubblica sicurezza. In passato, la ricerca di oggetti e persone all'interno dei video avrebbe richiesto molte ore di visualizzazione con rischio di errore da parte di un essere umano. Video Amazon Rekognition automatizza il rilevamento e la ricorrenza di elementi in un video.

Questa sezione illustra i tipi di analisi che Video Amazon Rekognition può eseguire, una panoramica sull'API ed esempi di utilizzo di Video Amazon Rekognition.

Argomenti

- Tipi di analisi
- Panoramica dell'API Video Amazon Rekognition
- Chiamata delle operazioni Video Amazon Rekognition
- Configurazione di Video Amazon Rekognition
- Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK)
- Analisi di un video con AWS Command Line Interface
- Riferimento: Notifica dei risultati dell'analisi video
- Risoluzione dei problemi di Video Amazon Rekognition

Tipi di analisi

Video Amazon Rekognition ti consente di analizzare i video per ottenere le informazioni seguenti:

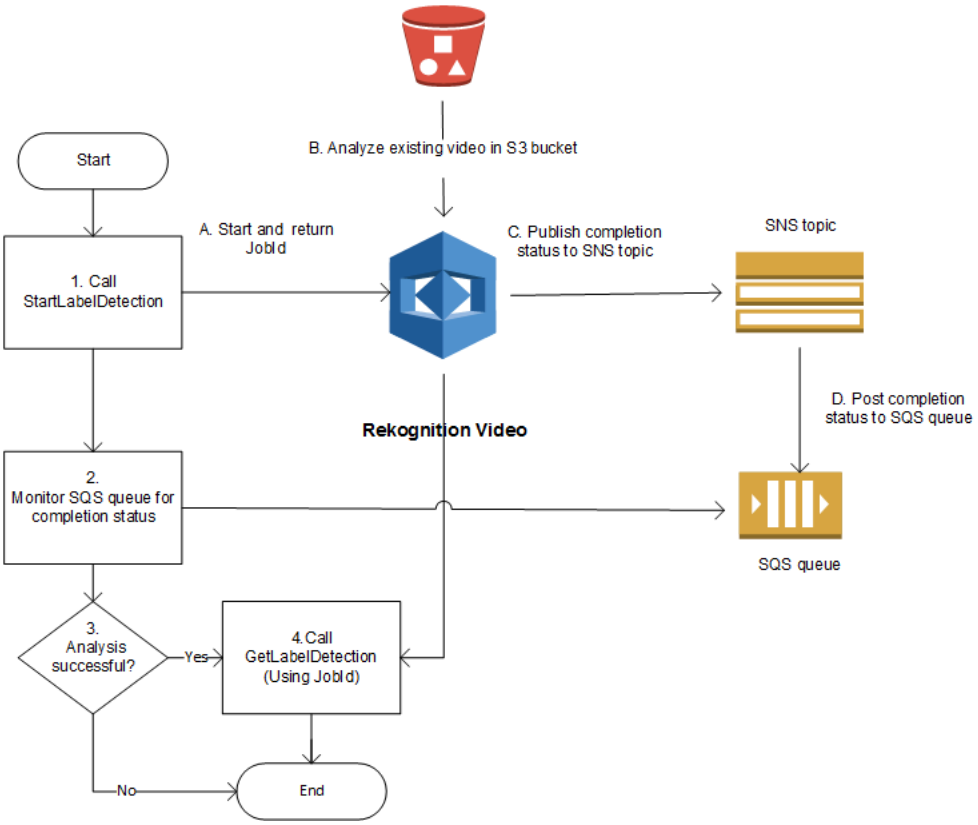
- Segmenti video
- Etichette
- Contenuti per adulti espliciti e spinti
- Text (Testo)
- Volti celebri
- Volti
- Persone

Per ulteriori informazioni, consulta Come funziona Amazon Rekognition.

Panoramica dell'API Video Amazon Rekognition

Video Amazon Rekognition elabora un video archiviato in un bucket Amazon S3. Il modello di progettazione è costituito da una serie asincrona di operazioni. L'analisi di un video viene avviata chiamando un'operazione `Start` come ad esempio StartLabelDetection. Lo stato di completamento della richiesta è pubblicato in un argomento Amazon Simple Notification Service (Amazon SNS). Per conoscere lo stato di completamento dell'argomento Amazon SNS, puoi utilizzare una coda o una funzione di Amazon Simple Queue Service (Amazon SQS). AWS Lambda Dopo aver acquisito lo stato di completamento, chiama un'operazione `Get`, come ad esempio GetLabelDetection, per ottenere i risultati della richiesta.

Il seguente diagramma mostra il processo di rilevamento delle etichette in un video archiviato in un bucket Amazon S3. Nel diagramma, una coda Amazon SQS ottiene lo stato di completamento dall'argomento Amazon SNS. In alternativa, puoi usare una funzione. AWS Lambda



Il processo è lo stesso delle altre operazioni Video Amazon Rekognition. La tabella seguente elenca le operazioni `Start` e `Get` per ciascuna delle operazioni Amazon Rekognition non basate su storage.

Rilevamento	Operazione Start	Operazione Get
Segmenti video	StartSegmentDetection	GetSegmentDetection
Etichette	StartLabelDetection	GetLabelDetection
Contenuti per adulti espliciti o spinti	StartContentModeration	GetContentModeration
Testo	StartTextDetection	GetTextDetection
Volti celebri	StartCelebrityRecognition	GetCelebrityRecognition

Rilevamento	Operazione Start	Operazione Get
Volti	StartFaceDetection	GetFaceDetection
Persone	StartPersonTracking	GetPersonTracking

Per le operazioni `Get` diverse da `GetCelebrityRecognition`, Video Amazon Rekognition restituisce informazioni di monitoraggio relative al momento in cui le entità vengono rilevate nel video di input.

Per ulteriori informazioni sull'uso dell'API Video Amazon Rekognition, consulta Chiamata delle operazioni Video Amazon Rekognition. Per un esempio in cui l'analisi video viene eseguita tramite Amazon SQS, consulta Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK). Analisi di un video con AWS Command Line InterfacePer esempi, consulta. AWS CLI

Storage e formati video

Le operazioni di Amazon Rekognition possono analizzare i video archiviati in bucket Amazon S3. Per un elenco di tutti i limiti alle operazioni di analisi delle immagini, consultareLinee guida e quote.

Il video deve essere codificato attraverso il codec H.264. I formati di file supportati sono MPEG-4 e MOV.

Un codec è un software o hardware che comprime i dati per una distribuzione più rapida e decomprime i dati ricevuti nella forma originaria. Il codec H.264 viene utilizzato in genere per la registrazione, la compressione e la distribuzione dei contenuti video. Un formato di file video può contenere uno o più codec. Se il tuo file video di formato MOV o MPEG-4 non funziona con Video Amazon Rekognition, verifica che il codec utilizzato per codificare il video sia H.264.

Qualsiasi API Video Amazon Rekognition che analizza i dati audio supporta solo codec audio AAC.

La dimensione massima di un file video archiviato è 10 GB.

Ricerca di persone

Puoi utilizzare i metadata facciali archiviati in una raccolta per cercare persone in un video. Ad esempio, puoi eseguire la ricerca di una persona specifica o di più persone in un video di archiviato. I metadata facciali delle immagini di origine vengono archiviati in una raccolta tramite l'operazione IndexFaces. Puoi quindi utilizzare StartFaceSearch per avviare la ricerca asincrona dei volti nella raccolta. Puoi utilizzare GetFaceSearch per ottenere i risultati della ricerca. Per ulteriori informazioni,

consulta Ricerca di volti in video archiviati. La ricerca di persone è un esempio di operazione di Amazon Rekognition basata su storage. Per ulteriori informazioni, consulta Operazioni API basate su storage.

Puoi inoltre cercare persone in un video in streaming. Per ulteriori informazioni, consulta Utilizzo degli eventi video in streaming.

Chiamata delle operazioni Video Amazon Rekognition

Video Amazon Rekognition è un'API asincrona che puoi utilizzare per analizzare i video archiviati in un bucket Amazon Simple Storage Service (Amazon S3). Puoi avviare l'analisi di un video chiamando un'operazione Amazon Rekognition `Start` Video, ad esempio. StartPersonTracking Video Amazon Rekognition pubblica il risultato della richiesta di analisi in un argomento Amazon Simple Notification Service (Amazon SNS). Puoi utilizzare una coda Amazon Simple Queue Service (Amazon SQS) o AWS Lambda una funzione per ottenere lo stato di completamento della richiesta di analisi video dall'argomento Amazon SNS. Infine, puoi ottenere i risultati della richiesta di analisi video chiamando un'operazione di `Get` Amazon Rekognition, ad esempio. GetPersonTracking

Le informazioni incluse nelle sezioni seguenti utilizzano le operazioni di rilevamento delle etichette per mostrare come Video Amazon Rekognition rileva etichette (oggetti, eventi, concetti e attività) in un video archiviato in un bucket Amazon S3. Lo stesso approccio funziona per le altre operazioni di Amazon Rekognition Video, ad esempio e. StartFaceDetectionStartPersonTracking L'esempio Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK) mostra come analizzare un video utilizzando una coda Amazon SQS per ottenere lo stato di completamento dall'argomento Amazon SNS. Viene utilizzato come base anche per altri esempi di Video Amazon Rekognition quali Rilevamento dei movimenti delle persone. AWS CLI Analisi di un video con AWS Command Line InterfacePer esempi, vedi.

Argomenti

- Avvio di analisi video
- Ottenere lo stato di completamento di una richiesta di analisi Video Amazon Rekognition
- Ottenere i risultati dell'analisi di Video Amazon Rekognition

Avvio di analisi video

Puoi avviare una richiesta di rilevamento delle etichette Amazon Rekognition Video chiamando. StartLabelDetection Di seguito è riportato un esempio di una richiesta JSON passata da `StartLabelDetection`.

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

Il parametro di input `Video` fornisce il nome del file video e il bucket Amazon S3 da cui recuperarlo. `NotificationChannel` contiene il nome della risorsa Amazon (ARN) dell'argomento Amazon SNS che viene notificato da Video Amazon Rekognition al termine della richiesta di analisi video. L'argomento Amazon SNS deve trovarsi nella stessa regione AWS dell'endpoint Video Amazon Rekognition che stai chiamando. `NotificationChannel` contiene inoltre l'ARN relativo a un ruolo che consente a Video Amazon Rekognition di pubblicare nell'argomento Amazon SNS. Puoi fornire autorizzazioni per la pubblicazione di Amazon Rekognition ai tuoi argomenti creando un ruolo del servizio IAM. Per ulteriori informazioni, consulta Configurazione di Video Amazon Rekognition.

Puoi inoltre specificare un parametro di input opzionale, `JobTag`, che ti consente di identificare il processo in stato di completamento pubblicato sull'argomento Amazon SNS.

Per evitare la duplicazione accidentale dei processi di analisi, puoi facoltativamente fornire un token idempotente, `ClientRequestToken`. Se specifichi un valore per `ClientRequestToken`, l'operazione `Start` restituisce lo stesso `JobId` per più chiamate identiche dell'operazione start, ad esempio `StartLabelDetection`. Un token `ClientRequestToken` ha un ciclo di vita di 7 giorni. Dopo 7 giorni, puoi riutilizzarlo. Se riutilizzi il token durante il suo ciclo di vita, si verifica quanto segue:

- Se riutilizzi il token con la stessa operazione `Start` e gli stessi parametri di input, viene restituito lo stesso `JobId`. Il processo non viene rieseguito e Video Amazon Rekognition non invia uno stato di completamento all'argomento Amazon SNS registrato.

- Se riutilizzi il token con la stessa operazione `Start` e un parametro di input secondario viene modificato, viene sollevata un'eccezione `IdempotentParameterMismatchException` (codice di stato HTTP: 400).
- Non riutilizzare un token con operazioni `Start` diverse poiché otterrai risultati imprevedibili da Amazon Rekognition.

La risposta all'operazione `StartLabelDetection` è un identificatore del processo (`JobId`). Utilizza `JobId` per tenere traccia delle richieste e ottenere i risultati dell'analisi dopo che Video Amazon Rekognition ha pubblicato lo stato di completamento nell'argomento Amazon SNS. Per esempio:

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

Se avvii troppi processi simultaneamente, le chiamate a `StartLabelDetection` sollevano un'eccezione `LimitExceededException` (codice di stato HTTP: 400) finché il numero di processi simultanei in esecuzione è inferiore al limite di servizio Amazon Rekognition.

Se riscontri che vengono sollevate eccezioni `LimitExceededException` con picchi di attività, potresti utilizzare una coda Amazon SQS per gestire le richieste in arrivo. Contatta l' AWS assistenza se scopri che il numero medio di richieste simultanee non può essere gestito da una coda Amazon SQS e continui a ricevere eccezioni. `LimitExceededException`

Ottenere lo stato di completamento di una richiesta di analisi Video Amazon Rekognition

Video Amazon Rekognition invia una notifica di completamento dell'analisi all'argomento Amazon SNS registrato. La notifica include l'identificatore del processo e lo stato di completamento dell'operazione in una stringa JSON. Una richiesta di analisi video corretta presenta uno stato `SUCCEEDED`. Ad esempio, il risultato seguente mostra l'elaborazione corretta di un processo di rilevamento delle etichette.

```
{
    "JobId": "270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1nnnnnnnnnnn",
    "Status": "SUCCEEDED",
    "API": "StartLabelDetection",
    "JobTag": "DetectingLabels",
    "Timestamp": 1510865364756,
    "Video": {
        "S3ObjectName": "video.mp4",
```

```
        "S3Bucket": "amzn-s3-demo-bucket"
    }
}
```

Per ulteriori informazioni, consulta Riferimento: Notifica dei risultati dell'analisi video.

Per ottenere le informazioni sullo stato pubblicate nell'argomento Amazon SNS da Video Amazon Rekognition utilizza una delle seguenti opzioni:

- AWS Lambda - Puoi sottoscrivere una funzione AWS Lambda scritta in un argomento Amazon SNS. La funzione viene chiamata quando Amazon Rekognition notifica all'argomento Amazon SNS che la richiesta è stata completata. Utilizza una funzione Lambda se desideri che il codice lato server elabori i risultati di una richiesta di analisi video. Ad esempio, potresti voler utilizzare il codice lato server per annotare il video o creare un report sui contenuti video prima di restituire le informazioni a un'applicazione client. Ti consigliamo inoltre di eseguire l'elaborazione lato server per i video di grandi dimensioni perché l'API Amazon Rekognition può restituire volumi di dati di grandi dimensioni.
- Amazon Simple Queue Service: È possibile effettuare la sottoscrizione di una coda Amazon SQS a un argomento SNS. Puoi quindi eseguire il polling della coda Amazon SQS per recuperare lo stato di completamento pubblicato da Amazon Rekognition al completamento di una richiesta di analisi video. Per ulteriori informazioni, consulta Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK). Utilizza una coda Amazon SQS se desideri chiamare le operazioni Video Amazon Rekognition solo da un'applicazione client.

> **Important**
>
> Non è consigliabile ottenere lo stato di completamento della richiesta chiamando ripetutamente l'operazione `Get` di Video Amazon Rekognition. Ciò si verifica perché, se vengono effettuate troppe richieste, Video Amazon Rekognition sottopone a throttling l'operazione `Get`. Se si elaborano più video simultaneamente, è più semplice ed efficiente monitorare una coda SQS per la notifica di completamento anziché eseguire il polling di Video Amazon Rekognition per lo stato di ciascun video singolarmente.

Ottenere i risultati dell'analisi di Video Amazon Rekognition

Per ottenere i risultati di una richiesta di analisi video, accertati prima che lo stato di completamento recuperato dall'argomento Amazon SNS sia `SUCCEEDED`. Quindi chiama `GetLabelDetection` che

trasferisce il valore `JobId` restituito da `StartLabelDetection`. La struttura JSON della richiesta è simile all'esempio riportato di seguito:

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

JobId è l'identificatore per l'operazione di analisi video. Poiché l'analisi video può generare grandi quantità di dati, utilizza `MaxResults` per specificare il numero massimo di risultati da restituire in una singola operazione Get. Il valore predefinito per `MaxResults` è 1000. Se si specifica un valore maggiore di 1.000, vengono restituiti al massimo 1.000 risultati. Se l'operazione non restituisce l'intera serie di risultati, viene restituito un token di paginazione per la pagina successiva nella risposta dell'operazione. Se disponi di un token di paginazione di una richiesta Get precedente, utilizzalo con `NextToken` per ottenere la pagina successiva di risultati.

> **Note**
>
> Amazon Rekognition conserva i risultati di un'operazione di analisi video per 7 giorni. Non potrai recuperare i risultati dell'analisi dopo questo periodo.

La struttura JSON della risposta dell'operazione `GetLabelDetection` è simile a quello riportato di seguito:

```
{
    "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
    }
}
```

Le operazioni `GetLabelDetection` e `GetContentModeration` consentono di ordinare i risultati dell'analisi per data e ora o per nome dell'etichetta. I risultati possono essere aggregati per timestamp o segmenti video.

Puoi ordinare i risultati per tempo di rilevamento (millisecondi dall'inizio del video) o in ordine alfabetico per entità rilevate (oggetto, volto, celebrità, etichetta di moderazione o persona). Per ordinare in base al tempo, imposta il valore del parametro di input `SortBy` su `TIMESTAMP`. Se non è specificato `SortBy`, il comportamento predefinito è quello di elencare per tempo. L'esempio precedente è stato ordinato per tempo. Per ordinare per entità, utilizza il parametro di input `SortBy` con il valore appropriato per l'operazione in esecuzione. Ad esempio, per ordinare per etichetta rilevata in una chiamata a `GetLabelDetection`, utilizza il valore `NAME`.

Per aggregare i risultati per timestamp, imposta il valore del parametro `AggregateBy` su `TIMESTAMPS`. Per aggregare per segmento video, impostate il valore di `AggregateBy` a `SEGMENTS`. La modalità di aggregazione `SEGMENTS` aggregherà le etichette nel tempo, fornendo al contempo `TIMESTAMPS` il timestamp in cui è stata rilevata un'etichetta, utilizzando un campionamento a 2 FPS e un output per fotogramma (Nota: questa frequenza di campionamento attuale è soggetta a modifiche, non si devono fare ipotesi sulla frequenza di campionamento attuale). Se nessun valore è specificato, il metodo di aggregazione predefinito è `TIMESTAMPS`.

Configurazione di Video Amazon Rekognition

Per utilizzare l'API Video Amazon Rekognition con video archiviati, è necessario configurare l'utente e un ruolo di servizio IAM per accedere agli argomenti Amazon SNS. Inoltre, è necessario sottoscrivere una coda Amazon SQS ai tuoi argomenti Amazon SNS.

> **Note**
>
> Se utilizzi queste istruzioni per configurare l'esempio Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK), non eseguire le fasi 3, 4, 5 e 6. L'esempio include il codice per creare e configurare l'argomento Amazon SNS e la coda Amazon SQS.

Gli esempi in questa sezione creano un nuovo argomento Amazon SNS utilizzando le istruzioni che forniscono a Video Amazon Rekognition l'accesso a più argomenti. Se desideri utilizzare un argomento Amazon SNS esistente, usa Accesso a un argomento Amazon SNS esistente per la fase 3.

Per configurare Video Amazon Rekognition

1. Configura un AWS account per accedere ad Amazon Rekognition Video. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
2. Installa e configura l'SDK richiesto. AWS Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
3. Per eseguire gli esempi di codice in questa guida per sviluppatori, assicurati che l'utente prescelto disponga dell'accesso programmatico. Per ulteriori informazioni, consulta Concessione dell'accesso programmatico.

   L'utente necessita inoltre almeno delle seguenti autorizzazioni:

   - SQSFullAccesso Amazon
   - AmazonRekognitionFullAccess
   - Amazon S3 FullAccess
   - SNSFullAccesso Amazon

   Se utilizzi IAM Identity Center per l'autenticazione, aggiungi le autorizzazioni al set di autorizzazioni per il tuo ruolo, altrimenti aggiungi le autorizzazioni al tuo ruolo IAM.

4. Crea un argomento Amazon SNS utilizzando la console Amazon SNS. Anteponi il nome dell'argomento con. AmazonRekognition Prendi nota del nome della risorsa Amazon (ARN) dell'argomento. Assicurati che l'argomento sia nella stessa regione dell'endpoint AWS che stai utilizzando.
5. Crea una coda Amazon SQS usando la console Amazon SQS. Prendere nota dell'ARN della coda.
6. Sottoscrivi la coda all'argomento creato nella fase 3.
7. Concedere all'argomento Amazon SNS l'autorizzazione a inviare messaggi alla coda Amazon SQS.
8. Crea un ruolo di servizio IAM per fornire a Video Amazon Rekognition l'accesso ai tuoi argomenti Amazon SNS. Prendere nota del valore nome della risorsa Amazon (ARN) del ruolo di servizio, Per ulteriori informazioni, consulta Accesso a più argomenti Amazon SNS.
9. Per garantire la sicurezza del tuo account, vorrai limitare l'ambito di accesso di Rekognition alle sole risorse che stai utilizzando. Questo può essere fatto associando una policy di attendibilità al ruolo di servizio IAM. Per informazioni su come fare, consulta Prevenzione del problema "confused deputy" tra servizi.
10. Aggiungere la seguente policy inline all'utente creato nella fase 1:

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

    Dai alla policy inline un nome a scelta.
11. Se utilizzi una AWS Key Management Service chiave gestita dal cliente per crittografare i video nel tuo bucket Amazon S3, aggiungi le autorizzazioni alla chiave che consentono al ruolo di servizio che hai creato nel passaggio 7 di decrittografare i video. Il ruolo di servizio richiede almeno l'autorizzazione per le azioni `kms:GenerateDataKey` e `kms:Decrypt`. Per esempio:

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

Per maggiori informazioni, consulta la sezione Il mio bucket Amazon S3 dispone di una crittografia predefinita che utilizza una chiave AWS KMS personalizzata. Come posso consentire agli utenti di scaricare e caricare file nel bucket? e protezione dei dati con la crittografia lato server con chiavi KMS archiviate in AWS Key Management Service (SSE-KMS).

12. È ora possibile eseguire gli esempi in Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK) e in Analisi di un video con AWS Command Line Interface.

Accesso a più argomenti Amazon SNS

Usa un ruolo di servizio IAM per fornire a Video Amazon Rekognition l'accesso agli argomenti Amazon SNS creati. IAM fornisce il caso d'uso di Rekognition per la creazione di un ruolo di servizio Video Amazon Rekognition.

Puoi consentire ad Amazon Rekognition Video di accedere a più argomenti di Amazon SNS utilizzando la policy di autorizzazione e anteponendo ai nomi `AmazonRekognitionServiceRole` degli argomenti —ad esempio,. AmazonRekognition`AmazonRekognitionMyTopicName`

Per consentire ad Video Amazon Rekognition di accedere a più argomenti di Amazon SNS

1. Crea un ruolo di servizio IAM. Per creare il ruolo di servizio IAM, utilizza le informazioni seguenti:

    1. Scegli Rekognition come nome del servizio.
    2. Scegli Rekognition come caso d'uso del ruolo del servizio. Dovresti vedere la politica delle autorizzazioni elencata. AmazonRekognitionServiceRole AmazonRekognitionServiceRoleoffre

ad Amazon Rekognition Video l'accesso agli argomenti di Amazon SNS con il prefisso. AmazonRekognition

3. Dai al ruolo del servizio un nome a scelta.

2. Prendere nota dell'ARN del ruolo del servizio. necessario per avviare le operazioni di analisi video.

Accesso a un argomento Amazon SNS esistente

Puoi creare una policy di autorizzazione che consente a Video Amazon Rekognition l'accesso a un argomento Amazon SNS esistente.

Per consentire ad Video Amazon Rekognition di accedere a argomenti di Amazon SNS esistenti

1. Crea una nuova policy di autorizzazione con l'editor delle policy JSON IAM e utilizza la policy seguente. Sostituisci `topicarn` con il nome della risorsa Amazon (ARN) dell'argomento Amazon SNS desiderato.

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

2. Crea un ruolo di servizio IAM o aggiornane uno esistente. Per creare il ruolo di servizio IAM, utilizza le informazioni seguenti:

    1. Scegli Rekognition come nome del servizio.
    2. Scegli Rekognition come caso d'uso del ruolo del servizio.
    3. Collega la policy di autorizzazione creata nella fase 1.

3. Prendere nota dell'ARN del ruolo del servizio. necessario per avviare le operazioni di analisi video.

Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK)

Questa procedura illustra come rilevare le etichette in un video tramite le operazioni di rilevamento delle etichette Video Amazon Rekognition, un video archiviato in un bucket Amazon S3 e un argomento Amazon SNS. La procedura, inoltre, illustra come utilizzare una coda Amazon SQS per ottenere lo stato di completamento dall'argomento Amazon SNS. Per ulteriori informazioni, consulta Chiamata delle operazioni Video Amazon Rekognition. È consentito utilizzare una coda Amazon SQS. Ad esempio, puoi utilizzare una AWS Lambda funzione per ottenere lo stato di completamento. Per ulteriori informazioni, consulta Invocazione di funzioni Lambda tramite le notifiche di Amazon SNS.

Il codice di esempio nella procedura illustra come effettuare le seguenti operazioni:

1. Creare l'argomento Amazon SNS.
2. Creazione di una coda Amazon SQS.
3. Concedere ad Video Amazon Rekognition l'autorizzazione per pubblicare lo stato di completamento di un'operazione di analisi video nell'argomento Amazon SNS.
4. Sottoscrizione della coda Amazon SQS all'argomento Amazon SNS.
5. Avvio della richiesta di analisi video chiamando StartLabelDetection.
6. Ottenimento dello stato di completamento dalla coda Amazon SQS. L'esempio rileva l'identificatore del processo (`JobId`) restituito in `StartLabelDetection` e ottiene solo i risultati relativi agli identificatori del processo corrispondenti letti dallo stato di completamento. Si tratta di una considerazione importante se altre applicazioni utilizzano la stessa coda e lo stesso argomento. Per semplicità, l'esempio elimina i processi che non corrispondono. Si può valutare di aggiungerli a una coda DLQ di Amazon SQS per ulteriori controlli.
7. Ottenimento e visualizzazione dei risultati di analisi video chiamando GetLabelDetection.

Prerequisiti

Il codice di esempio per questa procedura viene fornito in Java e Python. È necessario che sia installato l' AWS SDK appropriato. Per ulteriori informazioni, consulta Nozioni di base su Amazon Rekognition. L'account AWS utilizzato deve disporre di autorizzazioni di accesso all'API Amazon Rekognition. Per ulteriori informazioni, consulta la sezione Operazioni definite da Amazon Rekognition.

Per rilevare le etichette in un video

1. Configura l'accesso utente ad Video Amazon Rekognition e configura l'accesso di Video Amazon Rekognition ad Amazon SNS. Per ulteriori informazioni, consulta Configurazione di Video Amazon Rekognition. Non è necessario eseguire le fasi 3, 4, 5 e 6 perché il codice di esempio crea e configura l'argomento Amazon SNS e la coda Amazon SQS.
2. Carica un file video in formato MOV o MPEG-4 nel bucket Amazon S3. A scopo di verifica, carica un video non più lungo di 30 secondi.

   Per le istruzioni, consulta Caricamento di oggetti in Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

3. Utilizza i seguenti esempi di codice per rilevare le etichette in un video.

   Java

   Nella funzione `main`:

   - Sostituisci `roleArn` con l'ARN del ruolo di servizio IAM creato nella fase 7 di Per configurare Video Amazon Rekognition.
   - Sostituisci i valori di `amzn-s3-demo-bucket` e `video` con il nome del file video e del bucket specificati nella fase 2.

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

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

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

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

public class VideoDetect {


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

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


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

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

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

```
        CreateTopicandQueue();

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

        StartLabelDetection(bucket, video);

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

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


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

    }


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


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

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

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

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


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

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

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

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

```
        return success;
    }


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

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


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

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

    }

    private static void GetLabelDetectionResults() throws Exception{

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

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

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

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

            VideoMetadata videoMetaData=labelDetectionResult.getVideoMetadata();

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


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

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

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

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

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

   }

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

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

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

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

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


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


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

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

Python

Nella funzione `main`:

- Sostituisci `roleArn` con l'ARN del ruolo di servizio IAM creato nella fase 7 di Per configurare Video Amazon Rekognition.
- Sostituisci i valori di `amzn-s3-demo-bucket` e `video` con il nome del file video e del bucket specificati nella fase 2.
- Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

- Puoi anche includere criteri di filtraggio nel parametro delle impostazioni. Ad esempio, è possibile utilizzare un `LabelsInclusionFilter` o a `LabelsExclusionFilter` accanto a un elenco di valori desiderati. Nel codice seguente, puoi decommentare la sezione `Features` e `Settings` e fornire i tuoi valori per limitare i risultati restituiti alle sole etichette che ti interessano.
- Nella chiamata a`GetLabelDetection`, puoi fornire valori per gli argomenti `SortBy` e `AggregateBy`. Per ordinare in base al tempo, imposta il valore del parametro di input `SortBy` su `TIMESTAMP`. Per ordinare per entità, utilizza il parametro di input `SortBy` con il valore appropriato per l'operazione in esecuzione. Per aggregare i risultati per timestamp, imposta il valore del parametro `AggregateBy` su `TIMESTAMPS`. Per aggregare per segmento video, usa `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

Nel seguente codice di esempio:

- Infine, sostituisci il valore di `REGION` con il nome della regione operativa associata al tuo account.
- Sostituisci il valore di `amzn-s3-demo-bucket` con il nome del bucket Amazon S3 che contiene il file video.
- Sostituisci il valore di `videoName` con il nome del file video nel tuo bucket Amazon S3.
- Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.
- Sostituisci `roleArn` con l'ARN del ruolo di servizio IAM creato nella fase 7 di Per configurare Video Amazon Rekognition.

```
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

Questo codice è tratto dal repository degli esempi GitHub di AWS Documentation SDK. Guarda l'esempio completo qui.

```
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. Crea ed esegui il codice. Il completamento dell'operazione potrebbe richiedere alcuni minuti. Una volta terminata, viene visualizzato un elenco delle etichette rilevate nel video. Per ulteriori informazioni, consulta Rilevamento di etichette in un video.

Analisi di un video con AWS Command Line Interface

Puoi usare il AWS Command Line Interface (AWS CLI) per chiamare le operazioni di Amazon Rekognition Video. Il modello di progettazione è lo stesso dell'utilizzo dell'API Amazon Rekognition Video con o altro AWS. AWS SDK per Java SDKs Per ulteriori informazioni, consulta Panoramica dell'API Video Amazon Rekognition. Le seguenti procedure mostrano come utilizzare il per AWS CLI rilevare le etichette in un video.

Puoi iniziare a rilevare le etichette in un video chiamando `start-label-detection`. Quando Video Amazon Rekognition termina l'analisi video, lo stato di completamento viene inviato all'argomento specificato nel parametro `--notification-channel` di `start-label-detection`. Puoi ottenere lo stato di completamento dall'argomento Amazon SNS, utilizzando una coda Amazon Simple Queue Service (Amazon SQS). Puoi quindi eseguire il polling di receive-message per ottenere lo stato di completamento dalla coda Amazon SQS.

Quando chiami`StartLabelDetection`, puoi filtrare i risultati fornendo argomenti di filtraggio agli argomenti `LabelsInclusionFilter` e/o `LabelsExclusionFilter`. Per ulteriori informazioni, consulta Rilevamento di etichette in un video.

La notifica relativa allo stato di completamento è una struttura JSON all'interno della risposta `receive-message`. È necessario estrarre la struttura JSON dalla risposta. Per informazioni sulla struttura JSON dello stato di completamento, consulta Riferimento: Notifica dei risultati dell'analisi video. Se il valore del campo `Status` della struttura JSON relativa allo stato di completamento è `SUCCEEDED`, puoi ottenere i risultati della richiesta di analisi video chiamando `get-label-detection`. Durante la chiamata`GetLabelDetection`, è possibile ordinare e aggregare i risultati restituiti utilizzando gli argomenti `SortBy` and`AggregateBy`.

Le seguenti procedure non includono il codice per eseguire il polling della coda Amazon SQS. Inoltre, non includono il codice per analizzare la struttura JSON restituita dalla coda Amazon SQS. Per un esempio in formato Java, consulta Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK).

Prerequisiti

Per eseguire questa procedura, è necessario aver AWS CLI installato il. Per ulteriori informazioni, consulta Nozioni di base su Amazon Rekognition. L'account AWS utilizzato deve disporre di autorizzazioni di accesso all'API Amazon Rekognition. Per ulteriori informazioni, Azioni definite da Video Amazon Rekognition.

Per configurare Video Amazon Rekognition e caricare un video

1. Configura l'accesso utente ad Video Amazon Rekognition e configura l'accesso di Video Amazon Rekognition ad Amazon SNS. Per ulteriori informazioni, consulta Configurazione di Video Amazon Rekognition.
2. Carica un file video in formato MOV o MPEG-4 nel bucket S3. Durante le fasi di sviluppo e test, consigliamo l'utilizzo di video brevi, non più lunghi di 30 secondi.

   Per le istruzioni, consulta Caricamento di oggetti in Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

Per rilevare le etichette in un video

1. Esegui il seguente AWS CLI comando per iniziare a rilevare le etichette in un video.

```
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"]}}
```

   Aggiorna i seguenti valori:

   - Modifica `amzn-s3-demo-bucket` e `videofile` con il nome del bucket Amazon S3 e il nome del file specificati nella fase 2.
   - Cambia `us-east-1` con la regione AWS che stai utilizzando.
   - Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.
   - Cambia `TopicARN` con l'ARN dell'argomento Amazon SNS creato nella fase 3 di Configurazione di Video Amazon Rekognition.
   - Modifica `RoleARN` con l'ARN del ruolo di servizio IAM creato nella fase 7 di Configurazione di Video Amazon Rekognition.
   - Se richiesto, puoi specificare `endpoint-url`. L'AWS CLI dovrebbe determinare automaticamente l'URL dell'endpoint corretto in base alla regione fornita. Tuttavia, se utilizzi un endpoint dal tuo VPC privato, potrebbe essere necessario specificare l'`endpoint-url`.

La risorsa AWS Service Endpoints elenca la sintassi per specificare gli URL degli endpoint e i nomi e i codici per ogni regione.

- Puoi anche includere criteri di filtraggio nel parametro delle impostazioni. Ad esempio, è possibile utilizzare un `LabelsInclusionFilter` o a `LabelsExclusionFilter` accanto a un elenco di valori desiderati.

Se accedi alla CLI da un dispositivo Windows, usa le virgolette doppie anziché le virgolette singole ed evita le virgolette doppie interne tramite barra rovesciata (ovvero, \) per risolvere eventuali errori del parser che potresti riscontrare. Un esempio è fornito di seguito:

```
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. Prendi nota del valore di `JobId` nella risposta. La risposta si presenta in maniera analoga all'esempio di struttura JSON riportato di seguito.

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

3. Scrivi il codice per eseguire il polling della coda Amazon SQS in base alla struttura JSON dello stato di completamento (utilizzando receive-message).
4. Scrivi il codice per estrarre il campo `Status` dalla struttura JSON dello stato di completamento.
5. Se il valore di `Status` è`SUCCEEDED`, esegui il AWS CLI comando seguente per mostrare i risultati del rilevamento delle etichette.

```
aws rekognition get-label-detection  --job-id JobId \
--region us-east-1 --sort-by TIMESTAMP aggregate-by TIMESTAMPS
```

Aggiorna i seguenti valori:

- Modifica `JobId` in modo che corrisponda all'identificatore del processo di cui hai preso nota nella fase 2.

- Modifica `Endpoint` e `us-east-1` nella regione e nell'endpoint AWS in uso.

I risultati si presentano in maniera analoga all'esempio di struttura JSON riportato di seguito:


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


Riferimento: Notifica dei risultati dell'analisi video

Amazon Rekognition pubblica il risultato della richiesta di analisi Video Amazon Rekognition, incluso lo stato di completamento, all’argomento Amazon Simple Notification Service (Amazon SNS). Per ricevere la notifica da un argomento di Amazon SNS, usa una coda o una funzione di Amazon

Simple Queue Service. AWS Lambda Per ulteriori informazioni, consulta the section called “Chiamata delle operazioni Video Amazon Rekognition”. Per vedere un esempio, consulta Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK).

Il payload si presenta nel formato JSON seguente:

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

Nome	Descrizione
JobId	Identificatore del processo. Corrisponde a un identificatore del processo restituito da un'operazione `Start`, ad esempio StartPersonTracking.
Stato	Stato del processo. I valori validi sono SUCCEEDED (RIUSCITO), FAILED (NON RIUSCITO) o ERROR (ERRORE).
API	L'operazione di Video Amazon Rekognition utilizzata per analizzare il video di input.
JobTag	Identificatore per il processo. Puoi specificare `JobTag` in una chiamata all'operazione Start, ad esempio StartLabelDetection.
Timestamp	Il time stamp Unix con l'ora di completamento del processo.

Nome	Descrizione
Video	Dettagli sul video elaborato. Include il nome del file e il bucket Amazon S3 in cui è archiviato.

Di seguito è riportato un esempio di una notifica corretta inviata a un argomento Amazon SNS.

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

Risoluzione dei problemi di Video Amazon Rekognition

Le indicazioni che seguono sono informazioni sulla risoluzione dei problemi nell'utilizzo di Video Amazon Rekognition e dei video archiviati.

Non ricevo mai lo stato di completamento che viene inviato all'argomento Amazon SNS

Video Amazon Rekognition pubblica le informazioni sullo stato su un argomento Amazon SNS quando l'analisi video viene completata. In genere, ricevi il messaggio relativo allo stato di completamento sottoscrivendo l'argomento con una coda Amazon SQS o una funzione Lambda. Per facilitare la tua analisi, sottoscrivi l'argomento Amazon SNS per e-mail per ricevere i messaggi inviati all'argomento Amazon SNS nella tua casella di posta elettronica. Per ulteriori informazioni, consulta Iscrizione a un argomento Amazon SNS.

Se non ricevi il messaggio nella tua applicazione, procedi nel seguente modo:

- Verifica che l'analisi sia stata completata. Verifica il `JobStatus` valore nella risposta dell'operazione Get (`GetLabelDetection`, ad esempio). Se il valore è `IN_PROGRESS`, l'analisi

non è completa e lo stato di completamento non è ancora stato pubblicato su un argomento Amazon SNS.

- Verifica che disponi di un ruolo di servizio IAM che offre a Video Amazon Rekognition le autorizzazioni per pubblicare sui tuoi argomenti Amazon SNS. Per ulteriori informazioni, consulta Configurazione di Video Amazon Rekognition.
- Verifica che il ruolo di servizio IAM che stai utilizzando possa essere pubblicato sull'argomento Amazon SNS utilizzando le credenziali del ruolo e che le autorizzazioni del tuo ruolo di servizio siano limitate in modo sicuro alle risorse che stai utilizzando. Segui questi passaggi:
  - Ottieni il nome della risorsa Amazon (ARN) dell'utente:

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

  - Aggiungere l'utente ARN a una relazione di trust. Per ulteriori informazioni, consulta Modifica un ruolo. L'esempio seguente di politica di fiducia specifica le credenziali del ruolo dell'utente e limita le autorizzazioni del ruolo di servizio alle sole risorse che stai utilizzando (per ulteriori informazioni sulla limitazione sicura dell'ambito delle autorizzazioni di un ruolo di servizio, consulta): Prevenzione del problema "confused deputy" tra servizi

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

- Assumi il ruolo: `aws sts assume-role --role-arn arn:Role ARN --role-session-name SessionName --profile RekognitionUser`
- Pubblica in un argomento Amazon SNS: `aws sns publish --topic-arn arn:Topic ARN --message "Hello World!" --region us-east-1 --profile RekognitionUser`

Se il comando AWS CLI funziona, riceverai il messaggio (nella tua casella di posta elettronica, se ti sei iscritto all'argomento tramite e-mail). Se non ricevi il messaggio:

- Controlla che Video Amazon Rekognition sia stato configurato. Per ulteriori informazioni, consulta Configurazione di Video Amazon Rekognition.
- Controlla gli altri suggerimenti per la risoluzione dei problemi relativi a questa domanda.

- Verifica che stai utilizzando l'argomento Amazon SNS corretto:
  - Se utilizzi un ruolo di servizio IAM per fornire a Video Amazon Rekognition l'accesso a un singolo argomento Amazon SNS, controlla di aver concesso le autorizzazioni necessarie all'argomento Amazon SNS corretto. Per ulteriori informazioni, consulta Accesso a un argomento Amazon SNS esistente.
  - Se utilizzi un ruolo di servizio IAM per consentire ad Amazon Rekognition Video l'accesso a più argomenti SNS, verifica di utilizzare l'argomento corretto e che il nome dell'argomento sia preceduto da. AmazonRekognition Per ulteriori informazioni, consulta Accesso a più argomenti Amazon SNS.
  - Se usi una AWS Lambda funzione, conferma che la tua funzione Lambda sia iscritta all'argomento Amazon SNS corretto. Per ulteriori informazioni, consulta Fanout alle funzioni Lambda.
- Se sottoscrivi una coda Amazon SQS relativa all'argomento Amazon SNS, verifica che il tuo argomento Amazon SNS disponga delle autorizzazioni per inviare messaggi alla coda Amazon SQS. Per ulteriori informazioni, consulta Concedere le autorizzazioni a un argomento Amazon SNS per inviare messaggi alla coda Amazon SQS.

Ho bisogno di ulteriore assistenza per la risoluzione di Amazon SNS

Puoi utilizzarlo AWS X-Ray con Amazon SNS per tracciare e analizzare i messaggi che viaggiano attraverso la tua applicazione. Per ulteriori informazioni, consulta Amazon SNS e. AWS X-Ray

Per ulteriore assistenza, puoi pubblicare la tua domanda sul forum Amazon Rekognition o prendere in considerazione la possibilità di iscriverti al AWS supporto tecnico.

Utilizzo degli eventi video in streaming

È possibile utilizzare Video Amazon Rekognition per rilevare e riconoscere i volti o gli oggetti in un video in streaming. Video Amazon Rekognition usa Flusso di video Amazon Kinesis per ricevere e processare un flusso video. Crea un elaboratore di flussi con parametri che mostrano gli elementi da rilevare dal flusso video. Rekognition invia i risultati del rilevamento delle etichette dagli eventi video in streaming come notifiche Amazon SNS e Amazon S3. Rekognition invia i risultati della ricerca del volto in un flusso di dati Kinesis.

I processori di stream Face Search utilizzano `FaceSearchSettings` per cercare volti da una raccolta. Per ulteriori informazioni su come implementare i processori di stream Face Search per analizzare i volti nei video in streaming, consultathe section called “Ricerca di volti in una raccolta in streaming video”.

Gli stream processor di rilevamento delle etichette `ConnectedHomeSettings` vengono utilizzati per cercare persone, pacchi e animali domestici negli eventi video in streaming. Per ulteriori informazioni su come implementare gli elaboratori di flussi di rilevamento di tag, consultathe section called “Rilevamento delle etichette negli eventi video in streaming”.

Panoramica delle operazioni del processore di streaming Video Amazon Rekognition

Per iniziare ad analizzare un video in streaming, avviare un elaboratore di flussi Video Amazon Rekognition e riprodurre il video in streaming in Video Amazon Rekognition. Un elaboratore di flussi Video Amazon Rekognition consente di avviare, arrestare e gestire elaboratori di flussi. Un elaboratore di flussi viene creato chiamando CreateStreamProcessor. I parametri di richiesta per la creazione di un processore di stream di ricerca facciale includono Amazon Resource Names (ARNs) per il flusso video Kinesis, il flusso di dati Kinesis e l'identificatore per la raccolta utilizzato per riconoscere i volti nel video in streaming. I parametri di richiesta per la creazione di un processore di stream di monitoraggio della sicurezza includono Amazon Resource Names (ARNs) per il flusso video Kinesis e l'argomento Amazon SNS, i tipi di oggetti che desideri rilevare nel flusso video e le informazioni per un bucket Amazon S3 per i risultati di output. Includi inoltre il nome specificato per l'elaboratore di flussi.

L'elaborazione di un video viene avviata chiamando l'operazione StartStreamProcessor. Per ottenere le informazioni sullo stato di un elaboratore di flussi, chiamare DescribeStreamProcessor. Altre operazioni che puoi eseguire sono l'etichettatura TagResourcedi uno stream processor e DeleteStreamProcessorl'eliminazione di uno stream processor. Se utilizzi uno stream processor per

la ricerca di volti, puoi anche StopStreamProcessorutilizzarlo per arrestare uno stream processor. Per ottenere un elenco di elaboratori di flussi nell'account, chiamare ListStreamProcessors.

In seguito all'avvio dell'elaboratore di flussi, eseguire in streaming il video in Amazon Rekognition per video attraverso il flusso video Kinesis specificato in `CreateStreamProcessor`. È possibile utilizzare l'operazione Kinesis Video PutMediaStreams SDK per inviare video nel flusso video Kinesis. Per un esempio, consulta Esempio di API. PutMedia

Per informazioni su come la tua applicazione può utilizzare i risultati dell'analisi di Video Amazon Rekognition da uno stream processor di ricerca facciale, consulta Lettura dei risultati delle analisi di video in streaming.

Aggiunta di tag a un nuovo elaboratore di flussi

È possibile identificare, organizzare, cercare e filtrare le raccolte Amazon Rekognition utilizzando i tag. Ogni tag è un'etichetta composta da una chiave e da un valore definiti dall'utente.

Puoi aggiungere tag a un elaboratore di flussi al momento della creazione utilizzando l'operazione `CreateStreamProcessor`. Specifica uno o più tag nel parametro di input dell'array `Tags`. Di seguito è riportato un esempio di JSON per la richiesta `CreateStreamProcessor` con tag.

```
{
       "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"
       }
}
```

Aggiunta di tag a un nuovo elaboratore di flussi

È possibile identificare, organizzare, cercare e filtrare le raccolte Amazon Rekognition utilizzando i tag. Ogni tag è un'etichetta composta da una chiave e da un valore definiti dall'utente.

Per aggiungere uno o più tag a una raccolta esistente, utilizza l'operazione `TagResource`. Specifica il nome della risorsa Amazon (ARN) dell'elaboratore di flussi (`ResourceArn`) e i tag (`Tags`) da aggiungere. L'esempio seguente mostra come aggiungere due tag.

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

> **Note**
>
> Se non conosci il nome della risorsa Amazon dell'elaboratore, puoi utilizzare l'operazione `DescribeStreamProcessor`.

Elenca tag in un elaboratore di flussi

È possibile identificare, organizzare, cercare e filtrare le raccolte Amazon Rekognition utilizzando i tag. Ogni tag è un'etichetta composta da una chiave e da un valore definiti dall'utente.

Per elencare i tag collegati a uno stream processor, utilizzate l'`ListTagsForResource`operazione e specificate l'ARN dello stream processor ()`ResourceArn`. La risposta è una mappa di chiavi e valori di tag associati a un modello specificato.

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

L'output mostra un elenco di tag associati all'elaboratore di flussi:

```
{
```

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

Eliminazione di tag da un elaboratore di flussi

È possibile identificare, organizzare, cercare e filtrare le raccolte Amazon Rekognition utilizzando i tag. Ogni tag è un'etichetta composta da una chiave e da un valore definiti dall'utente.

Per rimuovere uno o più tag da un elaboratore di flussi, utilizza l'operazione `UntagResource`. Specifica l'ARN del modello (`ResourceArn`) e le chiavi di tag (`Tag-Keys`) che desideri rimuovere.

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

In alternativa, puoi specificare le chiavi di tag in questo formato:

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

Gestione degli errori

Questa sezione descrive gli errori di runtime e come gestirli. Descrive inoltre i messaggi e codici di errore specifici di Amazon Rekognition.

Argomenti

- Componenti degli errori
- Messaggi e codici di errore
- Gestione degli errori nell'applicazione

Componenti degli errori

Quando il tuo programma invia una richiesta, Amazon Rekognition tenta di elaborarla. Se la richiesta ha esito positivo, Amazon Rekognition restituisce un codice di stato HTTP di operazione riuscita (`200` 0K), insieme ai risultati dell'operazione richiesta.

Se la richiesta ha esito negativo, Amazon Rekognition restituisce un errore. Ogni errore comprende tre componenti:

- Un codice di stato HTTP (ad esempio, `400`).
- Un nome di eccezione (ad esempio, `InvalidS3ObjectException`).
- Un messaggio di errore (ad esempio `Unable to get object metadata from S3. Check object key, region and/or access permissions.`).

AWS SDKs si occupa della propagazione degli errori nella tua applicazione, in modo che tu possa intraprendere le azioni appropriate. Ad esempio, in un programma Java puoi scrivere logica `try-catch` per gestire un errore `ResourceNotFoundException`.

Se non stai utilizzando un AWS SDK, devi analizzare il contenuto della risposta di basso livello di Amazon Rekognition. Di seguito è riportato un esempio di tale risposta:

```
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."}
```

Messaggi e codici di errore

Il seguente è un elenco di eccezioni restituite da Amazon Rekognition, raggruppate in base al codice di stato HTTP. Se OK to retry? (OK riprovare?) è Yes (Sì), puoi inviare nuovamente la stessa richiesta. Se OK to retry? (OK riprovare?) è No, è necessario risolvere il problema sul lato client prima di inviare una nuova richiesta.

Codice di stato HTTP 400

Un codice di stato HTTP `400` indica un problema con la richiesta. Alcuni esempi di problemi sono errori di autenticazione, parametri obbligatori mancanti o il superamento del throughput assegnato a un'operazione. È necessario risolvere il problema nella tua applicazione prima di inviare nuovamente la richiesta.

AccessDeniedException

Messaggio: si è verificato un errore (AccessDeniedException) durante la chiamata dell'operazione: User: is not authorized to perform: on resource:. <Operation><User ARN><Operation><Resource ARN>

Non sei autorizzato a eseguire l'operazione. Utilizzare il nome della risorsa Amazon (ARN) di un utente autorizzato o un ruolo IAM per eseguire l'operazione.

OK riprovare? No

GroupFacesInProgressException

Messaggio: impossibile pianificare GroupFaces il lavoro. È presente un processo di volti di gruppo per questa raccolta.

Riprova l'operazione al termine del processo esistente.

OK riprovare? No

IdempotentParameterMismatchException

Messaggio: Il ClientRequestToken: che hai fornito è già in uso. <Token>

Un parametro ClientRequestToken di input è stato riutilizzato con un'operazione, ma almeno uno degli altri parametri di input è diverso dalla precedente chiamata all'operazione.

OK riprovare? No

ImageTooLargeException

Messaggio: L'immagine è troppo grande.

La dimensione dell'immagine di input supera il limite consentito. Se state chiamando DetectProtectiveEquipment, la dimensione o la risoluzione dell'immagine superano il limite consentito. Per ulteriori informazioni, consulta Linee guida e quote in Amazon Rekognition.

OK riprovare? No

InvalidImageFormatException

Messaggio: La richiesta ha un formato di immagine non valido.

Il formato di immagine fornito non è supportato. Utilizza un formato di immagine supportato (.JPEG e.PNG). Per ulteriori informazioni, consulta Linee guida e quote in Amazon Rekognition.

OK riprovare? No

InvalidPaginationTokenException

Messaggi

- Token non valido
- Token di paginazione non valido

Il token di paginazione della richiesta non è valido. Il token potrebbe essere scaduto.

OK riprovare? No

InvalidParameterException

Messaggio: La richiesta ha parametri non validi.

Un parametro di input ha violato un vincolo. Convalida i parametri prima di chiamare nuovamente l'operazione API.

OK riprovare? No

Non valido (3) ObjectException

Messaggi:

- La richiesta ha un oggetto S3 non valido.
- Impossibile ottenere i metadati dell'oggetto da S3. Controlla la chiave dell'oggetto, la regione e/o le autorizzazioni di accesso.

Amazon Rekognition non è in grado di accedere all'oggetto S3 specificato nella richiesta. Per ulteriori informazioni, consulta Gestione delle autorizzazioni di accesso alle risorse di Amazon S3. Per informazioni sulla risoluzione dei problemi, consulta Risoluzione dei problemi di Amazon S3.

OK riprovare? No

LimitExceededException

Messaggi:

- Limite del processore di flusso superato per l'account, limite - <Limite corrente>.
- <Numero di processi aperti> processi aperti per l'utente <ARN dell'utente> Limite massimo: <Limite massimo>

È stato superato un limite per il servizio Amazon Rekognition. Se avvii troppi processi Video Amazon Rekognition simultaneamente, le chiamate alle operazioni di avvio, come `StartLabelDetection`, generano un'eccezione `LimitExceededException` (codice di stato HTTP: 400) finché il numero di processi simultanei in esecuzione è inferiore al limite del servizio Amazon Rekognition.

OK riprovare? No

ProvisionedThroughputExceededException

Messaggi:

- Velocità assegnata superata.
- Limite di download S3 superato.

Il numero di richieste ha superato il limite di throughput. Per ulteriori informazioni, consulta l'argomento Limiti del servizio Amazon Rekognition.

Per richiedere un aumento del limite, segui le istruzioni riportate in the section called “Crea un caso per modificare le quote TPS”.

OK riprovare? Sì

ResourceAlreadyExistsException

Messaggio: L'ID raccolta: <ID raccolta> esiste già.

Una collezione con l'ID specificato esiste già.

OK riprovare? No

ResourceInUseException

Messaggi:

- Nome del processore di flussi già in uso.
- La risorsa specificata è in uso.
- Il processore non è disponibile per l'arresto del flusso.
- Impossibile eliminare il processore di flussi.

Riprova quando la risorsa è disponibile.

OK riprovare? No

ResourceNotFoundException

Messaggio: Vari messaggi a seconda della chiamata API.

La risorsa specificata non esiste.

OK riprovare? No

ThrottlingException

Messaggio: Rallenta per improvviso aumento del numero di richieste.

La velocità di aumento delle richieste è troppo elevata. Rallenta e aumenta gradualmente la velocità delle richieste. Consigliamo di eseguire il backoff esponenziale e ripetizione dei tentativi. Per impostazione predefinita, AWS SDKs utilizzano la logica di ripetizione automatica e il backoff esponenziale. Per ulteriori informazioni, consulta Ripetizione dei tentativi in caso di errore e backoff esponenziale in AWS e l'argomento relativo al backoff esponenziale e jitter.

OK riprovare? Sì

VideoTooLargeException

Messaggio: Le dimensioni del video in byte <Dimensione Video> superano il limite massimo di <Dimensioni massime> byte.

La dimensione del file o la durata del supporto fornito è troppo grande. Per ulteriori informazioni, consulta Linee guida e quote in Amazon Rekognition.

OK riprovare? No

Codice di stato HTTP 5xx

Un codice di stato HTTP `5xx` indica un problema che deve essere risolto da AWS. Potrebbe trattarsi di un errore temporaneo. In questo caso, è possibile riprovare la richiesta finché non va a buon fine. In caso contrario, accedi al Pannello di controllo per lo stato dei servizi AWS per verificare se ci sono problemi operativi relativi al servizio.

InternalServerError (HTTP 500)

> Messaggio: Errore interno del server
>
> Amazon Rekognition ha riscontrato un problema del servizio. Riprova la chiamata. Dovresti eseguire il backoff esponenziale e riprovare. Per impostazione predefinita, AWS SDKs utilizzano la logica di ripetizione automatica e il backoff esponenziale. Per ulteriori informazioni, consulta Ripetizione dei tentativi in caso di errore e backoff esponenziale in AWS e l'argomento relativo al backoff esponenziale e jitter.
>
> OK riprovare? Sì

ThrottlingException (HTTP 500)

> Messaggio: Servizio non disponibile
>
> Amazon Rekognition non è temporaneamente in grado di elaborare la richiesta. Riprova la chiamata. Consigliamo di eseguire il backoff esponenziale e ripetizione dei tentativi. Per impostazione predefinita, AWS SDKs utilizzano la logica di ripetizione automatica e il backoff esponenziale. Per ulteriori informazioni, consulta Ripetizione dei tentativi in caso di errore e backoff esponenziale in AWS e l'argomento relativo al backoff esponenziale e jitter.
>
> OK riprovare? Sì

Gestione degli errori nell'applicazione

Per il buon funzionamento dell'applicazione devi aggiungere logica che intercetti gli errori e risponda adeguatamente. Gli approcci tipici includono l'utilizzo di blocchi `try-catch` o di istruzioni `if-then`.

L'AWS SDKs esegue i propri tentativi e verifica degli errori. Se riscontri un errore durante l'utilizzo di uno degli AWS SDKs, il codice di errore e la descrizione possono aiutarti a risolverlo.

Dovresti anche vedere un `Request ID` nella risposta. Il `Request ID` può essere utile se devi collaborare con AWS Support per diagnosticare un problema.

Il seguente frammento di codice Java tenta di rilevare oggetti in un'immagine ed esegue una gestione rudimentale degli errori. In questo caso, informa l'utente che la richiesta non è riuscita.

```
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 questo frammento di codice, il costrutto `try-catch` gestisce due tipi diversi di eccezioni:

- `AmazonRekognitionException`: questa eccezione si verifica se la richiesta client è stata trasmessa correttamente ad Amazon Rekognition, ma Amazon Rekognition non è in grado di elaborarla e restituisce invece una risposta di errore.
- `AmazonClientException`: questa eccezione si verifica se il client non è stato in grado di ottenere una risposta da un servizio, oppure se il client non è stato in grado di analizzare la risposta di un servizio.

Utilizzo di Amazon Rekognition come servizio autorizzato FedRamp

Il programma di AWS conformità FedRAMP include Amazon Rekognition come servizio autorizzato da FedRAMP. Se sei un cliente federale o commerciale, puoi utilizzare il servizio per elaborare e archiviare carichi di lavoro sensibili nelle regioni degli AWS Stati Uniti orientali e occidentali degli Stati Uniti occidentali, con dati fino al livello di impatto moderato. Puoi utilizzare il servizio per carichi di lavoro sensibili nel limite di autorizzazione della regione AWS GovCloud (Stati Uniti), con dati fino al livello di impatto elevato. Per ulteriori informazioni sulla conformità FedRamp, consulta la sezione relativa alla Conformità al programma Fedramp AWS.

Per essere conforme al programma Fedramp, è possibile utilizzare un endpoint FIPS (Federal Information Processing Standard). Ciò consente di accedere ai moduli crittografici convalidati FIPS 140-2 quando si lavora con informazioni sensibili. Per ulteriori informazioni sugli endpoint FIPS, consulta Panoramica su FIPS 140-2.

Puoi utilizzare AWS Command Line Interface (AWS CLI) o uno degli AWS SDKs per specificare l'endpoint utilizzato da Amazon Rekognition.

Per gli endpoint che è possibile utilizzare con Amazon Rekognition, consulta Amazon Rekognition Regions ed Endpoints.

Di seguito sono riportati esempi tratti dall'argomento Elenchi delle raccolte nella Guida per gli sviluppatori di Amazon Rekognition. Vengono modificati per specificare l'endpoint Regione e FIPS attraverso il quale si accede a Amazon Rekognition.

Java

Per Java, utilizzare il metodo `withEndpointConfiguration` quando si costruisce il client Amazon Rekognition. In questo esempio vengono illustrate le raccolte che utilizzano l'endpoint FIPS nella regione Stati Uniti orientali (Virginia Settentrionale):

```
//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

Per la AWS CLI, usa l'`--endpoint-url`argomento per specificare l'endpoint attraverso il quale si accede ad Amazon Rekognition. In questo esempio vengono illustrate le raccolte che utilizzano l'endpoint FIPS nell'area Stati Uniti orientali (Ohio):

```
aws rekognition list-collections --endpoint-url https://rekognition-fips.us-
east-2.amazonaws.com --region us-east-2
```

Python

Per Python, utilizzare l'argomento `endpoint_url` nella funzione boto3.client. Impostarlo sull'endpoint che si desidera specificare. In questo esempio vengono illustrate le raccolte che utilizzano l'endpoint FIPS nell'area Stati Uniti occidentali (Oregon):

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

import boto3

def list_collections():

    max_results=2

    client=boto3.client('rekognition', endpoint_url='https://rekognition-fips.us-
west-2.amazonaws.com', region_name='us-west-2')

    #Display all the collections
    print('Displaying collections...')
    response=client.list_collections(MaxResults=max_results)
    collection_count=0
    done=False

    while done==False:
        collections=response['CollectionIds']

        for collection in collections:
            print (collection)
            collection_count+=1
        if 'NextToken' in response:
            nextToken=response['NextToken']
```

```
 response=client.list_collections(NextToken=nextToken,MaxResults=max_results)

        else:
            done=True

    return collection_count

def main():

    collection_count=list_collections()
    print("collections: " + str(collection_count))
if __name__ == "__main__":
    main()
```

Best practice per sensori, immagini di input e video

Questa sezione contiene informazioni di best practice per l'utilizzo di Amazon Rekognition. Queste best practice ti aiuteranno a ottenere prestazioni ottimali dalle operazioni che richiami. Se avete difficoltà a ottenere i risultati attesi da un'operazione, assicuratevi di seguire le migliori pratiche documentate qui.

Per informazioni sulla latenza delle operazioni sulle immagini, consultate quanto segue:

- the section called “Latenza delle operazioni di Immagini Amazon Rekognition”

Le operazioni di confronto facciale e di ricerca dei volti richiedono il rispetto di best practice specifiche per trovare i volti in un'immagine. Questi requisiti sono stati documentati anche nel seguente link:

- the section called “Raccomandazioni per immagini di input di confronto di volti”
- the section called “Raccomandazioni per la ricerca dei volti in una raccolta”

Le seguenti sezioni spiegano come configurare la videocamera per ogni tipo di supporto che Amazon Rekognition è in grado di analizzare:

- the section called “Raccomandazioni per le impostazioni della fotocamera (immagini e video)”
- the section called “Raccomandazioni per le impostazioni della fotocamera (video archiviati e in streaming)”
- the section called “Raccomandazioni per le impostazioni della fotocamera (video in streaming)”

Le operazioni di Face Liveness hanno anche le proprie best practice da seguire per ottenere le migliori prestazioni dallo strumento di controllo Liveness:

- the section called “Raccomandazioni per l'uso di Face Liveness”

Latenza delle operazioni di Immagini Amazon Rekognition

Per garantire la latenza più bassa possibile per le operazioni di Immagini Amazon Rekognition, tenere in considerazione quanto segue:

- La regione del bucket Amazon S3 contenente le immagini deve corrispondere alla regione utilizzata per le operazioni API di Immagini Amazon Rekognition.
- Chiamare un'operazione Immagini Amazon Rekognition con i byte di immagine è più veloce che caricare l'immagine in un bucket Amazon S3 e fare poi riferimento all'immagine caricata in un'operazione Immagini Amazon Rekognition. Considera questo approccio se stai caricando immagini su Immagini Amazon Rekognition per l'elaborazione quasi in tempo reale. Ad esempio, le immagini caricate da una telecamera IP o tramite un portale Web.
- Se l'immagine è già in un bucket Amazon S3, farvi riferimento in un'operazione Immagini Amazon Rekognition sarà probabilmente più veloce del trasferimento di byte di immagine all'operazione.

Raccomandazioni per immagini di input di confronto di volti

I modelli utilizzati per le operazioni di confronto facciale sono progettati per funzionare con un'ampia varietà di pose, espressioni, età, rotazioni, condizioni di illuminazione e dimensioni. Per la scelta delle foto per CompareFaces o per aggiungere facce a una raccolta utilizzando IndexFaces, ti consigliamo di adottare le seguenti linee guida di riferimento.

Raccomandazioni generali per le immagini di input per le operazioni relative ai volti

- Utilizzare immagini luminose e nitide. Se possibile, evita immagini che potrebbero risultare sfocate per il movimento del soggetto o della fotocamera. DetectFaces può essere utilizzato per determinare la luminosità e nitidezza di un volto.
- Ai fini del rilevamento dello sguardo, si consiglia di caricare l'immagine originale con le dimensioni e la qualità originali.
- Utilizza un'immagine in cui il volto rientri nell'intervallo di angolazioni consigliato. Il beccheggio (inclinazione verticale) non deve essere superiore a 30 gradi verso il basso e 45 gradi verso l'alto. L'imbardata (rotazione laterale) non deve superare i 45 gradi verso destra o sinistra. Non vi sono limiti per il rollio (inclinazione laterale).
- Utilizza un'immagine in cui entrambi gli occhi siano bene aperti e visibili.
- L'immagine del volto non deve essere oscurata o molto ritagliata. L'immagine deve contenere la testa intera e le spalle della persona. Non deve essere ritagliata sul riquadro di delimitazione del volto.
- Evita gli elementi che nascondano il viso, come ad esempio fasce e maschere.

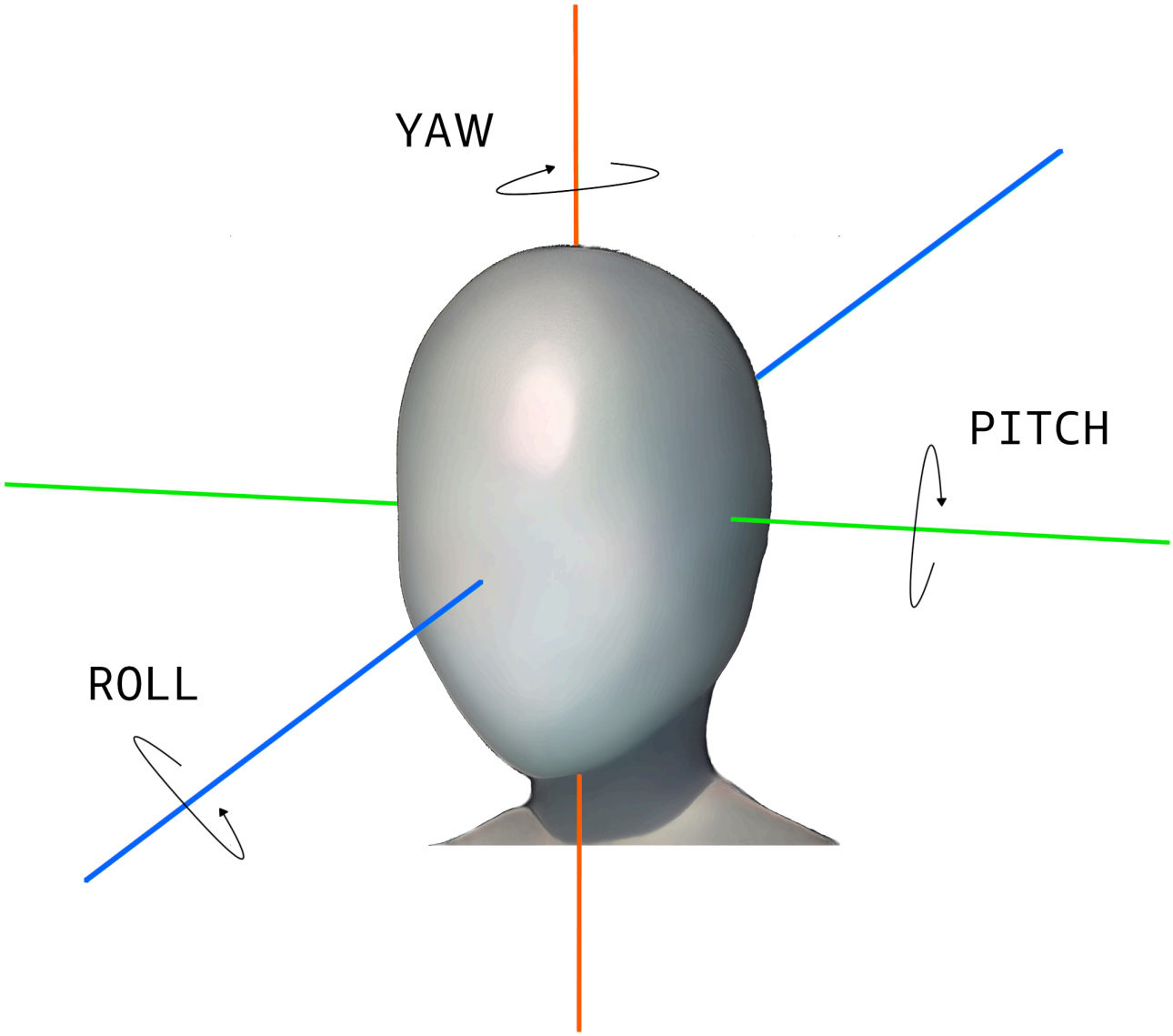
- Il volto deve occupare un'ampia porzione dell'immagine. Il confronto risulta più preciso se il volto occupa la maggior parte dell'immagine.
- Assicurati che le immagini siano sufficientemente grandi in termini di risoluzione. Amazon Rekognition è in grado di riconoscere volti di dimensioni fino a 50 x 50 pixel con risoluzioni di immagine fino a 1920 x 1080. Immagini con una risoluzione più alta richiedono una dimensione minima del volto maggiore. Se le dimensioni del viso sono superiori al minimo richiesto i risultati del riconoscimento saranno più accurati.
- Le immagini devono essere a colori.
- Il volto deve avere un'illuminazione uniforme, senza grandi variazioni di luci e ombre.
- Utilizzare immagini che abbiano un contrasto sufficiente con lo sfondo. Uno sfondo monocromatico ad alto contrasto è l'ideale.
- Per le applicazioni che richiedono la massima precisione, utilizza immagini di volti con espressioni neutre: bocca chiusa e poco sorriso.

Raccomandazioni per la ricerca dei volti in una raccolta

- Quando cerchi volti in una raccolta, assicurati che le immagini dei volti recenti siano indicizzate.
- Durante la creazione di una raccolta con `IndexFaces`, utilizza più immagini di una stessa persona con diverse inclinazioni e rotazioni (all'interno dell'intervallo di angolazioni consigliato). Consigliamo di indicizzare almeno cinque immagini di una persona: viso rivolto direttamente alla fotocamera, girato a sinistra di 45 gradi o meno, girato a destra di 45 gradi o meno, abbassato di 30 gradi o meno e sollevato di 45 gradi o meno. Per avere la certezza che le varie istanze di un volto appartengano alla stessa persona, prova a utilizzare l'attributo ID immagine esterna, se nell'immagine è indicizzato un unico viso. Ad esempio, è possibile tenere traccia di cinque immagini di John Doe nella raccolta con un'immagine esterna as. IDs `John_Doe_1.jpg, … John_Doe_5.jpg`

Raccomandazioni per le impostazioni della fotocamera (immagini e video)

I suggerimenti forniti di seguito sono un'aggiunta a quanto riportato in Raccomandazioni per immagini di input di confronto di volti.



- Risoluzione immagine: non è previsto alcun requisito minimo per la risoluzione dell'immagine, purché la risoluzione del volto sia 50x50 pixel nel caso di immagini con una risoluzione complessiva fino a 1920x1080. Immagini con una risoluzione più alta richiedono una dimensione minima del volto maggiore.

> **Note**
>
> Il suggerimento precedente fa riferimento alla risoluzione nativa della fotocamera. La creazione di un'immagine ad alta risoluzione partendo da una a bassa risoluzione non produce risultati utili per la ricerca facciale (a causa degli artefatti generati dal campionamento dell'immagine).

- Angolazione della fotocamera: esistono tre misurazioni per l'angolazione: beccheggio, rollio e imbardata.
  - Beccheggio: è consigliabile un beccheggio inferiore a 30 gradi se la fotocamera è rivolta verso il basso e inferiore a 45 gradi se è invece rivolta verso l'alto.
  - Rollio: non esiste un requisito minimo per questo parametro. Amazon Rekognition è in grado di gestire qualsiasi quantità di rollio.
  - Imbardata: è consigliabile un’imbardata inferiore a 45 gradi in entrambe le direzioni.

  L'angolazione del volto su qualsiasi asse acquisito dalla fotocamera corrisponde alla somma dell'angolazione della fotocamera rispetto alla scena e della testa del soggetto all'interno della scena. Ad esempio, se la fotocamera è inclinata di 30 gradi verso il basso e la persona ha a sua volta la testa abbassata di 30 gradi, il beccheggio effettivo risultante nell'immagine sarà di 60 gradi. In questo caso, Amazon Rekognition non sarebbe in grado di riconoscere il volto. Ti consigliamo di impostare l'angolo delle fotocamere partendo dal presupposto che di solito le persone tendono a guardare direttamente l'obiettivo, con un beccheggio complessivo (somma di volto e fotocamera) non superiore a 30 gradi.
- Zoom della fotocamera: questa impostazione della fotocamera deve partire dal presupposto che la risoluzione minima per il riconoscimento di un volto è di 50x50 pixel. È consigliabile usare lo zoom della fotocamera, in modo che i volti da acquisire abbiano una risoluzione di almeno 50X50 pixel.
- Altezza della fotocamera: scegli questa impostazione tenendo a mente i limiti consigliati per il beccheggio.

Raccomandazioni per le impostazioni della fotocamera (video archiviati e in streaming)

I suggerimenti forniti di seguito sono un'aggiunta a quanto riportato in Raccomandazioni per le impostazioni della fotocamera (immagini e video).

- Il codec deve avere la codifica h.264.
- La frequenza di frame consigliata è 30 fps (non deve essere inferiore a 5 fps).
- Il bitrate consigliato per l'encoder è di 3 Mbps (non deve essere inferiore a 1,5 Mbps).
- Frequenza di frame e risoluzione dei frame: se il bitrate dell'encoder fosse un vincolo, è consigliabile scegliere una risoluzione di frame elevata piuttosto che una maggiore frequenza di frame. Tale impostazione migliora i risultati delle ricerche di volti. In questo modo, Amazon Rekognition ottiene il frame della migliore qualità all'interno del bitrate allocato. Tuttavia, questa situazione presenta un aspetto negativo. A causa della bassa frequenza di frame, la telecamera perde il movimento rapido in una scena. È importante comprendere i pro e i contro di questi due parametri per scegliere il compromesso più adatto alle varie esigenze. Ad esempio, se il massimo bitrate possibile è di 1,5 Mbps, una fotocamera potrà acquisire 1080p a 5 fps oppure 720p a 15 fps. La scelta tra i due dipende dall'applicazione, purché venga mantenuta la risoluzione raccomandata di 50x50 pixel per i volti.

Raccomandazioni per le impostazioni della fotocamera (video in streaming)

Il suggerimento fornito di seguito rappresenta un'aggiunta a quanto riportato in Raccomandazioni per le impostazioni della fotocamera (video archiviati e in streaming).

Un ulteriore vincolo delle applicazioni in streaming è costituito dalla larghezza di banda di Internet. Nel caso di video live, Amazon Rekognition accetta solo Flusso di video Amazon Kinesis come input. È consigliabile avere una buona comprensione della dipendenza tra il bitrate dell'encoder e la larghezza di banda disponibile nella rete. La larghezza di banda disponibile deve supportare almeno lo stesso bitrate usato dalla fotocamera per codificare lo streaming in tempo reale. In questo modo, tutto ciò che viene acquisito dalla fotocamera viene inoltrato tramite Flusso di video Amazon Kinesis. Se la larghezza di banda disponibile è inferiore al bitrate dell'encoder, Flusso di video Amazon Kinesis riduce i bit in base alla larghezza di banda di rete. Ciò si traduce in una bassa qualità video.

Le impostazioni di streaming più comuni prevedono la connessione di più fotocamere a un hub di rete che inoltra i flussi. In questo caso, la larghezza di banda deve contenere la somma cumulativa dei flussi provenienti da tutte le fotocamere connesse all'hub. Ad esempio, se l'hub è connesso a cinque fotocamere con codifica a 1,5 Mbps, la larghezza di banda di rete disponibile deve essere di almeno 7,5 Mbps. Per garantire che nessun pacchetto venga eliminato, occorre valutare l'opportunità

di mantenere una larghezza di banda di rete superiore a 7,5 Mbps per compensare le perdite di connessione tra una fotocamera e l'hub. Il valore effettivo dipende dall'affidabilità della rete interna.

Raccomandazioni per l'uso di Face Liveness

Consigliamo di seguire le best practice per l'utilizzo di Rekognition Face Liveness:

- Gli utenti devono completare il controllo Face Liveness in ambienti non troppo scuri o troppo luminosi e con un'illuminazione abbastanza uniforme.
- Gli utenti devono aumentare la luminosità dello schermo fino al livello massimo quando effettuano controlli sui browser Web. Mobile Native SDKs regola automaticamente la luminosità dello schermo.
- Scegli una soglia di punteggio di affidabilità che rifletta la natura del tuo caso d'uso. Per i casi d'uso con maggiori preoccupazioni in materia di sicurezza, utilizza una soglia alta.
- Esegui regolarmente controlli di verifica umani sulle immagini di controllo per assicurarti che gli attacchi di tipo spoof vengano mitigati alla soglia di confidenza impostata.
- Offri un percorso alternativo di controllo del riconoscimento facciale ai tuoi utenti se sono fotosensibili o non vogliono controllare il riconoscimento facciale utilizzando Rekognition.
- Non inviare o visualizzare il punteggio di verifica del riconoscimento sull'applicazione utente. Invia solo un segnale positivo o negativo.
- Consenti solo cinque controlli di riconoscimento non riusciti in tre minuti da un singolo dispositivo. Dopo cinque tentativi non riusciti, rendi inattivo l'utente per 30-60 minuti. Se lo schema viene rilevato ripetutamente da 3 a 5 volte, impedisci al dispositivo utente di effettuare chiamate aggiuntive.
- Implementa la schermata di preparazione nel tuo flusso di lavoro in modo che gli utenti possano superare più facilmente i controlli Face Liveness.
- L'utente è responsabile della fornitura di informative sulla privacy legalmente adeguate agli utenti finali e dell'ottenimento del consenso necessario da parte degli stessi per l'elaborazione, l'archiviazione, l'uso e il trasferimento dei contenuti da parte di Face Liveness.

Rilevamento di oggetti e concetti

Questa sezione fornisce informazioni per rilevare le etichette nelle immagini e nei video con Immagini Amazon Rekognition e Video Amazon Rekognition.

Un'etichetta o tag è un oggetto o un concetto (incluse scene o azioni) rilevato in un'immagine o un video in base ai relativi contenuti. Ad esempio, un'immagine di persone su una spiaggia tropicale può contenere etichette come Palma (oggetto), Spiaggia (scena), Corsa (azione) e Ambiente esterno (concept).

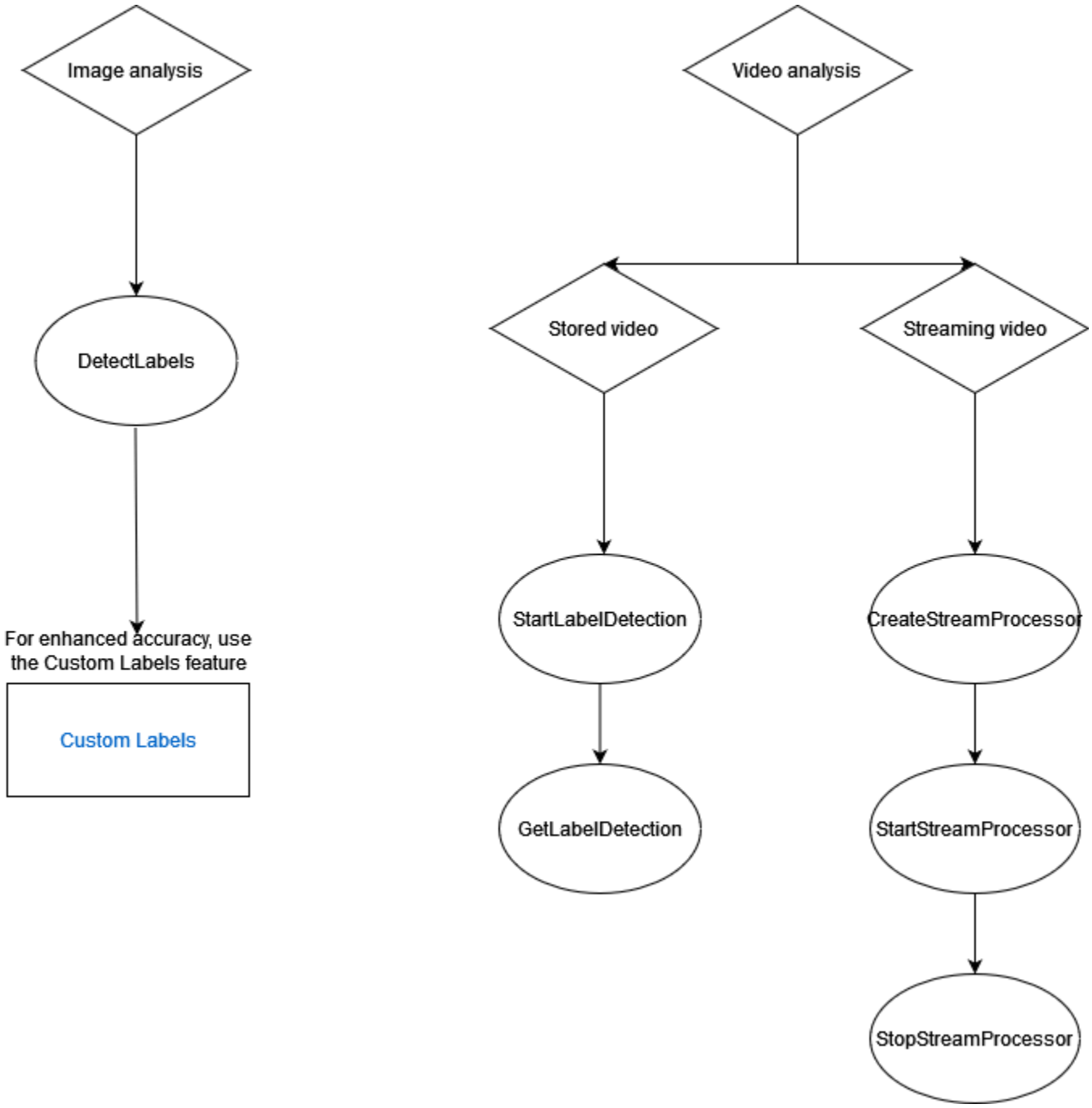
Etichette supportate dalle operazioni di rilevamento delle etichette di Rekognition

- Per scaricare l'elenco più recente di etichette e riquadri di delimitazione degli oggetti supportati da Amazon Rekognition, fai clic qui.
- Per scaricare l'elenco precedente di etichette e riquadri di delimitazione degli oggetti, fai clic qui.

> **Note**
>
> Amazon Rekognition fa previsioni binarie di genere (uomo, donna, ragazza, ecc.) basate sull'aspetto fisico di una persona in una particolare immagine. Questo tipo di previsione non è progettata per categorizzare l'identità di genere di una persona e non è necessario utilizzare Amazon Rekognition per determinare tale indicazione. Ad esempio, un attore maschio che indossa una parrucca con i capelli lunghi e degli orecchini per un ruolo potrebbe essere considerato una donna.
> L'uso di Amazon Rekognition per fare previsioni binarie di genere è più adatto per i casi d'uso in cui è necessario analizzare le statistiche aggregate sulla distribuzione di genere senza identificare utenti specifici. Ad esempio, la percentuale di utenti donne rispetto agli uomini su una piattaforma di social media.
> Non è consigliabile utilizzare le previsioni binarie di genere per prendere decisioni che influiscono sui diritti, sulla privacy o sull'accesso ai servizi di un individuo.

Amazon Rekognition restituisce le etichette in inglese. Puoi usare Amazon Translate per tradurre etichette inglesi in altre lingue.

Il diagramma seguente mostra l'ordine delle operazioni di chiamata, a seconda degli obiettivi per l'utilizzo delle operazioni Amazon Rekognition Image o Amazon Rekognition Video:



Etichetta oggetti di risposta

Riquadri di delimitazione

Immagini Amazon Rekognition e Video Amazon Rekognition è in grado di restituire il riquadro di delimitazione per vari oggetti comuni, come ad esempio automobili, mobili, abbigliamento o animali. Le informazioni relative ai riquadri non vengono restituite per le etichette di oggetti meno comuni.

Puoi utilizzare i riquadri di delimitazione per trovare l'esatta ubicazione degli oggetti all'interno di un'immagine, contare le istanze di oggetti rilevati o misurare le dimensioni di un oggetto tramite quelle del riquadro.

Ad esempio, nell'immagine seguente, Immagini Amazon Rekognition è in grado di rilevare la presenza di una persona, uno skateboard, auto parcheggiate e altre informazioni. Immagini Amazon Rekognition restituisce anche il riquadro di delimitazione per una persona rilevata e altri oggetti rilevati come auto e ruote.

Punteggio di attendibilità

Video Amazon Rekognition e Immagini Amazon Rekognition forniscono un punteggio percentuale che indica la fiducia di Amazon Rekognition nella precisione di ogni etichetta rilevata.

Genitori

Immagini Amazon Rekognition e Video Amazon Rekognition utilizzano una tassonomia gerarchica delle etichette dei predecessori per classificare le etichette. Ad esempio, una persona che cammina

su una strada potrebbe essere rilevata come Pedone. L'etichetta padre di Pedone è Persona. Nella risposta vengono restituite entrambe le etichette. Tutti predecessori vengono restituiti e una determinata etichetta contiene un elenco con l'etichetta padre e altri predecessori. Ad esempio, potrebbero essere indicate l'etichetta nonno e bisnonno (se esistenti). Le etichette padre possono essere usate per creare gruppi di etichette correlate e consentire l'esecuzione di query di etichette simili in una o più immagini. Ad esempio, una query per tutti i Veicoli potrebbe restituire un'auto da un'immagine e una moto da un'altra.

Categories

Immagini Amazon Rekognition e Video Amazon Rekognition restituiscono informazioni sulle categorie di etichette. Le etichette fanno parte di categorie che raggruppano singole etichette in base a funzioni e contesti comuni, come «Veicoli e automobili» e «Prodotti alimentari e bevande». Una categoria di etichette può essere una sottocategoria di una categoria principale.

Alias

Oltre a restituire le etichette, Immagini Amazon Rekognition e Video Amazon Rekognition restituiscono tutti gli alias associati all'etichetta. Gli alias sono etichette con lo stesso significato o etichette visivamente intercambiabili con l'etichetta principale restituita. Ad esempio, «Telefono cellulare» è un alias di «Telefono mobile».

Nelle versioni precedenti, Immagini Amazon Rekognition restituiva alias come «Telefono cellulare» nello stesso elenco di nomi di etichette primarie che contenevano «Telefono mobile». Immagini Amazon Rekognition ora restituisce «Telefono cellulare» in un campo chiamato «alias» e «Telefono mobile» nell'elenco dei nomi delle etichette principali. Se l'applicazione si basa sulle strutture restituite da una versione precedente di Rekognition, potrebbe essere necessario trasformare la risposta corrente restituita dalle operazioni di rilevamento delle etichette di immagini o video nella struttura di risposta precedente, in cui tutte le etichette e gli alias vengono restituiti come etichette primarie.

Se devi trasformare la risposta corrente dell' DetectLabels API (per il rilevamento delle etichette nelle immagini) nella struttura di risposta precedente, consulta l'esempio di codice in. Trasformare la risposta DetectLabels

Se devi trasformare la risposta corrente dell' GetLabelDetection API (per il rilevamento delle etichette nei video archiviati) nella struttura di risposta precedente, consulta l'esempio di codice inTrasformazione della risposta GetLabelDetection .

Proprietà immagine

Immagini Amazon Rekognition restituisce informazioni sulla qualità dell'immagine (nitidezza, luminosità e contrasto) per l'intera immagine. La nitidezza e la luminosità vengono restituite anche per il primo piano e lo sfondo dell'immagine. Le proprietà dell'immagine possono essere usate anche per rilevare i colori dominanti nell'intera immagine, in primo piano, sullo sfondo e negli oggetti con riquadri di delimitazione.

Di seguito è riportato un esempio dei ImageProperties dati contenuti nella risposta di un' DetectLabels operazione per l'immagine successiva:

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

Proprietà immagine non è disponibile per Video Amazon Rekognition.

Versione del modello

Immagini Amazon Rekognition e Video Amazon Rekognition restituiscono la versione del modello utilizzato per rilevare le etichette in un'immagine o un video archiviato.

Filtri di inclusione ed esclusione

Puoi filtrare i risultati restituiti dalle operazioni di rilevamento delle etichette di Immagini Amazon Rekognition e Video Amazon Rekognition. Filtra i risultati fornendo criteri di filtrazione per etichette e categorie. I filtri per etichette possono essere inclusivi o esclusivi.

Consulta Rilevamento di etichette in un'immagine per ulteriori informazioni sulla filtrazione dei risultati ottenuti con `DetectLabels`.

Consulta Rilevamento di etichette in un video per ulteriori informazioni sulla filtrazione dei risultati ottenuti da `GetLabelDetection`.

Ordinamento e aggregazione di risultati

I risultati ottenuti da determinate operazioni di Video Amazon Rekognition possono essere ordinati e aggregati in base a timestamp e segmenti video. Quando recuperi i risultati di un processo di rilevamento delle etichette o di moderazione dei contenuti, con `GetLabelDetection` o `GetContentModeration` rispettivamente, puoi utilizzare gli argomenti `SortBy` e `AggregateBy` per specificare come desideri che vengano restituiti i risultati. È possibile utilizzare `SortBy` con `TIMESTAMP` o `NAME` (nomi delle etichette) e utilizzare `TIMESTAMPS` o `SEGMENTS` con l'argomento. AggregateBy

Rilevamento di etichette in un'immagine

È possibile utilizzare l'DetectLabelsoperazione per rilevare etichette (oggetti e concetti) in un'immagine e recuperare informazioni sulle proprietà di un'immagine. Le proprietà dell'immagine includono attributi come il colore del primo piano e dello sfondo e la nitidezza, la luminosità e il contrasto dell'immagine. È possibile recuperare solo le etichette di un'immagine, solo le proprietà dell'immagine o entrambe. Per vedere un esempio, consulta Analisi di immagini archiviate in un bucket Amazon S3.

Gli esempi seguenti utilizzano various AWS SDKs and the AWS CLI to call`DetectLabels`. Per informazioni sulla risposta dell'operazione `DetectLabels`, consulta DetectLabels - risposta.

Per rilevare le etichette in un'immagine

1. Se non lo hai già fatto:

   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Carica nel bucket S3 un'immagine contenente uno o più oggetti, ad esempio alberi, abitazioni e barche. L'immagine deve essere in formato .jpg o .png.

   Per le istruzioni, consulta Caricamento di oggetti in Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

3. Utilizzare i seguenti esempi per richiamare l'operazione `DetectLabels`.

   Java

   Questo esempio mostra un elenco di etichette rilevate nell'immagine di input. Sostituisci i valori `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2.

```
package com.amazonaws.samples;
import java.util.List;

import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.DetectLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.Instance;
import com.amazonaws.services.rekognition.model.Label;
import com.amazonaws.services.rekognition.model.Parent;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;

public class DetectLabels {

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

        String photo = "photo";
        String bucket = "bucket";

        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

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

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

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

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

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

AWS CLI

Questo esempio mostra l'output JSON dell'operazione CLI `detect-labels`. Sostituisci i valori `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2. Sostituisci il valore di `profile-name` con il nome del tuo profilo di sviluppatore.

```
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
```

Se accedi alla CLI da un dispositivo Windows, usa le virgolette doppie anziché le virgolette singole ed evita le virgolette doppie interne tramite barra rovesciata (ovvero, \) per risolvere eventuali errori del parser che potresti riscontrare. Per un esempio, consulta quanto segue:

```
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

Questo esempio mostra le etichette rilevate nell'immagine di input. Nella funzione `main`, sostituisci i valori di `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

Questo esempio mostra un elenco di etichette rilevate nell'immagine di input. Sostituisci i valori `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2.

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

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

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

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        DetectLabelsRequest detectlabelsRequest = new DetectLabelsRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
```

```
                {
                    Name = photo,
                    Bucket = bucket
                },
            },
            MaxLabels = 10,
            MinConfidence = 75F
        };

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

Ruby

Questo esempio mostra un elenco di etichette rilevate nell'immagine di input. Sostituisci i valori `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2.

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

```
    image: {
      s3_object: {
        bucket: bucket,
        name: photo
      },
    },
    max_labels: 10
  }
response = client.detect_labels attrs
puts "Detected labels for: #{photo}"
response.labels.each do |label|
  puts "Label:      #{label.name}"
  puts "Confidence: #{label.confidence}"
  puts "Instances:"
  label['instances'].each do |instance|
    box = instance['bounding_box']
    puts "  Bounding box:"
    puts "    Top:        #{box.top}"
    puts "    Left:       #{box.left}"
    puts "    Width:      #{box.width}"
    puts "    Height:     #{box.height}"
    puts "  Confidence: #{instance.confidence}"
  end
  puts "Parents:"
  label.parents.each do |parent|
    puts "  #{parent.name}"
  end
  puts "------------"
  puts ""
end
```

Node.js

Questo esempio mostra un elenco di etichette rilevate nell'immagine di input. Sostituisci i valori `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

Se si utilizzano TypeScript definizioni, potrebbe essere necessario utilizzare `import AWS from 'aws-sdk'` invece di`const AWS = require('aws-sdk')`, per eseguire il programma con Node.js. Puoi consultare l'AWS SDK per Javascript per maggiori dettagli.

A seconda di come sono state impostate le configurazioni, potrebbe essere necessario specificare anche la regione con `AWS.config.update({region:region});`.

```
// Load the SDK
var AWS = require('aws-sdk');
const bucket = 'bucket-name' // the bucketname without s3://
const photo  = 'image-name' // the name of file

var credentials = new AWS.SharedIniFileCredentials({profile: 'profile-name'});
AWS.config.credentials = credentials;
AWS.config.update({region:'region-name'});

const client = new AWS.Rekognition();
const params = {
  Image: {
    S3Object: {
      Bucket: bucket,
      Name: photo
    },
  },
  MaxLabels: 10
}
client.detectLabels(params, function(err, response) {
  if (err) {
    console.log(err, err.stack); // if an error occurred
  } else {
    console.log(`Detected labels for: ${photo}`)
    response.Labels.forEach(label => {
      console.log(`Label:      ${label.Name}`)
      console.log(`Confidence: ${label.Confidence}`)
      console.log("Instances:")
      label.Instances.forEach(instance => {
        let box = instance.BoundingBox
        console.log("  Bounding box:")
        console.log(`    Top:        ${box.Top}`)
        console.log(`    Left:       ${box.Left}`)
        console.log(`    Width:      ${box.Width}`)
        console.log(`    Height:     ${box.Height}`)
        console.log(`  Confidence: ${instance.Confidence}`)
      })
      console.log("Parents:")
```

```
      label.Parents.forEach(parent => {
        console.log(`  ${parent.Name}`)
      })
      console.log("------------")
      console.log("")
    }) // for response.labels
  } // if
});
```

Java V2

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
//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 richiesta di operazione

L'input per `DetectLabel` è un'immagine. In questo input JSON di esempio, l'immagine di origine viene caricata da un bucket Amazon S3. `MaxLabels` è il numero massimo di etichette da restituire in una risposta. `MinConfidence` è l'affidabilità minima che Immagini Amazon Rekognition deve avere nella precisione dell'etichetta rilevata per ottenerne la restituzione nella risposta.

Features consente di specificare una o più caratteristiche dell'immagine che si desidera restituire, consentendoti di selezionare `GENERAL_LABELS` e`IMAGE_PROPERTIES`. Includere `GENERAL_LABELS` restituirà le etichette rilevate nell'immagine di input, mentre includere `IMAGE_PROPERTIES` consentirà di accedere al colore e alla qualità dell'immagine.

Le impostazioni ti consentono di filtrare gli articoli restituiti sia per le funzionalità `GENERAL_LABELS` che per `IMAGE_PROPERTIES`. Per le etichette puoi utilizzare filtri inclusivi ed esclusivi. Puoi anche filtrare per etichette singole, specifiche per etichetta o per categoria di etichette:

- LabelInclusionFilters - Consente di specificare quali etichette si desidera includere nella risposta.
- LabelExclusionFilters - Consente di specificare quali etichette si desidera escludere dalla risposta.
- LabelCategoryInclusionFilters - Consente di specificare quali categorie di etichette si desidera includere nella risposta.

- LabelCategoryExclusionFilters - Consente di specificare quali categorie di etichette si desidera escludere dalla risposta.

Puoi anche combinare filtri inclusivi ed esclusivi in base alle tue esigenze, escludendo alcune etichette o categorie e includendone altre.

`IMAGE_PROPERTIES` fanno riferimento ai colori e agli attributi di qualità dominanti di un'immagine come nitidezza, luminosità e contrasto. Durante il rilevamento `IMAGE_PROPERTIES` è possibile specificare il numero massimo di colori dominanti da restituire (il valore predefinito è 10) utilizzando il parametro `MaxDominantColors`.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "MaxLabels": 10,
    "MinConfidence": 75,
    "Features": [ "GENERAL_LABELS", "IMAGE_PROPERTIES" ],
    "Settings": {
        "GeneralLabels": {
            "LabelInclusionFilters": [<Label(s)>],
            "LabelExclusionFilters": [<Label(s)>],
            "LabelCategoryInclusionFilters": [<Category Name(s)>],
            "LabelCategoryExclusionFilters": [<Category Name(s)>]
        },
        "ImageProperties": {
            "MaxDominantColors":10
        }
    }
}
```

DetectLabels - risposta

La risposta di `DetectLabels` è una matrice di etichette rilevate nell'immagine e il livello di affidabilità con cui sono state rilevate.

Di seguito è riportata una risposta di esempio da `DetectLabels`. La risposta di esempio seguente contiene una serie di attributi restituiti per GENERAL_LABELS, tra cui:

- Nome - Il nome dell'etichetta rilevata. In questo esempio, l'operazione ha rilevato un oggetto con l'etichetta Mobile Phone.
- Affidabilità - A ogni etichetta è associato un livello di affidabilità. In questo esempio, l'affidabilità dell'etichetta era del 99,36%.
- Genitori - le etichette degli antenati per un'etichetta rilevata. In questo esempio, l'etichetta Mobile Phone ha un'etichetta principale denominata Phone.
- Alias - Informazioni sui possibili alias per l'etichetta. In questo esempio, l'etichetta Mobile Phone ha un possibile alias di Cell Phone.
- Categorie - La categoria di etichette a cui appartiene l'etichetta rilevata. In questo esempio, si tratta di Tecnologia e informatica.

La risposta per le etichette relative a oggetti comuni include informazioni sul riquadro di delimitazione per la posizione dell'etichetta nell'immagine in input. Ad esempio, l'etichetta Persona dispone di una serie di istanze contenenti due riquadri di delimitazione che si riferiscono a due persone rilevate all'interno dell'immagine.

La risposta include anche gli attributi relativi a IMAGE_PROPERTIES. Gli attributi presentati dalla funzione IMAGE_PROPERTIES sono:

- Qualità - Informazioni su nitidezza, luminosità e contrasto dell'immagine in ingresso, con un punteggio compreso tra 0 e 100. La qualità viene riportata per l'intera immagine e per lo sfondo e il primo piano dell'immagine, se disponibili. Tuttavia, il contrasto viene riportato solo per l'intera immagine, mentre la nitidezza e la luminosità vengono riportate anche per lo sfondo e il primo piano.
- Colore dominante - Una serie di colori dominanti nell'immagine. Ogni colore dominante è descritto con un nome di colore semplificato, una tavolozza di colori CSS, valori RGB e un codice esadecimale.
- Primo piano - Informazioni sui colori, la nitidezza e la luminosità dominanti del primo piano dell'immagine di input.
- Sfondo - Informazioni sui colori, la nitidezza e la luminosità dominanti dello sfondo dell'immagine di input.

Quando GENERAL_LABELS e IMAGE_PROPERTIES vengono utilizzati insieme come parametri di input, Immagini Amazon Rekognition restituirà anche i colori dominanti degli oggetti con riquadri di delimitazione.

Il campo `LabelModelVersion` contiene il numero di versione del modello di rilevamento utilizzato da `DetectLabels`.


```
{

   "Labels": [
        {
            "Name": "Mobile Phone",
            "Parents": [
              {
                "Name": "Phone"
              }
            ],
            "Aliases": [
              {
                "Name": "Cell Phone"
              }
            ],
            "Categories": [
              {
                "Name": "Technology and Computing"
              }
            ],
            "Confidence": 99.9364013671875,
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.26779675483703613,
                        "Height": 0.8562285900115967,
                        "Left": 0.3604024350643158,
                        "Top": 0.09245597571134567,
                    }
                    "Confidence": 99.9364013671875,
                    "DominantColors": [
                    {
                "Red": 120,
                "Green": 137,
                "Blue": 132,
                "HexCode": "3A7432",
                "SimplifiedColor": "red",
                "CssColor": "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"
}
```

Trasformare la risposta DetectLabels

Quando si utilizza l' DetectLabels API, potrebbe essere necessario che la struttura di risposta imiti la precedente struttura di risposta dell'API, in cui sia le etichette primarie che gli alias erano contenuti nello stesso elenco.

Di seguito è riportato un esempio dell'attuale risposta dell'API di: DetectLabels

```
"Labels": [
        {
            "Name": "Mobile Phone",
            "Confidence": 99.99717712402344,
            "Instances": [],
            "Parents": [
                {
                "Name": "Phone"
                }
             ],
            "Aliases": [
                {
                "Name": "Cell Phone"
                }
             ]
        }
 ]
```

L'esempio seguente mostra la risposta precedente dell'DetectLabelsAPI:

```
"Labels": [
        {
            "Name": "Mobile Phone",
            "Confidence": 99.99717712402344,
            "Instances": [],
            "Parents": [
                {
                "Name": "Phone"
                }
             ]
         },
         {
            "Name": "Cell Phone",
            "Confidence": 99.99717712402344,
            "Instances": [],
            "Parents": [
                {
                "Name": "Phone"
                }
             ]
         },
]
```

Se necessario, puoi trasformare la risposta corrente in modo che segua il formato della risposta precedente. È possibile utilizzare il seguente codice di esempio per trasformare la risposta API più recente nella precedente struttura di risposta dell'API:

Python

Il seguente esempio di codice mostra come trasformare la risposta corrente dall' DetectLabels API. Nell'esempio di codice riportato di seguito, è possibile sostituire il valore di *EXAMPLE_INFERENCE_OUTPUT* con l'output di un' DetectLabels operazione eseguita.

```
from copy import deepcopy

LABEL_KEY = "Labels"
ALIASES_KEY = "Aliases"
INSTANCE_KEY = "Instances"
NAME_KEY = "Name"
```

```
#Latest API response sample
EXAMPLE_INFERENCE_OUTPUT = {
    "Labels": [
        {
            "Name": "Mobile Phone",
            "Confidence": 97.530106,
            "Categories": [
                {
                    "Name": "Technology and Computing"
                }
            ],
            "Aliases": [
                {
                    "Name": "Cell Phone"
                }
            ],
            "Instances":[
                {
                    "BoundingBox":{
                        "Height":0.1549897,
                        "Width":0.07747964,
                        "Top":0.50858885,
                        "Left":0.00018205095
                    },
                    "Confidence":98.401276
                }
            ]
        },
        {
            "Name": "Urban",
            "Confidence": 99.99982,
            "Categories": [
                "Colors and Visual Composition"
            ]
        }
    ]
}

def expand_aliases(inferenceOutputsWithAliases):

    if LABEL_KEY in inferenceOutputsWithAliases:
        expandInferenceOutputs = []
        for primaryLabelDict in inferenceOutputsWithAliases[LABEL_KEY]:
            if ALIASES_KEY in primaryLabelDict:
```

```
                for alias in primaryLabelDict[ALIASES_KEY]:
                    aliasLabelDict = deepcopy(primaryLabelDict)
                    aliasLabelDict[NAME_KEY] = alias[NAME_KEY]
                    del aliasLabelDict[ALIASES_KEY]
                    if INSTANCE_KEY in aliasLabelDict:
                        del aliasLabelDict[INSTANCE_KEY]
                    expandInferenceOutputs.append(aliasLabelDict)

        inferenceOutputsWithAliases[LABEL_KEY].extend(expandInferenceOutputs)

    return inferenceOutputsWithAliases


if __name__ == "__main__":

    outputWithExpandAliases = expand_aliases(EXAMPLE_INFERENCE_OUTPUT)
    print(outputWithExpandAliases)
```

Di seguito è riportato un esempio della risposta trasformata:

```
#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"
            ]
        }
    ]
}
```

Rilevamento di etichette in un video

Video Amazon Rekognition è in grado di rilevare etichette (oggetti e concetti) e l'ora in cui viene rilevata un'etichetta in un video. Per un esempio di codice SDK, consultare Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK). Per un AWS CLI esempio, vediAnalisi di un video con AWS Command Line Interface.

Il rilevamento dell'etichetta di Video Amazon Rekognition è un'operazione asincrona. Per avviare il rilevamento delle etichette in un video, chiama StartLabelDetection.

Amazon Rekognition per video pubblica lo stato di completamento dell'analisi video in un argomento Amazon Simple Notification Service. Se l'analisi video ha esito positivo, chiamare GetLabelDetection per ottenere le etichette rilevate. Per informazioni su come richiamare le operazioni dell'API di analisi video, consulta Chiamata delle operazioni Video Amazon Rekognition.

StartLabelDetectionRichiesta

L'esempio seguente è una richiesta per l'operazione `StartLabelDetection`. Fornisci l'operazione `StartLabelDetection` con un video archiviato in un bucket Amazon S3. Nella richiesta di esempio vengono specificati JSON, il bucket Amazon S3 e il nome del video, oltre a `MinConfidence`, `Features`, `Settings` e `NotificationChannel`.

`MinConfidence` è l'affidabilità minima che Video Amazon Rekognition deve avere nella precisione dell'etichetta rilevata o un'istanza di riquadro di delimitazione (se rilevato) per ottenerne la restituzione nella risposta.

Con `Features`, puoi specificare che desideri che GENERAL_LABELS venga restituito come parte della risposta.

Con `Settings`, puoi filtrare gli articoli restituiti per GENERAL_LABELS. Per le etichette puoi utilizzare filtri inclusivi ed esclusivi. Puoi anche filtrare per etichette singole, specifiche per etichetta o per categoria di etichette:

- `LabelInclusionFilters` - Usato per specificare quali etichette si desidera includere nella risposta
- `LabelExclusionFilters` - Usato per specificare quali etichette si desidera escludere dalla risposta.
- `LabelCategoryInclusionFilters` - Usato per specificare quali categorie di etichette si desidera includere nella risposta.
- `LabelCategoryExclusionFilters` - Usato per specificare quali categorie di etichette si desidera escludere dalla risposta.

Puoi anche combinare filtri inclusivi ed esclusivi in base alle tue esigenze, escludendo alcune etichette o categorie e includendone altre.

`NotificationChannel` è l'ARN dell'argomento Amazon SNS su cui desideri che Video Amazon Rekognition pubblichi lo stato di completamento dell'operazione di rilevamento delle etichette. Se utilizzi la politica delle autorizzazioni `AmazonRekognitionServiceRole`, l'argomento Amazon SNS deve avere un nome che inizi con Rekognition.

Di seguito è riportato un esempio di richiesta `StartLabelDetection` in formato JSON, compresi i filtri:

```
{
    "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 Risposta all'operazione

`GetLabelDetection` restituisce una matrice (`Labels`) che contiene informazioni sulle etichette rilevati nel video. L'array può essere ordinato in base all'ora o in base all'etichetta rilevata quando si specifica il parametro `SortBy`. È inoltre possibile selezionare il modo in cui gli elementi di risposta vengono aggregati utilizzando il parametro `AggregateBy`.

Di seguito è riportato un esempio di risposta JSON dell'operazione `GetLabelDetection`. Nella risposta, tenere presente quanto segue:

- Ordinamento - La matrice di etichette restituite viene ordinata in base all'ora. Per ordinare in base all'etichetta, specificare `NAME` nel parametro di input `SortBy` per `GetLabelDetection`. Se l'etichetta viene visualizzata più volte nel video, saranno presenti più istanze dell'elemento (LabelDetection). L'ordinamento predefinito è `TIMESTAMP`, mentre l'ordinamento secondario è `NAME`.

- Informazioni sull'etichetta - L'elemento della matrice `LabelDetection` contiene un (Etichetta) oggetto che a sua volta include il nome dell'etichetta e il livello di affidabilità dell'etichetta rilevata da Amazon Rekognition. Un oggetto `Label` include anche una tassonomia gerarchica delle etichette e le informazioni sul riquadro di delimitazione per le etichette comuni. `Timestamp` è l'orario di rilevamento dell'etichetta, definito come numero in millisecondi trascorsi dall'inizio del video.

  Vengono inoltre restituite informazioni su eventuali categorie o alias associati a un'etichetta. Per i risultati aggregati per video `SEGMENTS`, vengono restituite le strutture `StartTimestampMillis`, `EndTimestampMillis` e `DurationMillis` che definiscono rispettivamente l'ora di inizio, l'ora di fine e la durata di un segmento.
- Aggregazione – Specifica in che modo i risultati vengono aggregati quando vengono restituiti. L'impostazione predefinita prevede l'aggregazione entro `TIMESTAMPS`. Puoi anche scegliere di aggregare per `SEGMENTS`, che aggrega i risultati in una finestra temporale. Se si aggrega per `SEGMENTS`, le informazioni sulle istanze rilevate con riquadri di delimitazione non vengono restituite. Vengono restituite solo le etichette rilevate durante i segmenti.
- Informazioni di paginazione - L'esempio illustra una pagina di informazioni di rilevamento dell'etichetta. È possibile specificare il numero di oggetti `LabelDetection` da restituire nel parametro di input `MaxResults` per `GetLabelDetection`. Se esiste un numero di risultati maggiore di `MaxResults`, `GetLabelDetection` restituisce un token (`NextToken`) utilizzato per ottenere la pagina di risultati successiva. Per ulteriori informazioni, consulta Ottenere i risultati dell'analisi di Video Amazon Rekognition.
- Informazioni video – La risposta include informazioni sul formato video (`VideoMetadata`) in ogni pagina di informazioni restituita da `GetLabelDetection`.

Di seguito è riportato un esempio di GetLabelDetection risposta in formato JSON con aggregazione tramite 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
    }
}
```

Di seguito è riportato un esempio di GetLabelDetection risposta in formato JSON con aggregazione per SEGMENTS:

```
{
    "JobStatus": "SUCCEEDED",
    "LabelModelVersion": "3.0",
    "Labels": [
        {
            "StartTimestampMillis": 225,
            "EndTimestampMillis": 3578,
            "DurationMillis": 3353,
            "Label": {
                "Name": "Car",
                "Categories": [
                  {
                    "Name": "Vehicles and Automotive"
                  }
                ],
                "Aliases": [
                  {
                    "Name": "Automobile"
                  }
                ],
                "Parents": [
                  {
                    "Name": "Vehicle"
                  }
                ],
                "Confidence": 99.9364013671875 // Maximum confidence score for Segment
 mode
            }
        },
        {
            "StartTimestampMillis": 7578,
            "EndTimestampMillis": 12371,
            "DurationMillis": 4793,
            "Label": {
                "Name": "Kangaroo",
                "Categories": [
                  {
                    "Name": "Animals and Pets"
                  }
                ],
```

```
                "Aliases": [
                  {
                    "Name": "Wallaby"
                  }
                ],
                "Parents": [
                  {
                    "Name": "Mammal"
                  }
                ],
                "Confidence": 99.9364013671875
            }
        },
        {
            "StartTimestampMillis": 22225,
            "EndTimestampMillis": 22578,
            "DurationMillis": 2353,
            "Label": {
                "Name": "Bicycle",
                "Categories": [
                  {
                    "Name": "Hobbies and Interests"
                  }
                ],
                "Aliases": [
                  {
                    "Name": "Bike"
                  }
                ],
                "Parents": [
                  {
                    "Name": "Vehicle"
                  }
                ],
                "Confidence": 99.9364013671875
            }
        }
    ],
    "VideoMetadata": {
        "ColorRange": "FULL",
        "DurationMillis": 5000,
        "Format": "MP4",
        "FrameWidth": 1280,
        "FrameHeight": 720,
```

```
        "FrameRate": 24
    }
}
```

Trasformazione della risposta GetLabelDetection

Quando si recuperano i risultati con l'operazione GetLabelDetection API, potrebbe essere necessario che la struttura di risposta imiti la precedente struttura di risposta dell'API, in cui sia le etichette primarie che gli alias erano contenuti nello stesso elenco.

L'esempio di risposta JSON riportato nella sezione precedente mostra il formato corrente della risposta API di. GetLabelDetection

L'esempio seguente mostra la risposta precedente dell' 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
    }
}
```

Se necessario, puoi trasformare la risposta corrente in modo che segua il formato della risposta precedente. È possibile utilizzare il seguente codice di esempio per trasformare la risposta API più recente nella precedente struttura di risposta dell'API:

```
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
```

Rilevamento delle etichette negli eventi video in streaming

Puoi usare Video Amazon Rekognition per rilevare le etichette nei video in streaming. A tale scopo, create uno stream processor (CreateStreamProcessor) per avviare e gestire l'analisi dei video in streaming.

Video Amazon Rekognition usa Flusso di video Amazon Kinesis per ricevere e processare un flusso video. Quando crei un elaboratore di flussi, scegli ciò che desidera che il processore del flusso rilevi. Puoi scegliere persone, pacchi e animali domestici oppure persone e pacchi. I risultati dell'analisi vengono inviati al bucket Amazon S3 e nelle notifiche Amazon SNS. Tieni presente che Video Amazon Rekognition rileva la presenza di una persona nel video, ma non rileva se si tratta di una persona specifica. Per cercare un volto da una raccolta in un video in streaming, consulta the section called “Ricerca di volti in una raccolta in streaming video”.

Per usare Video Amazon Rekognition con un video in streaming, l'applicazione necessita di quanto segue:

- Un flusso video Kinesis per l'invio di video in streaming ad Video Amazon Rekognition. Per ulteriori informazioni, consulta la Guida per gli sviluppatori di Flusso di video Amazon Kinesis.

- Un elaboratore di flussi Video Amazon Rekognition per gestire l'analisi del video in streaming. Per ulteriori informazioni, consulta Panoramica delle operazioni del processore di streaming Video Amazon Rekognition.
- Un bucket Amazon S3. Video Amazon Rekognition pubblica l'output della sessione nel bucket S3. L'output include i frame dell'immagine in cui una persona o un oggetto di interesse è stato rilevato per la prima volta. È necessario essere proprietari del bucket S3.
- Un argomento di Amazon SNS su cui Amazon Rekognition Video pubblica avvisi intelligenti e un riepilogo. end-of-session

Argomenti

- Configurazione delle risorse Video Amazon Rekognition e Amazon Kinesis
- Chiamare le operazioni di rilevamento delle etichette per eventi video in streaming

Configurazione delle risorse Video Amazon Rekognition e Amazon Kinesis

Le seguenti procedure descrivono i passaggi da seguire per effettuare il provisioning del flusso video Kinesis e altre risorse utilizzate per rilevare le etichette in un video in streaming.

Prerequisiti

Per eseguire questa procedura, è necessario che sia installato. AWS SDK per Java Per ulteriori informazioni, consulta Nozioni di base su Amazon Rekognition. L'account AWS che usi richiede le autorizzazioni di accesso all'API Amazon Rekognition. Per ulteriori informazioni, consulta la sezione Operazioni definite da Amazon Rekognition nella Guida per l'utente IAM.

Per rilevare le etichette in un flusso video (AWS SDK)

1. Creare un bucket Amazon S3. Prendi nota del nome del bucket e di tutti i prefissi della chiave che desideri utilizzare. Queste informazioni serviranno in seguito.
2. Creazione di un argomento Amazon SNS. Puoi utilizzarlo per ricevere notifiche quando un oggetto di interesse viene rilevato per la prima volta nel flusso video. Prendi nota del nome della risorsa Amazon (ARN) per l’argomento. Per ulteriori informazioni, consulta Creazione di un argomento Amazon SNS nella Guida per gli sviluppatori Amazon SNS.
3. Iscrivere un endpoint all’argomento Amazon SNS. Per ulteriori informazioni, consulta Sottoscrizione a un argomento di Amazon SNS nella Guida per sviluppatori di Amazon SNS.
4. Crea un flusso video Kinesis e prendi nota del nome della risorsa Amazon (ARN) del flusso.

5. Se non l'hai già fatto, crea un ruolo di servizio IAM per consentire ad Video Amazon Rekognition di accedere ai tuoi flussi video Kinesis, al tuo bucket S3 e al tuo argomento Amazon SNS. Per ulteriori informazioni, consulta Concessione dell'accesso ai processori di flussi di rilevamento delle etichette.

Puoi quindi creare il processore di flussi di rilevamento delle etichette e avviare il processore di flussi utilizzando il nome del processore di flussi scelto.

> **Note**
>
> Avvia l'elaboratore di flussi solo dopo aver verificato la possibilità di importare contenuti multimediali nel flusso video Kinesis.

Orientamento e configurazione della fotocamera

Video Amazon Rekognition Streaming Video Events può supportare tutte le videocamere supportate da Kinesis Video Streams. Per ottenere risultati ottimali, consigliamo di posizionare la videocamera a una distanza compresa tra 0 e 45 gradi da terra. La fotocamera deve essere nella sua posizione verticale canonica. Ad esempio, se c'è una persona nell'inquadratura, la persona dovrebbe essere orientata verticalmente e la testa della persona dovrebbe essere più alta nell'inquadratura rispetto ai piedi.

Concessione dell'accesso ai processori di flussi di rilevamento delle etichette

Utilizzi un ruolo di servizio AWS Identity and Access Management (IAM) per consentire ad Amazon Rekognition Video l'accesso in lettura ai flussi video Kinesis. A tale scopo, utilizza i ruoli IAM per consentire ad Video Amazon Rekognition di accedere al tuo bucket Amazon S3 e a un argomento di Amazon SNS.

Puoi creare una policy di autorizzazione che consente a Video Amazon Rekognition l'accesso a un argomento Amazon SNS esistente, bucket Amazon S3 e flusso video Kinesis. Per una step-by-step procedura che utilizza il, consulta. AWS CLIthe section called “AWS CLI comandi per configurare un ruolo IAM per il rilevamento delle etichette”

Consentire ad Video Amazon Rekognition di accedere alle risorse per il rilevamento delle etichette

1. Crea una nuova policy di autorizzazione con l'editor delle policy JSON IAM e utilizza la policy seguente. Sostituisci `kvs-stream-name` con il nome dello flusso video Kinesis, `topicarn`

con il nome della risorsa Amazon (ARN) dell'argomento Amazon SNS che desideri utilizzare e `bucket-name` con il nome del bucket Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "KinesisVideoPermissions",
            "Effect": "Allow",
            "Action": [
                "kinesisvideo:GetDataEndpoint",
                "kinesisvideo:GetMedia"
            ],
            "Resource": [
                "arn:aws:kinesisvideo:::stream/kvs-stream-name/*"
            ]
        },
        {
            "Sid": "SNSPermissions",
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:::sns-topic-name"
            ]
        },
        {
            "Sid": "S3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

2. Crea un ruolo di servizio IAM o aggiornane uno esistente. Per creare il ruolo di servizio IAM, utilizza le informazioni seguenti:

   1. Scegli Rekognition come nome del servizio.
   2. Scegli Rekognition come caso d'uso del ruolo del servizio.
   3. Collega la policy di autorizzazione creata nella fase 1.

3. Prendere nota dell'ARN del ruolo del servizio. È necessario per creare il processore di flussi prima di eseguire operazioni di analisi video.

4. (Facoltativo) Se utilizzi la tua AWS KMS chiave per crittografare i dati inviati al tuo bucket S3, devi aggiungere la seguente istruzione con il ruolo IAM. (Questo è il ruolo IAM che hai creato per la policy della chiave gestita dal cliente che desideri utilizzare.)

```
            {
                        "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 comandi per configurare un ruolo IAM per il rilevamento delle etichette

Se non l'hai già fatto, configurali e configurali AWS CLI con le tue credenziali.

Inserisci i seguenti comandi in AWS CLI per configurare un ruolo IAM con le autorizzazioni necessarie per il rilevamento delle etichette.

1. `export IAM_ROLE_NAME=labels-test-role`
2. `export AWS_REGION=us-east-1`

3. Crea un file di policy sulle relazioni di fiducia (ad esempio, assume-role-rekognition .json) con il seguente contenuto.

```
{
  "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. Se il nome dell'argomento SNS con cui desideri ricevere notifiche non inizia con il prefisso "AmazonRekognition", aggiungi la seguente politica:

   `aws iam attach-role-policy --role-name $IAM_ROLE_NAME --policy-arn "arn:aws:iam::aws:policy/AmazonSNSFullAccess" --region $AWS_REGION`
7. Se utilizzi la tua chiave AWS KMS per crittografare i dati inviati al bucket Amazon S3, aggiorna la policy della chiave gestita dal cliente che desideri utilizzare.

   a. Crea un file kms_key_policy.json che contenga il seguente contenuto:

```
{
"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`. Sostituisci KMS_KEY_ID con l'ID della chiave KMS che hai creato.

c. `aws kms put-key-policy --policy-name default --key-id $KMS_KEY_ID --policy file://path-to-kms-key-policy.json`

Chiamare le operazioni di rilevamento delle etichette per eventi video in streaming

Video Amazon Rekognition è in grado di rilevare persone o oggetti pertinenti in un video in streaming e avvisarti quando vengono rilevati. Quando crei uno stream processor per il rilevamento delle etichette, scegli quali etichette vuoi che Video Amazon Rekognition rilevi. Possono essere persone, pacchi e animali domestici oppure persone, pacchi e animali domestici. Scegli solo le etichette specifiche che desideri rilevare. In questo modo, le uniche etichette pertinenti creano notifiche. È possibile configurare le opzioni per determinare quando archiviare le informazioni video e quindi eseguire un'ulteriore elaborazione in base alle etichette rilevate nel frame.

Dopo aver impostato le risorse, il processo per rilevare le etichette in un video in streaming è il seguente:

1. Creare l'elaboratore di flussi
2. Avvia l'elaboratore di flussi
3. Se viene rilevato un oggetto di interesse, ricevi una notifica Amazon SNS per la prima occorrenza di ogni oggetto di interesse.
4. Lo stream processor si arresta quando il tempo specificato in `MaxDurationInSeconds` è completo.
5. Riceverai una notifica finale di Amazon SNS con un riepilogo dell'evento.
6. Video Amazon Rekognition pubblica l'output della sessione nel bucket S3.

Argomenti

- Creazione dell'elaboratore di flussi per il rilevamento delle etichette Video Amazon Rekognition
- Avvio dell'elaboratore di flussi per il rilevamento delle etichette Video Amazon Rekognition
- Analisi dei risultati del rilevamento delle etichette

Creazione dell'elaboratore di flussi per il rilevamento delle etichette Video Amazon Rekognition

Prima di poter analizzare un video in streaming, devi creare uno stream processor Amazon Rekognition Video (). CreateStreamProcessor

Se desideri creare uno elaboratore di flussi per rilevare etichette di interesse e persone, fornisci come input un flusso video Kinesis (`Input`), informazioni sul bucket Amazon S3 (`Output`) e un argomento di Amazon SNS ARN (`StreamProcessorNotificationChannel`). Puoi inoltre fornire un ID chiave KMS per crittografare i dati inviati al bucket S3. Nella sezione dell'utente è possibile indicare gli oggetti da rilevare `Settings`, ad esempio persone, pacchetti e persone o animali domestici, persone e pacchetti. Puoi anche specificare la parte del frame con cui vuoi che Amazon Rekognition effettui il monitoraggio `RegionsOfInterest`. Di seguito è riportato un esempio di JSON per la richiesta `CreateStreamProcessor`.

```
{
  "DataSharingPreference": { "OptIn":TRUE
  },
  "Input": {
    "KinesisVideoStream": {
      "Arn": "arn:aws:kinesisvideo:us-east-1:nnnnnnnnnnnn:stream/muh_video_stream/
nnnnnnnnnnnn"
    }
  },
  "KmsKeyId": "muhkey",
  "Name": "muh-default_stream_processor",
  "Output": {
    "S3Destination": {
      "Bucket": "s3bucket",
      "KeyPrefix": "s3prefix"
    }
  },
  "NotificationChannel": {
```

```
    "SNSTopicArn": "arn:aws:sns:us-east-2:nnnnnnnnnnnn:MyTopic"
  },
  "RoleArn": "arn:aws:iam::nnnnnnnnn:role/Admin",
  "Settings": {
    "ConnectedHome": {
      "Labels": [
        "PET"
      ]
    "MinConfidence": 80
    }
  },
  "RegionsOfInterest": [
    {
      "BoundingBox": {
        "Top": 0.11,
        "Left": 0.22,
        "Width": 0.33,
        "Height": 0.44
      }
    },
    {
      "Polygon": [
        {
          "X": 0.11,
          "Y": 0.11
        },
        {
          "X": 0.22,
          "Y": 0.22
        },
        {
          "X": 0.33,
          "Y": 0.33
        }
      ]
    }
  ]
}
```

Tieni presente che puoi modificare il valore `MinConfidence` quando specifichi il `ConnectedHomeSettings` per l'elaboratore di flussi. `MinConfidence` è un valore numerico

compreso tra 0 e 100 che indica la certezza dell'algoritmo in merito alle sue previsioni. Ad esempio, una notifica `person` con un valore di confidenza di 90 indica che l'algoritmo è assolutamente certo che la persona sia presente nel video. Un valore di confidenza pari a 10 indica che potrebbe esserci una persona. È possibile impostare un valore desiderato `MinConfidence` a scelta tra 0 e 100 a seconda della frequenza con cui si desidera ricevere la notifica. Ad esempio, se desideri ricevere una notifica solo quando Rekognition è assolutamente certa che ci sia un pacchetto nel fotogramma video, puoi impostare `MinConfidence` su 90.

Per impostazione predefinita, `MinConfidence` è impostato su 50. Se desideri ottimizzare l'algoritmo per una maggiore precisione, puoi impostarlo `MinConfidence` su un valore superiore a 50. Riceverai quindi meno notifiche, ma ogni notifica è più affidabile. Se desideri ottimizzare l'algoritmo per aumentare i richiami, puoi impostarlo su `MinConfidence` un valore inferiore a 50 per ricevere più notifiche.

Avvio dell'elaboratore di flussi per il rilevamento delle etichette Video Amazon Rekognition

Per iniziare ad analizzare il video in streaming, chiamare StartStreamProcessor con il nome dell'elaboratore di flussi specificato in `CreateStreamProcessor`. Quando esegui l'operazione `StartStreamProcessor` su un elaboratore di flussi di rilevamento delle etichette, inserisci le informazioni di inizio e fine per determinare il tempo di elaborazione.

Quando si avvia l'elaboratore di flussi, lo stato dell'elaboratore di flussi di rilevamento delle etichette cambia nei seguenti modi:

1. Quando chiami `StartStreamProcessor`, lo stato dell'elaboratore di flussi di rilevamento delle etichette varia da `STOPPED` o `FAILED` a `STARTING`.
2. Mentre è in funzione, l'elaboratore di flussi di rilevamento delle etichette rimane acceso `STARTING`.
3. Quando l'elaboratore di flussi di rilevamento delle etichette ha concluso il processo, lo stato diventa `STOPPED` o`FAILED`.

`StartSelector` specifica il punto di partenza nel flusso Kinesis per avviare l'elaborazione. Puoi usare il timestamp di KVS Producer o il numero di KVS Fragment. Per ulteriori informazioni, consulta Fragment.

> **Note**
>
> Se usi il timestamp di KVS Producer, devi inserire l'ora in millisecondi.

`StopSelector` specifica quando interrompere l'elaborazione del flusso. È possibile specificare un tempo massimo per l'elaborazione del video. L'impostazione predefinita è una durata massima di 10 secondi. Nota che il tempo di elaborazione effettivo potrebbe essere un po' più lungo della durata massima, a seconda della dimensione dei singoli frammenti KVS. Se la durata massima è stata raggiunta o superata alla fine di un frammento, il tempo di elaborazione si interrompe.

Di seguito è riportato un esempio di JSON per la richiesta `StartStreamProcessor`.

```
{
   "Name": "string",
   "StartSelector": {
     "KVSStreamStartSelector": {
         "KVSProducerTimestamp": 1655930623123
      },
        "StopSelector": {
            "MaxDurationInSeconds": 11
      }
   }
}
```

All'avvio dell'elaboratore di flussi viene restituita una risposta HTTP 200. È incluso un corpo JSON vuoto.

Analisi dei risultati del rilevamento delle etichette

Esistono tre modi in cui Amazon Rekognition Video pubblica le notifiche da un processore di stream di rilevamento delle etichette: notifiche Amazon SNS per eventi di rilevamento di oggetti, una notifica Amazon SNS per un riepilogo end-of-session e un rapporto dettagliato sui bucket Amazon S3.

- Notifiche Amazon SNS per eventi di rilevamento di oggetti.

  Se vengono rilevate etichette nel flusso video, ricevi notifiche Amazon SNS per gli eventi di rilevamento degli oggetti. Amazon Rekognition pubblica una notifica la prima volta che un oggetto di interesse o una persona viene rilevata nel flusso video. Le notifiche includono informazioni come

il tipo di etichetta rilevata, la confidenza e un link all'immagine del protagonista. Includono anche un'immagine ritagliata della persona o dell'oggetto rilevato e un timestamp del rilevamento. La notifica ha il formato seguente:

```
{"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
    }
}
```

- Riepilogo di Amazon SNS. end-of-session

  Al termine della sessione di elaborazione del flusso, inoltre, ricevi una notifica Amazon SNS. Questa notifica elenca i metadati per la sessione. I dettagli includono la durata del flusso che è stato elaborato. La notifica ha il formato seguente:

```
{"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
    }
}
```

- Report del bucket Amazon S3.

  Video Amazon Rekognition pubblica i risultati dell'operazione di analisi video nel bucket Amazon S3 forniti dall'operazione `CreateStreamProcessor`. Questi risultati includono i frame dell'immagine in cui una persona o un oggetto di interesse è stato rilevato per la prima volta.

  I frame sono disponibili in S3 nel seguente percorso: ObjectKeyPrefix//StreamProcessorName/SessionId. *`service_determined_unique_path`* In questo percorso, LabelKeyPrefixc'è un argomento opzionale fornito dal cliente, StreamProcessorNameè il nome della risorsa dello stream

processor ed SessionIdè un ID univoco per la sessione di elaborazione dello stream. Sostituiscili in base alla tua situazione.

Rilevamento di Custom Labels

Etichette personalizzate Amazon Rekognition può identificare gli oggetti e le scene nelle immagini secondo le tue necessità aziendali, come loghi o parti di macchine. Per ulteriori informazioni, consulta la pagina relativa a Cos'è Etichette personalizzate Amazon Rekognition? nella Guida per gli sviluppatori di etichette personalizzate Amazon Rekognition.

Rilevamento e analisi facciale

Amazon Rekognition APIs ti consente di rilevare e analizzare volti in immagini e video. Questa sezione fornisce una panoramica delle operazioni diverse dallo storage per l'analisi facciale. Queste operazioni includono funzionalità come il rilevamento di punti di riferimento facciali, l'analisi delle emozioni e il confronto dei volti.

Amazon Rekognition è in grado di identificare i punti di riferimento del viso (ad esempio, la posizione degli occhi), rilevare emozioni (ad esempio felicità o tristezza) e altri attributi (ad esempio, presenza di occhiali, occlusione facciale). Quando viene rilevato un volto, il sistema analizza gli attributi facciali e restituisce un punteggio di confidenza per ogni attributo.

Questa sezione contiene esempi di operazioni con immagini e video.

Per ulteriori informazioni sull'utilizzo delle operazioni sulle immagini di Rekognition, vedere. Lavorare con le immagini

Per ulteriori informazioni sull'utilizzo delle operazioni video di Rekognition, vedere. Utilizzo delle operazioni di analisi video archiviate

Nota che queste operazioni non sono operazioni di archiviazione. Puoi utilizzare le operazioni di archiviazione e le raccolte di volti per salvare i metadati dei volti rilevati in un'immagine. In seguito, puoi cercare i volti memorizzati sia nelle immagini che nei video. Ad esempio, questa funzionalità consente la ricerca per una persona specifica in un video. Per ulteriori informazioni, consulta Ricerca di volti in una raccolta.

Per ulteriori informazioni, consulta la sezione Faces di Amazon Rekognition FAQs.

> **Note**
>
> I modelli di rilevamento facciale utilizzati da Immagini Amazon Rekognition e Video Amazon Rekognition non supportano il rilevamento di volti in caratteri cartoni animati o entità non umane. Se si desidera rilevare i caratteri dei cartoni animati nelle immagini o nei video, si consiglia di utilizzare Etichette personalizzate Amazon Rekognition. Per ulteriori informazioni, consulta la guida per gli sviluppatori Etichette personalizzate Amazon Rekognition.

Argomenti

- Panoramica del rilevamento e del confronto facciale
- Linee guida sugli attributi facciali
- Rilevamento di volti in un'immagine
- Confronto dei volti nelle immagini
- Rilevamento di volti in un video archiviato

Panoramica del rilevamento e del confronto facciale

Amazon Rekognition offre agli utenti l'accesso a due applicazioni principali di machine learning per immagini contenenti volti: rilevamento e confronto di volti. Offrono funzionalità cruciali come l'analisi facciale e la verifica dell'identità, rendendole fondamentali per varie applicazioni, dalla sicurezza all'organizzazione delle foto personali.

Rilevamento dei volti

Un sistema di riconoscimento facciale risponde alla domanda: «C'è un volto in questa foto?» Gli aspetti chiave del riconoscimento facciale includono:

- Posizione e orientamento: determina la presenza, la posizione, la scala e l'orientamento dei volti nelle immagini o nei fotogrammi video.
- Attributi del viso: rileva i volti indipendentemente da attributi come sesso, età o peli sul viso.
- Informazioni aggiuntive: fornisce dettagli sull'occlusione del viso e sulla direzione dello sguardo.

Confronto facciale

Un sistema di confronto facciale si concentra sulla domanda: «Il volto in un'immagine corrisponde a un volto in un'altra immagine?» Le funzionalità del sistema di confronto facciale includono:

- Previsioni sulla corrispondenza dei volti: confronta un volto in un'immagine con un volto in un database fornito per prevedere le corrispondenze.
- Gestione degli attributi dei volti: gestisce gli attributi per confrontare i volti indipendentemente dall'espressione, dai peli del viso e dall'età.

Punteggi di confidenza e rilevamenti mancati

Sia i sistemi di riconoscimento facciale che quelli di confronto facciale utilizzano punteggi di confidenza. Un punteggio di confidenza indica la probabilità di previsioni, come la presenza di un volto o una corrispondenza tra volti. I punteggi più alti indicano una maggiore probabilità. Ad esempio, una confidenza del 90% suggerisce una maggiore probabilità di un rilevamento o di una corrispondenza corretti rispetto al 60%.

Se un sistema di riconoscimento facciale non rileva correttamente un volto o fornisce una previsione con scarsa attendibilità per un volto reale, si tratta di un dato detection/false negative. If the system incorrectly predicts the presence of a face at a high confidence level, this is a false alarm/false positivo perso.

Analogamente, un sistema di confronto facciale potrebbe non corrispondere a due volti che appartengono alla stessa persona (detection/false negative) or may incorrectly predict that two faces from different people are the same person (false alarm/falserisultato negativo mancato).

Progettazione dell'applicazione e impostazione delle soglie

- È possibile impostare una soglia che specifica il livello di confidenza minimo richiesto per restituire un risultato. La scelta delle soglie di confidenza appropriate è essenziale per la progettazione delle applicazioni e il processo decisionale in base agli output del sistema.

- Il livello di confidenza scelto deve riflettere il caso d'uso. Alcuni esempi di casi d'uso e soglie di confidenza:
- • Applicazioni fotografiche: una soglia inferiore (ad esempio 80%) potrebbe essere sufficiente per identificare i membri della famiglia nelle foto.
  - Scenari ad alto rischio: nei casi d'uso in cui il rischio di mancato rilevamento o di falsi allarmi è maggiore, come le applicazioni di sicurezza, il sistema dovrebbe utilizzare un livello di confidenza più elevato. In questi casi, si consiglia una soglia più alta (ad esempio il 99%) per abbinamenti facciali accurati.

Per ulteriori informazioni sull'impostazione e la comprensione delle soglie di confidenza, vedere. Ricerca di volti in una raccolta

Linee guida sugli attributi facciali

Di seguito sono riportate informazioni specifiche su come Amazon Rekognition elabora e restituisce gli attributi del volto.

- FaceDetail Oggetto: per ogni faccia rilevata, viene restituito un FaceDetail oggetto. FaceDetail Contiene dati sui punti di riferimento del viso, sulla qualità, sulla posa e altro ancora.
- Previsioni degli attributi: vengono previsti attributi come emozione, sesso, età e altri. A ogni previsione viene assegnato un livello di confidenza e le previsioni vengono restituite con il rispettivo punteggio di confidenza. Una soglia di confidenza del 99% è consigliata per casi d'uso sensibili. Per la stima dell'età, il punto medio della fascia di età prevista offre l'approssimazione migliore.

Tieni presente che le previsioni di genere ed emozione si basano sull'aspetto fisico e non devono essere utilizzate per determinare l'identità di genere o lo stato emotivo effettivi. Una previsione binaria di genere (maschio / femmina) si basa sull'aspetto fisico di un volto in una particolare immagine. Non indica l'identità di genere di una persona e non dovresti usare Rekognition per fare una tale determinazione. Non è consigliabile utilizzare le previsioni binarie di genere per prendere decisioni che influiscono sui diritti, sulla privacy o sull'accesso ai servizi di un individuo. Allo stesso modo, una previsione di un'emozione non indica l'effettivo stato emotivo interno di una persona, e non dovresti usare Rekognition per determinare tale situazione. Una persona che finge di avere una faccia felice in una foto potrebbe sembrare felice, ma potrebbe non provare felicità.

Casi di applicazione e utilizzo

Ecco alcune applicazioni pratiche e casi d'uso per questi attributi:

- Applicazioni: attributi come Smile, Pose e Sharpness possono essere utilizzati per selezionare le immagini del profilo o stimare i dati demografici in modo anonimo.
- Casi d'uso comuni: le applicazioni per i social media e la stima demografica in occasione di eventi o negozi al dettaglio sono esempi tipici.

Per informazioni più dettagliate su ciascun attributo, vedere FaceDetail.

Rilevamento di volti in un'immagine

Amazon Rekognition Image DetectFacesfornisce l'operazione che cerca le caratteristiche principali del viso come occhi, naso e bocca per rilevare i volti in un'immagine di input. Immagini Amazon Rekognition rileva le 100 facce più grandi di un'immagine.

Puoi fornire un'immagine di input come matrice di byte dell'immagine (byte dell'immagine codificata in formato Base64) o specificare un oggetto di Amazon S3. In questa procedura, viene caricata un'immagine (JPEG o PNG) nel bucket S3 e viene specificato il nome chiave dell'oggetto.

Per rilevare volti in un'immagine

1. Se non lo hai già fatto:

   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

   b. Installa e configura il e il. AWS CLI AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Carica un'immagine (contenente uno o più volti) nel bucket S3.

   Per le istruzioni, consulta Caricamento di oggetti in Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

3. Utilizza i seguenti esempi per richiamare `DetectFaces`.

   Java

   In questo esempio viene visualizzato l'intervallo di età stimato per i volti rilevati e sono elencati i JSON per tutti gli attributi facciali rilevati. Modifica il valore di `photo` nel nome del file immagine. Modifica il valore di `amzn-s3-demo-bucket` nel bucket Amazon S3 dove è archiviata l'immagine.

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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
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

Questo esempio visualizza l'output JSON dell'operazione. `detect-faces` AWS CLI Sostituisci `file` con il nome di un file immagine. Sostituisci `amzn-s3-demo-bucket` con il nome del bucket Amazon S3 che contiene il file immagine.

```
aws rekognition detect-faces --image "{"S3Object":{"Bucket":"amzn-s3-demo-
bucket,"Name":"image-name"}}"\
```

```
                              --attributes "ALL" --profile profile-name --region
 region-name
```

Se accedi alla CLI da un dispositivo Windows, usa le virgolette doppie anziché le virgolette singole ed evita le virgolette doppie interne tramite barra rovesciata (ovvero, \) per risolvere eventuali errori del parser che potresti riscontrare. Per un esempio, consulta quanto segue:

```
aws rekognition detect-faces --image "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --attributes "ALL"
--profile profile-name --region region-name
```

Python

In questo esempio viene visualizzato l'intervallo di età stimato e altri attributi per i volti rilevati e sono elencati i JSON per tutti gli attributi facciali rilevati. Modifica il valore di `photo` nel nome del file immagine. Modifica il valore di `amzn-s3-demo-bucket` nel bucket Amazon S3 dove è archiviata l'immagine. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
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 questo esempio viene visualizzato l'intervallo di età stimato per i volti rilevati e sono elencati i JSON per tutti gli attributi facciali rilevati. Modifica il valore di `photo` nel nome del file immagine. Modifica il valore di `amzn-s3-demo-bucket` nel bucket Amazon S3 dove è archiviata l'immagine.

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

Questo esempio mostra l'intervallo di età stimato per i volti rilevati ed elenca vari attributi facciali. Modifica il valore di `photo` nel nome del file immagine. Modifica il valore di `amzn-s3-demo-bucket` nel bucket Amazon S3 dove è archiviata l'immagine.

```
# 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

Questo esempio mostra l'intervallo di età stimato per i volti rilevati ed elenca vari attributi facciali. Modifica il valore di `photo` nel nome del file immagine. Modifica il valore di `amzn-s3-demo-bucket` nel bucket Amazon S3 dove è archiviata l'immagine.

Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

Se si utilizzano TypeScript definizioni, potrebbe essere necessario utilizzare `import AWS from 'aws-sdk'` invece di`const AWS = require('aws-sdk')`, per eseguire il programma con Node.js. Puoi consultare l'AWS SDK per Javascript per maggiori dettagli. A seconda di come sono state impostate le configurazioni, potrebbe essere necessario specificare anche la regione con `AWS.config.update({region:`*`region`*`});`.

```
// Load the SDK
var AWS = require('aws-sdk');
const bucket = 'bucket-name' // the bucketname without s3://
const photo  = 'photo-name' // the name of file

var credentials = new AWS.SharedIniFileCredentials({profile: 'profile-name'});
AWS.config.credentials = credentials;
AWS.config.update({region:'region-name'});

const client = new AWS.Rekognition();
const params = {
  Image: {
    S3Object: {
      Bucket: bucket,
      Name: photo
    },
  },
  Attributes: ['ALL']
}

client.detectFaces(params, function(err, response) {
    if (err) {
      console.log(err, err.stack); // an error occurred
    } else {
      console.log(`Detected faces for: ${photo}`)
      response.FaceDetails.forEach(data => {
        let low  = data.AgeRange.Low
        let high = data.AgeRange.High
        console.log(`The detected face is between: ${low} and ${high} years
 old`)
        console.log("All other attributes:")
        console.log(`  BoundingBox.Width:      ${data.BoundingBox.Width}`)
        console.log(`  BoundingBox.Height:     ${data.BoundingBox.Height}`)
        console.log(`  BoundingBox.Left:       ${data.BoundingBox.Left}`)
```

```
        console.log(`  BoundingBox.Top:         ${data.BoundingBox.Top}`)
        console.log(`  Age.Range.Low:          ${data.AgeRange.Low}`)
        console.log(`  Age.Range.High:         ${data.AgeRange.High}`)
        console.log(`  Smile.Value:            ${data.Smile.Value}`)
        console.log(`  Smile.Confidence:       ${data.Smile.Confidence}`)
        console.log(`  Eyeglasses.Value:       ${data.Eyeglasses.Value}`)
        console.log(`  Eyeglasses.Confidence:  ${data.Eyeglasses.Confidence}`)
        console.log(`  Sunglasses.Value:       ${data.Sunglasses.Value}`)
        console.log(`  Sunglasses.Confidence:  ${data.Sunglasses.Confidence}`)
        console.log(`  Gender.Value:           ${data.Gender.Value}`)
        console.log(`  Gender.Confidence:      ${data.Gender.Confidence}`)
        console.log(`  Beard.Value:            ${data.Beard.Value}`)
        console.log(`  Beard.Confidence:       ${data.Beard.Confidence}`)
        console.log(`  Mustache.Value:         ${data.Mustache.Value}`)
        console.log(`  Mustache.Confidence:    ${data.Mustache.Confidence}`)
        console.log(`  EyesOpen.Value:         ${data.EyesOpen.Value}`)
        console.log(`  EyesOpen.Confidence:    ${data.EyesOpen.Confidence}`)
        console.log(`  MouthOpen.Value:        ${data.MouthOpen.Value}`)
        console.log(`  MouthOpen.Confidence:   ${data.MouthOpen.Confidence}`)
        console.log(`  Emotions[0].Type:       ${data.Emotions[0].Type}`)
        console.log(`  Emotions[0].Confidence: ${data.Emotions[0].Confidence}`)
        console.log(`  Landmarks[0].Type:      ${data.Landmarks[0].Type}`)
        console.log(`  Landmarks[0].X:         ${data.Landmarks[0].X}`)
        console.log(`  Landmarks[0].Y:         ${data.Landmarks[0].Y}`)
        console.log(`  Pose.Roll:              ${data.Pose.Roll}`)
        console.log(`  Pose.Yaw:               ${data.Pose.Yaw}`)
        console.log(`  Pose.Pitch:             ${data.Pose.Pitch}`)
        console.log(`  Quality.Brightness:     ${data.Quality.Brightness}`)
        console.log(`  Quality.Sharpness:      ${data.Quality.Sharpness}`)
        console.log(`  Confidence:             ${data.Confidence}`)
        console.log("------------")
        console.log("")
      }) // for response.faceDetails
    } // if
  });
```

DetectFaces richiesta di operazione

L'input per `DetectFaces` è un'immagine. In questo esempio, l'immagine viene caricata da un bucket Amazon S3. Il parametro `Attributes` specifica che tutti gli attributi facciali devono essere restituiti. Per ulteriori informazioni, consulta Lavorare con le immagini.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "input.jpg"
        }
    },
    "Attributes": [
        "ALL"
    ]
}
```

DetectFaces risposta operativa

`DetectFaces` restituisce le informazioni riportate di seguito per ogni volto rilevato:

- Riquadro di delimitazione – Le coordinate del riquadro di delimitazione che circonda il volto.
- Affidabilità – Il livello di affidabilità che il riquadro di delimitazione contenga un volto.
- Punti di riferimento facciali – Matrice di punti di riferimento del volto. Per ogni punto di riferimento (come ad esempio occhio sinistro, occhio destro e bocca) la risposta fornisce le coordinate x e y.
- Attributi facciali – Un insieme di attributi facciali, ad esempio se il viso è occluso, restituito come un `FaceDetail` oggetto. Il set include: Beard AgeRange, Emotions EyeDirection, occhiali da vista,, genere EyesOpen FaceOccluded, baffi MouthOpen, sorriso e occhiali da sole. Per ogni attributo, la risposta fornisce un valore. Il valore può essere di diverso tipo, ad esempio un valore booleano (se una persona indossa occhiali da sole), una stringa (se la persona è maschio o femmina), o un valore angolare in gradi (per le direzioni di sguardo pitch/yaw). Inoltre, per la maggior parte degli attributi la risposta fornisce anche un livello di affidabilità nel valore rilevato per l'attributo. Tieni presente che, sebbene EyeDirection gli attributi FaceOccluded e siano supportati durante l'utilizzo`DetectFaces`, non lo sono quando si analizzano video con e. `StartFaceDetection` `GetFaceDetection`
- Qualità – Descrive la luminosità e la nitidezza del volto. Per informazioni su come garantire un rilevamento facciale ottimale, consulta Raccomandazioni per immagini di input di confronto di volti.
- Posa – Descrive la rotazione del volto all'interno dell'immagine.

La richiesta può descrivere una serie di attributi facciali che desideri vengano restituiti. Verrà sempre restituito un `DEFAULT` sottoinsieme di attributi facciali - `BoundingBox`, `Confidence`, `Pose`,

`Quality` e `Landmarks`. Puoi richiedere la restituzione di attributi facciali specifici (oltre all'elenco predefinito), utilizzando `["DEFAULT", "FACE_OCCLUDED", "EYE_DIRECTION"]` o solo un attributo, come`["FACE_OCCLUDED"]`. Puoi richiedere tutti gli attributi facciali utilizzando `["ALL"]`. La richiesta di più attributi può aumentare i tempi di risposta.

Di seguito è riportata una risposta di esempio per la chiamata API`DetectFaces`:


```
{
  "FaceDetails": [
    {
      "BoundingBox": {
        "Width": 0.7919622659683228,
        "Height": 0.7510867118835449,
        "Left": 0.08881539851427078,
        "Top": 0.151064932346344
      },
      "AgeRange": {
        "Low": 18,
        "High": 26
      },
      "Smile": {
        "Value": false,
        "Confidence": 89.77348327636719
      },
      "Eyeglasses": {
        "Value": true,
        "Confidence": 99.99996948242188
      },
      "Sunglasses": {
        "Value": true,
        "Confidence": 93.65237426757812
      },
      "Gender": {
        "Value": "Female",
        "Confidence": 99.85968780517578
      },
      "Beard": {
        "Value": false,
        "Confidence": 77.52591705322266
      },
      "Mustache": {
        "Value": false,
        "Confidence": 94.48904418945312
```

```
      },
      "EyesOpen": {
        "Value": true,
        "Confidence": 98.57169342041016
      },
      "MouthOpen": {
        "Value": false,
        "Confidence": 74.33953094482422
      },
      "Emotions": [
        {
          "Type": "SAD",
          "Confidence": 65.56403350830078
        },
        {
          "Type": "CONFUSED",
          "Confidence": 31.277774810791016
        },
        {
          "Type": "DISGUSTED",
          "Confidence": 15.553778648376465
        },
        {
          "Type": "ANGRY",
          "Confidence": 8.012762069702148
        },
        {
          "Type": "SURPRISED",
          "Confidence": 7.621500015258789
        },
        {
          "Type": "FEAR",
          "Confidence": 7.243380546569824
        },
        {
          "Type": "CALM",
          "Confidence": 5.8196024894714355
        },
        {
          "Type": "HAPPY",
          "Confidence": 2.2830512523651123
        }
      ],
      "Landmarks": [
```

```
        {
          "Type": "eyeLeft",
          "X": 0.30225440859794617,
          "Y": 0.41018882393836975
        },
        {
          "Type": "eyeRight",
          "X": 0.6439348459243774,
          "Y": 0.40341562032699585
        },
        {
          "Type": "mouthLeft",
          "X": 0.343580037355423,
          "Y": 0.6951127648353577
        },
        {
          "Type": "mouthRight",
          "X": 0.6306480765342712,
          "Y": 0.6898072361946106
        },
        {
          "Type": "nose",
          "X": 0.47164231538772583,
          "Y": 0.5763645172119141
        },
        {
          "Type": "leftEyeBrowLeft",
          "X": 0.1732882857322693,
          "Y": 0.34452149271965027
        },
        {
          "Type": "leftEyeBrowRight",
          "X": 0.3655243515968323,
          "Y": 0.33231860399246216
        },
        {
          "Type": "leftEyeBrowUp",
          "X": 0.2671719491481781,
          "Y": 0.31669262051582336
        },
        {
          "Type": "rightEyeBrowLeft",
          "X": 0.5613729953765869,
          "Y": 0.32813435792922974
```

```
        },
        {
          "Type": "rightEyeBrowRight",
          "X": 0.7665090560913086,
          "Y": 0.3318614959716797
        },
        {
          "Type": "rightEyeBrowUp",
          "X": 0.6612788438796997,
          "Y": 0.3082450032234192
        },
        {
          "Type": "leftEyeLeft",
          "X": 0.2416982799768448,
          "Y": 0.4085965156555176
        },
        {
          "Type": "leftEyeRight",
          "X": 0.36943578720092773,
          "Y": 0.41230902075767517
        },
        {
          "Type": "leftEyeUp",
          "X": 0.29974061250686646,
          "Y": 0.3971870541572571
        },
        {
          "Type": "leftEyeDown",
          "X": 0.30360740423202515,
          "Y": 0.42347756028175354
        },
        {
          "Type": "rightEyeLeft",
          "X": 0.5755768418312073,
          "Y": 0.4081145226955414
        },
        {
          "Type": "rightEyeRight",
          "X": 0.7050536870956421,
          "Y": 0.39924031496047974
        },
        {
          "Type": "rightEyeUp",
          "X": 0.642906129360199,
```

```
          "Y": 0.39026668667793274
        },
        {
          "Type": "rightEyeDown",
          "X": 0.6423097848892212,
          "Y": 0.41669243574142456
        },
        {
          "Type": "noseLeft",
          "X": 0.4122826159000397,
          "Y": 0.5987403392791748
        },
        {
          "Type": "noseRight",
          "X": 0.5394935011863708,
          "Y": 0.5960900187492371
        },
        {
          "Type": "mouthUp",
          "X": 0.478581964969635,
          "Y": 0.6660456657409668
        },
        {
          "Type": "mouthDown",
          "X": 0.483366996049881,
          "Y": 0.7497162818908691
        },
        {
          "Type": "leftPupil",
          "X": 0.30225440859794617,
          "Y": 0.41018882393836975
        },
        {
          "Type": "rightPupil",
          "X": 0.6439348459243774,
          "Y": 0.40341562032699585
        },
        {
          "Type": "upperJawlineLeft",
          "X": 0.11031254380941391,
          "Y": 0.3980775475502014
        },
        {
          "Type": "midJawlineLeft",
```

```
          "X": 0.19301874935626984,
          "Y": 0.7034031748771667
        },
        {
          "Type": "chinBottom",
          "X": 0.4939905107021332,
          "Y": 0.8877836465835571
        },
        {
          "Type": "midJawlineRight",
          "X": 0.7990140914916992,
          "Y": 0.6899225115776062
        },
        {
          "Type": "upperJawlineRight",
          "X": 0.8548634648323059,
          "Y": 0.38160091638565063
        }
      ],
      "Pose": {
        "Roll": -5.83309268951416,
        "Yaw": -2.4244730472564697,
        "Pitch": 2.6216139793395996
      },
      "Quality": {
        "Brightness": 96.16363525390625,
        "Sharpness": 95.51618957519531
      },
      "Confidence": 99.99872589111328,
      "FaceOccluded": {
        "Value": true,
        "Confidence": 99.99726104736328
      },
      "EyeDirection": {
        "Yaw": 16.299732,
        "Pitch": -6.407457,
        "Confidence": 99.968704
      }
    }
  ],
  "ResponseMetadata": {
    "RequestId": "8bf02607-70b7-4f20-be55-473fe1bba9a2",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
```

```
      "x-amzn-requestid": "8bf02607-70b7-4f20-be55-473fe1bba9a2",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "3409",
      "date": "Wed, 26 Apr 2023 20:18:50 GMT"
    },
    "RetryAttempts": 0
  }
}
```

Tieni presente quanto segue:

- I dati `Pose` descrivono la rotazione del volto rilevato. È possibile utilizzare la combinazione dei dati `BoundingBox` e `Pose` per tracciare il riquadro di delimitazione intorno ai volti visualizzati nell'applicazione.
- L'operazione `Quality` descrive la luminosità e la nitidezza del volto. Può risultare utile per confrontare i volti delle varie immagini e trovare quello migliore.
- La precedente risposta mostra tutti i `landmarks` facciali che il servizio è in grado di rilevare, tutti gli attributi facciali e le emozioni. Affinché la risposta comprenda tutti questi elementi, occorre specificare il parametro `attributes` con il valore `ALL`. Per impostazione predefinita, l'API `DetectFaces` restituisce solo questi cinque attributi: `BoundingBox`, `Confidence`, `Pose`, `Quality` e `landmarks`. I punti di riferimento predefiniti restituiti sono: `eyeLeft`, `eyeRightnose`, `mouthLeft` e `mouthRight`.

Confronto dei volti nelle immagini

Con Rekognition puoi confrontare i volti tra due immagini usando l'operazione. CompareFaces Questa funzione è utile per applicazioni come la verifica dell'identità o la corrispondenza delle foto.

CompareFaces confronta un volto nell'immagine sorgente con ogni volto nell'immagine di destinazione. Le immagini vengono passate CompareFaces a:

- Una rappresentazione codificata in base64 di un'immagine.
- Oggetti Amazon S3.

Riconoscimento facciale e confronto facciale

Il confronto dei volti è diverso dal rilevamento dei volti. Il rilevamento dei volti (che utilizza DetectFaces) identifica solo la presenza e la posizione dei volti in un'immagine o in un video. Al contrario, il confronto dei volti consiste nel confrontare un volto rilevato in un'immagine sorgente con i volti in un'immagine di destinazione per trovare le corrispondenze.

Soglie di somiglianza

Utilizzate il `similarityThreshold` parametro per definire il livello minimo di confidenza per le corrispondenze da includere nella risposta. Per impostazione predefinita, nella risposta vengono restituiti solo i volti con un punteggio di somiglianza maggiore o uguale all'80%.

> **Note**
>
> `CompareFaces`utilizza algoritmi di apprendimento automatico, che sono probabilistici. Un falso negativo è una previsione errata secondo cui un volto nell'immagine di destinazione ha un punteggio di confidenza di somiglianza basso rispetto al volto nell'immagine sorgente. Per ridurre la probabilità di falsi negativi, si consiglia di confrontare l'immagine di destinazione con più immagini di origine. Se prevedi di `CompareFaces` prendere una decisione che influisca sui diritti, sulla privacy o sull'accesso ai servizi di una persona, ti consigliamo di trasmettere il risultato a un essere umano per la revisione e l'ulteriore convalida prima di agire.

I seguenti esempi di codice mostrano come utilizzare le CompareFaces operazioni per vari. AWS SDKs Nell' AWS CLI esempio, carichi due immagini JPEG nel tuo bucket Amazon S3 e specifichi il nome della chiave dell'oggetto. Negli altri esempi, vengono caricati due file dal file system locale e vengono inseriti come matrici di byte dell'immagine.

Per confrontare i volti

1. Se non lo hai già fatto:
   a. Crea o aggiorna un utente con autorizzazioni `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess` (solo a titolo di AWS CLI esempio). Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Utilizza il seguente codice di esempio per richiamare l'operazione `CompareFaces`.

Java

Questo esempio visualizza le informazioni sulla corrispondenza dei volti nelle immagini di origine e di destinazione che vengono caricati dal file system locale.

Sostituisci i valori di `sourceImage` e `targetImage` con il percorso e il nome di file delle immagini di origine e destinazione.

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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
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

Questo esempio visualizza l'output JSON dell'operazione. `compare-faces` AWS CLI

Sostituisci `amzn-s3-demo-bucket` con il nome del bucket Amazon S3 che contiene le immagini di origine e destinazione. Sostituisci `source.jpg` e `target.jpg` con i nomi di file per le immagini di origine e di destinazione.

```
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
```

Se accedi alla CLI da un dispositivo Windows, usa le virgolette doppie anziché le virgolette singole ed evita le virgolette doppie interne tramite barra rovesciata (ovvero, \) per risolvere eventuali errori del parser che potresti riscontrare. Per un esempio, consulta quanto segue:

```
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

Questo esempio visualizza le informazioni sulla corrispondenza dei volti nelle immagini di origine e di destinazione che vengono caricati dal file system locale.

Sostituisci i valori di `source_file` e `target_file` con il percorso e il nome di file delle immagini di origine e destinazione. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

Questo esempio visualizza le informazioni sulla corrispondenza dei volti nelle immagini di origine e di destinazione che vengono caricati dal file system locale.

Sostituisci i valori di `sourceImage` e `targetImage` con il percorso e il nome di file delle immagini di origine e destinazione.

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

Questo esempio visualizza le informazioni sulla corrispondenza dei volti nelle immagini di origine e di destinazione che vengono caricati dal file system locale.

Sostituisci i valori di `photo_source` e `photo_target` con il percorso e il nome di file delle immagini di origine e destinazione.

```
 # 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

Questo esempio visualizza le informazioni sulla corrispondenza dei volti nelle immagini di origine e di destinazione che vengono caricati dal file system locale.

Sostituisci i valori di `photo_source` e `photo_target` con il percorso e il nome di file delle immagini di origine e destinazione. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
// 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 richiesta di operazione

L'input per `CompareFaces` è un'immagine. In questo esempio, le immagini di origine e di destinazione vengono caricate dal file system locale. Il parametro di input `SimilarityThreshold` specifica l'affidabilità minima che i volti confrontati devono corrispondere per essere inclusi nella risposta. Per ulteriori informazioni, consulta Lavorare con le immagini.

```
{
    "SourceImage": {
        "Bytes": "/9j/4AAQSk2Q==..."
    },
    "TargetImage": {
        "Bytes": "/9j/4O1Q==..."
    },
    "SimilarityThreshold": 70
}
```

CompareFaces risposta operativa

La risposta include:

- Una serie di corrispondenze facciali: un elenco di volti abbinati con punteggi di somiglianza e metadati per ogni faccia corrispondente. Se più facce coincidono, `faceMatches`

  l'array include tutte le corrispondenze facciali.
- Dettagli della corrispondenza facciale: ogni faccia abbinata fornisce anche un riquadro di delimitazione, un valore di confidenza, punti di riferimento e un punteggio di somiglianza.
- Un elenco di volti non corrispondenti: la risposta include anche volti dell'immagine di destinazione che non corrispondono al volto dell'immagine di origine. Include un riquadro di delimitazione per ogni faccia non corrispondente.
- Informazioni sul volto di origine: include informazioni sul volto tratte dall'immagine sorgente utilizzata per il confronto, inclusi il riquadro di delimitazione e il valore di confidenza.

L'esempio mostra che è stata trovata una corrispondenza facciale nell'immagine di destinazione. Per la corrispondenza del volto, fornisce un riquadro di delimitazione e un valore di affidabilità (il livello di affidabilità che Amazon Rekognition ha nel fatto che il riquadro di delimitazione contenga un volto). Il punteggio di somiglianza di 99,99 indica la somiglianza dei volti. L'esempio mostra anche un volto che Amazon Rekognition ha trovato nell'immagine di destinazione che non corrisponde al volto analizzato nell'immagine di origine.


```
{
    "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
    }
}
```


Rilevamento di volti in un video archiviato

Video Amazon Rekognition è in grado di rilevare volti nei video archiviati in un bucket Amazon S3 e di fornire informazioni quali:

- Quante volte i volti vengono rilevati in un video.
- Posizione dei volti nel fotogramma video al momento del rilevamento.
- Punti di riferimento facciali, ad esempio la posizione dell'occhio sinistro.
- Attributi aggiuntivi come spiegato nella pagina the section called “Linee guida sugli attributi facciali”.

Il rilevamento dei volti di Video Amazon Rekognition nei video archiviati è un'operazione asincrona. Per iniziare a rilevare i volti nei video, chiama. StartFaceDetection Amazon Rekognition per video pubblica lo stato di completamento dell'analisi video in un argomento Amazon Simple Notification Service (Amazon SNS). Se l'analisi video ha esito positivo, puoi effettuare nuovamente la chiamata

GetFaceDetection per ottenere i risultati dell'analisi video. Per ulteriori informazioni su come avviare analisi video e ottenere i risultati, consultare Chiamata delle operazioni Video Amazon Rekognition.

La procedura si espande nel codice in Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK), che utilizza una coda Amazon Simple Queue Service (Amazon SQS) per ottenere lo stato di completamento di una richiesta di analisi video.

Per rilevare volti in un video archiviato in un bucket Amazon S3 (SDK)

1. Eseguire Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK).
2. Aggiungere il seguente codice alla classe `VideoDetect` creata nella fase 1.

   AWS CLI

   - Nel seguente esempio di codice, cambia `amzn-s3-demo-bucket` e `video-name` con il nome del bucket Amazon S3 e il nome del file specificati nella fase 2.
   - Cambia `region-name` con la regione AWS che stai utilizzando. Sostituisci il valore di `profile_name` con il nome del tuo profilo di sviluppatore.
   - Cambia `TopicARN` con l'ARN dell'argomento Amazon SNS creato nella fase 3 di Configurazione di Video Amazon Rekognition.
   - Modifica `RoleARN` con l'ARN del ruolo di servizio IAM creato nella fase 7 di Configurazione di Video Amazon Rekognition.

   ```
   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
   ```

   Se accedi alla CLI da un dispositivo Windows, usa le virgolette doppie anziché le virgolette singole ed evita le virgolette doppie interne tramite barra rovesciata (ovvero, \) per risolvere eventuali errori del parser che potresti riscontrare. Per un esempio, consulta quanto segue:

   ```
   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
```

Dopo aver eseguito l'`StartFaceDetection`operazione e ottenuto il numero ID del lavoro, esegui l'`GetFaceDetection`operazione seguente e fornisci il numero ID del lavoro:

```
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);


}
```

Nella funzione `main`, sostituisci le righe:

```
        StartLabelDetection(amzn-s3-demo-bucket, video);
```

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

con:

```
        StartFaceDetection(amzn-s3-demo-bucket, video);

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

Java V2

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
//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
```

Nella funzione `main`, sostituisci le righe:

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

con:

```
    analyzer.StartFaceDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetFaceDetectionResults()
```

> **Note**
>
> Se hai già eseguito un video di esempio diverso da Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK), il nome della funzione da sostituire è diverso.

3. Eseguire il codice. Vengono visualizzate le informazioni sui volti rilevati nel video.

GetFaceDetection risposta operativa

`GetFaceDetection` restituisce una matrice (`Faces`) che contiene informazioni sui volti rilevati nel video. Esiste un elemento della matrice, FaceDetection, per ogni rilevamento di un volto nel video. Gli elementi della matrice restituiti sono ordinati in base al tempo, espresso in millisecondi, dall'inizio del video.

Di seguito è riportato un esempio di risposta JSON parziale dell'operazione `GetFaceDetection`. Nella risposta, tenere presente quanto segue:

- Riquadro di delimitazione – Le coordinate del riquadro di delimitazione che circonda il volto.
- Affidabilità – Il livello di affidabilità che il riquadro di delimitazione contenga un volto.

- Punti di riferimento facciali – Matrice di punti di riferimento del volto. Per ogni punto di riferimento (come ad esempio occhio sinistro, occhio destro e bocca) la risposta fornisce le coordinate `x` e `y`.
- Attributi del viso: un insieme di attributi facciali, che include: AgeRange barba, emozioni, occhiali da vista, genere EyesOpen, baffi MouthOpen, sorriso e occhiali da sole. Il valore può essere di diverso tipo, ad esempio un valore booleano (se una persona indossa occhiali da sole) o una stringa (se la persona è maschio o femmina). Inoltre, per la maggior parte degli attributi la risposta fornisce anche un livello di affidabilità nel valore rilevato per l'attributo. Tieni presente che, sebbene EyeDirection gli attributi FaceOccluded e siano supportati durante l'utilizzo`DetectFaces`, non lo sono quando si analizzano video con e. `StartFaceDetection` `GetFaceDetection`
- Timestamp – L'ora in cui il volto è stato rilevato nel video.
- Informazioni di paginazione – L'esempio illustra una pagina di informazioni di rilevamento del volto. Puoi specificare il numero di elementi della persona da restituire nel parametro di input `MaxResults` per `GetFaceDetection`. Se esiste un numero di risultati maggiore di `MaxResults`, `GetFaceDetection` restituisce un token (`NextToken`) che viene utilizzato per ottenere la pagina di risultati successiva. Per ulteriori informazioni, consulta Ottenere i risultati dell'analisi di Video Amazon Rekognition.
- Informazioni video - La risposta include informazioni sul formato video (`VideoMetadata`) in ogni pagina di informazioni restituita da `GetFaceDetection`.
- Qualità – Descrive la luminosità e la nitidezza del volto.
- Posa – Descrive la rotazione del volto.


```
{
    "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
    }
}
```

Ricerca di volti in una raccolta

Amazon Rekognition ti consente di utilizzare un volto di input per cercare corrispondenze in una raccolta di volti archiviati. Inizia archiviando le informazioni sui volti rilevati in container lato server chiamati "raccolte". Le raccolte memorizzano sia i singoli volti che gli utenti (più volti della stessa persona). I singoli volti vengono archiviati come vettori di volti, una rappresentazione matematica del volto (non un'immagine reale del volto). È possibile utilizzare immagini diverse della stessa persona per creare e archiviare più immagini vettoriali di volti nella stessa raccolta. È quindi possibile aggregare più vettori di volti della stessa persona per creare un vettore utente. I vettori utente possono offrire una maggiore precisione nella ricerca dei volti con rappresentazioni più robuste, contenenti diversi gradi di illuminazione, nitidezza, posa, aspetto, ecc.

Dopo aver creato una raccolta, è possibile utilizzare un volto di input per cercare i vettori utente o i vettori di volti corrispondenti in una raccolta. La ricerca tra i vettori utente può migliorare significativamente la precisione rispetto alla ricerca su singoli vettori di volti. È possibile utilizzare i volti rilevati nelle immagini, nei video archiviati e nei video in streaming per cercare tra i vettori di volti archiviati. È possibile utilizzare i volti rilevati nelle immagini per cercare tra i vettori utente archiviati.

Per archiviare le informazioni sui volti, è necessario effettuare le seguenti operazioni:

1. Crea una raccolta: per archiviare le informazioni sul viso, devi prima creare (CreateCollection) una raccolta di volti in una delle AWS regioni del tuo account. La raccolta di volti viene specificata chiamando l'operazione `IndexFaces`.
2. Index Faces: l'IndexFacesoperazione rileva i volti in un'immagine, estrae e memorizza i vettori di volti nella raccolta. È possibile usare questa operazione per rilevare i volti in un'immagine e mantenere le informazioni sulle caratteristiche dei volti rilevate in una raccolta. Di seguito è riportato un esempio di un'operazione API basata su storage, poiché il servizio archivia le informazioni dei vettori di volti nel server.

Per creare un utente e associare più vettori di volti a un utente, è necessario effettuare le seguenti operazioni:

1. Crea un utente: devi prima creare un utente con. CreateUser È possibile migliorare la precisione della corrispondenza dei volti aggregando più vettori di volti della stessa persona in un vettore utente. È possibile associare fino a 100 vettori di volti a un vettore utente.

2. Associa volti: dopo aver creato l'utente, puoi aggiungere vettori di volti esistenti a quell'utente con l'AssociateFacesoperazione. I vettori di volti devono risiedere nella stessa raccolta di un vettore utente per essere associati a quel vettore utente.

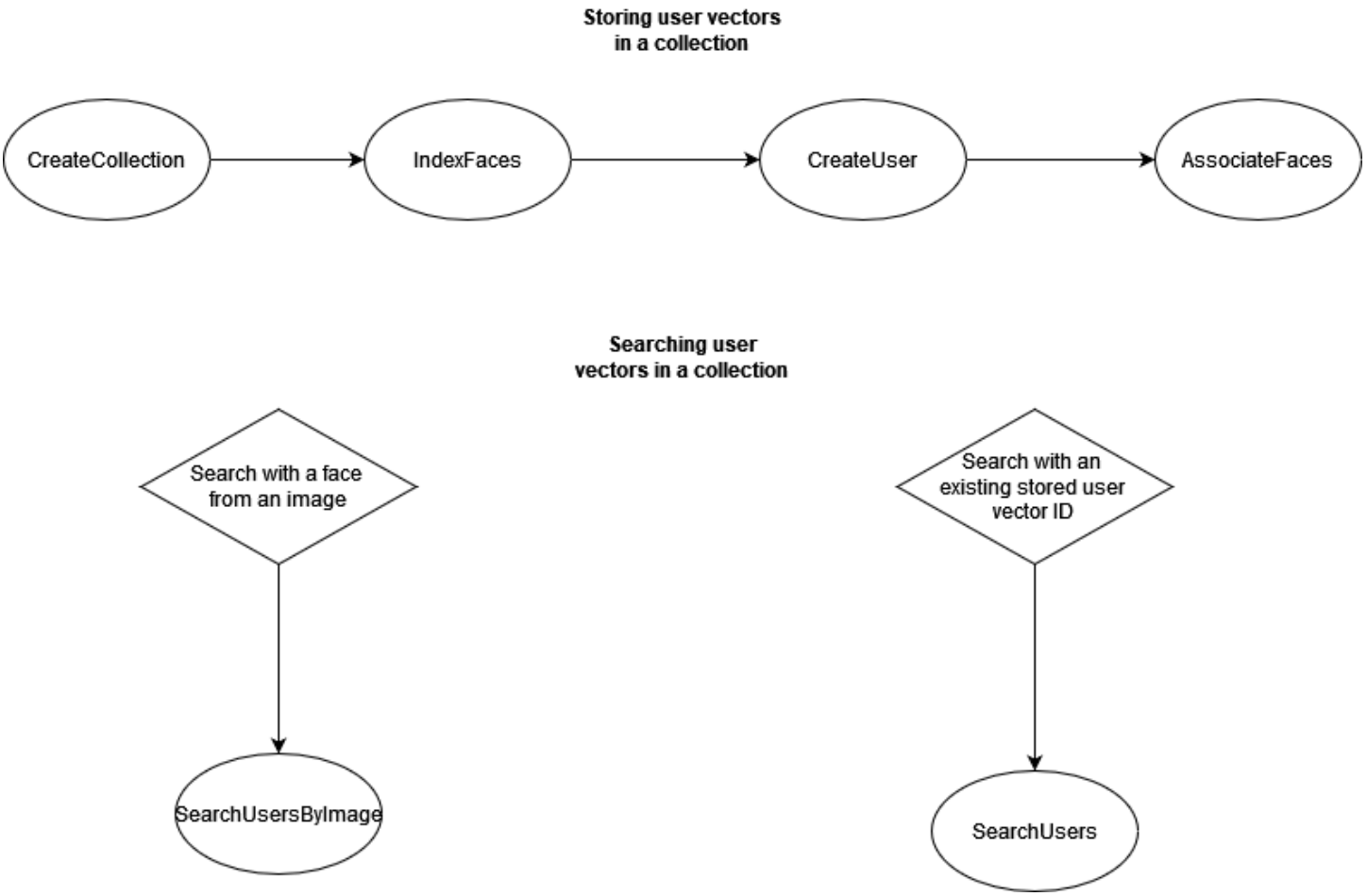
Dopo aver creato una raccolta e archiviato i vettori di volti e utenti, è possibile utilizzare le seguenti operazioni per cercare le corrispondenze dei volti:

- SearchFacesByImage- Per cercare singoli volti memorizzati con un volto di un'immagine.
- SearchFaces- Per cercare tra i singoli volti memorizzati con un face ID fornito.
- SearchUsers- Per cercare tra gli utenti memorizzati con un face ID o un ID utente forniti.
- SearchUsersByImage- Per eseguire una ricerca tra gli utenti archiviati con un volto tratto da un'immagine.
- StartFaceSearch- Per cercare volti in un video memorizzato.
- CreateStreamProcessor- Per cercare volti in un video in streaming.

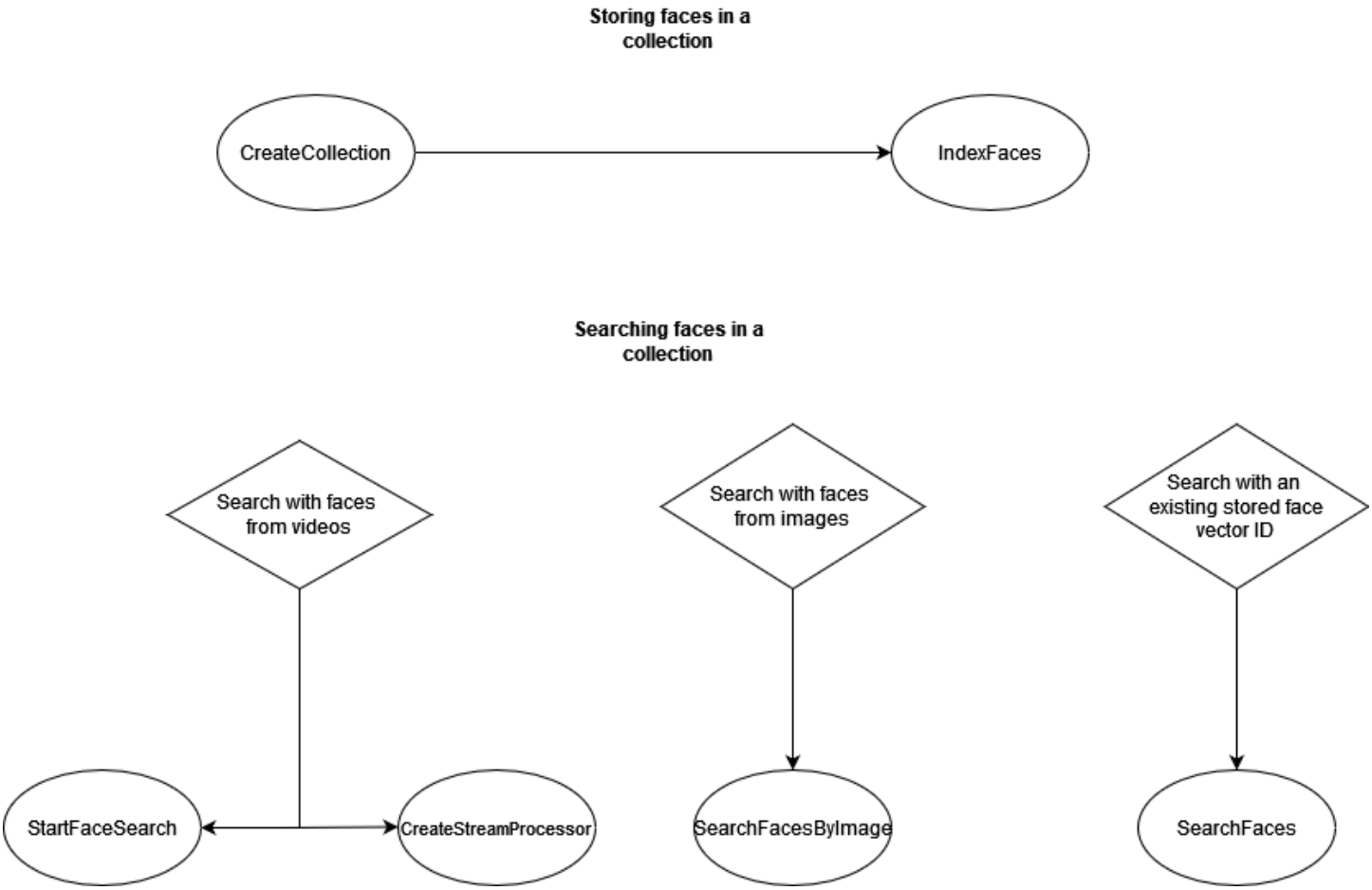
> **Note**
>
> Le raccolte memorizzano i vettori dei volti, che sono rappresentazioni matematiche dei volti. Le raccolte non memorizzano immagini di volti.

I seguenti diagrammi mostrano l'ordine delle operazioni di chiamata, in base agli obiettivi di utilizzo delle raccolte:

Per la massima precisione di abbinamento con i vettori utente:



Per una corrispondenza ad alta precisione con i singoli vettori facciali:



È possibile utilizzare le raccolte in diversi scenari. Ad esempio, è possibile creare una raccolta di volti che memorizzi i volti rilevati dalle immagini scansionate dei badge dei dipendenti e IDs dalle autorità emesse utilizzando le operazioni and. `IndexFaces` `AssociateFaces` Quando un dipendente entra nell'edificio, un'immagine del suo volto viene acquisita e inviata all'operazione `SearchUsersByImage`. Se la corrispondenza del volto produce un punteggio di somiglianza sufficientemente alto (pari al 99%), è possibile autenticare il collaboratore.

Gestione delle raccolte di volti, dei volti e degli utenti

La raccolta di volti è la risorsa utilizzata da Amazon Rekognition per contenere informazioni relative a volti e utenti. Utilizzi diverse operazioni dell'API Rekognition per gestire la raccolta stessa, nonché tutti i volti o i vettori di volti memorizzati nella raccolta.

Gestire una raccolta

La raccolta di volti è la risorsa principale di Amazon Rekognition e ogni raccolta creata dispone di un nome della risorsa Amazon (ARN) univoco. Puoi creare ogni raccolta di volti in una AWS regione

specifica del tuo account. Quando si crea una nuova raccolta, viene associata con la versione più recente del modello di rilevamento facciale. Per ulteriori informazioni, consulta Comprendere il controllo delle versioni dei modelli.

È possibile eseguire le seguenti operazioni di gestione su una raccolta:

- Creare una raccolta con CreateCollection. Per ulteriori informazioni, consulta Creazione di una raccolta.
- Elencare le raccolte disponibili con ListCollections. Per ulteriori informazioni, consulta Creazione dell'elenco delle raccolte.
- Descrivere una raccolta con DescribeCollection. Per ulteriori informazioni, consulta Descrizione di una raccolta.
- Eliminare una raccolta con DeleteCollection. Per ulteriori informazioni, consulta Eliminazione di una raccolta.

Gestione dei volti in una raccolta

Dopo aver creato una raccolta, è possibile archiviarvi volti. Amazon Rekognition fornisce le seguenti operazioni per gestire i volti in una raccolta:

- L'operazione IndexFaces rileva i volti nell'immagine di input (JPEG o PNG) e li aggiunge alla raccolta specificata. Per ogni volto rilevato nell'immagine viene restituito un ID volto univoco. Dopo aver reso persistenti i volti, sarà possibile cercare le corrispondenze nella raccolta. Per ulteriori informazioni, consulta Aggiunta di volti a una raccolta.
- L'operazione ListFaces elenca i volti in una raccolta. Per ulteriori informazioni, consulta Aggiunta di volti a una raccolta.
- L'operazione DeleteFaces elimina i volti da una raccolta. Per ulteriori informazioni, consulta Eliminazione dei volti da una raccolta.

Gestione degli utenti in una raccolta

Dopo aver archiviato più vettori di volti della stessa persona, è possibile migliorare la precisione associando tutti i vettori di volti in un unico vettore utente. Per gestire gli utenti, puoi utilizzare le operazioni elencate di seguito:

- CreateUser- L'operazione crea un nuovo utente in una raccolta con un ID utente univoco fornito.

- AssociateUsers- Aggiungi da 1 a 100 volti unici IDs a un ID utente. Dopo aver associato almeno un ID volto a un utente, puoi cercare le corrispondenze con quell'utente nella tua raccolta.
- ListUsers- Elenca gli utenti in una raccolta.
- DeleteUsers- Elimina un utente da una raccolta con l'ID utente fornito.
- DisassociateFaces- Rimuove uno o più volti IDs da un utente.

Utilizzo di soglie di somiglianza per associare e abbinare volti

Le soglie di somiglianza vengono utilizzate sia per associare che per abbinare i volti. Di seguito sono riportate le linee guida per l'utilizzo delle soglie di somiglianza per entrambi i casi d'uso.

Utilizzo delle soglie di somiglianza per l'associazione dei volti

Quando si associano volti utilizzando l'AssociateFacesoperazione, è importante assicurarsi che i volti associati a un utente appartengano tutti alla stessa persona. A titolo di aiuto, il parametro `UserMatchThreshold` specifica la confidenza minima di corrispondenza dell'utente richiesta per associare il nuovo volto a un `UserID` che contiene già almeno un `FaceID`. Questo aiuta a garantire che i `FaceIds` siano associati allo `UserID` corretto. Il valore è compreso tra 0 e 100 e il valore predefinito è 75.

Utilizzo di soglie di somiglianza per la corrispondenza dei volti

Ti consentiamo di controllare i risultati di tutte le operazioni di ricerca (CompareFaces,, SearchFaces, SearchFacesByImageSearchUsers, SearchUsersByImage) fornendo una soglia di somiglianza come parametro di input.

`FaceMatchThreshold` è l'attributo di input della soglia di somiglianza per `SearchFaces` e `SearchFacesByImage`, e consente di controllare il numero di risultati restituiti in base alla somiglianza al volto corrispondente. L'attributo della soglia di somiglianza per `SearchUsers` e `SearchUsersByImage`, `UserMatchThreshold`, consente di controllare il numero di risultati restituiti in base alla somiglianza al vettore utente corrispondente. L'attributo della soglia è `SimilarityThreshold` per `CompareFaces`.

Risposte con un valore di attributo di risposta `Similarity` inferiore alla soglia non vengono restituite. È importante calibrare questa soglia per il tuo caso d'uso, poiché può determinare il numero di falsi positivi inclusi nei risultati corrispondenti. Questo controlla il richiamo dei tuoi risultati di ricerca; minore sarà la soglia, maggiore sarà il richiamo.

Tutti i sistemi di Machine Learning sono probabilistici. Dovresti impostare autonomamente la giusta soglia di somiglianza, a seconda del tuo caso d'uso. Ad esempio, se stai cercando di creare un'applicazione di foto per identificare membri della famiglia somiglianti, potresti scegliere una soglia inferiore (ad esempio l'80%). Viceversa, per molti casi d'uso delle forze dell'ordine, si consiglia di impiegare un valore soglia elevato pari o superiore al 99%, al fine di ridurre errori di identificazione accidentali.

Oltre a `FaceMatchThreshold` e `UserMatchThreshold`, puoi utilizzare l'attributo di risposta `Similarity` per ridurre errori di identificazione accidentali. Ad esempio, puoi scegliere di utilizzare una soglia bassa (ad esempio l'80%) per restituire più risultati. Quindi, puoi utilizzare l'attributo di risposta Somiglianza (percentuale di somiglianza) per limitare la scelta e filtrare le risposte corrette nell'applicazione. Anche qui, maggiore è la somiglianza (ad esempio pari e superiore al 99%), minori saranno gli errori di identificazione.

Linee guida per l'indicizzazione dei volti in scenari comuni

Di seguito vengono fornite alcune indicazioni per l'utilizzo di `IndexFaces` in situazioni comuni.

Applicazioni critiche o di pubblica sicurezza

- Richiama IndexFaces con immagini che contengono un solo volto in ciascuna immagine e associa l'ID volto restituito con l'identificatore per l'oggetto dell'immagine.
- Puoi utilizzare DetectFaces prima dell'indicizzazione per verificare che l'immagine contenga effettivamente un solo volto. Se vengono rilevati più volti, invia di nuovo l'immagine dopo l'analisi e con un solo volto presente. In questo modo eviterai di indicizzare inavvertitamente più volti, associandoli alla stessa persona.

Condivisione di foto e applicazioni social media

- Se `IndexFaces` viene utilizzato, ad esempio, per creare un album di foto di famiglia, puoi richiamarlo senza limitare le immagini che contengono più volti. In questi casi, dovrai identificare ogni persona presente nelle foto e utilizzare tali informazioni per raggruppare le foto in base alle persone raffigurate.

Utilizzo generico

- Indicizza più immagini diverse della stessa persona, cercando di cogliere vari attributi facciali (espressioni, baffi, barba e così via), crea un utente e associa i diversi volti all'utente per migliorare la qualità della corrispondenza.
- Includi un processo di revisione in modo che gli errori di corrispondenza possano essere indicizzati con il corretto identificatore, per migliorare le capacità di trovare corrispondenze in futuro.
- Per informazioni sulla qualità delle immagini, consulta Raccomandazioni per immagini di input di confronto di volti.

Ricerca di volti e utenti all'interno di una raccolta

Dopo aver creato una raccolta di volti e aver archiviato vettori di volti e/o vettori utente, sarà possibile cercare le corrispondenze di volti al suo interno. Con Amazon Rekognition è possibile ricercare in una raccolta i volti che corrispondono a:

- Un ID volto fornito (SearchFaces). Per ulteriori informazioni, consulta Ricerca di un volto con un ID volto.
- Il volto più grande in un'immagine fornita (SearchFacesByImage). Per ulteriori informazioni, consulta Ricerca di un volto tramite immagine.
- I volti in un video archiviato. Per ulteriori informazioni, consulta Ricerca di volti in video archiviati.
- I volti in un video in streaming. Per ulteriori informazioni, consulta Utilizzo degli eventi video in streaming.

È possibile utilizzare l'operazione `CompareFaces` per confrontare un volto in un'immagine di origine con i volti nell'immagine di destinazione. L'ambito di questo confronto è limitato ai volti che vengono rilevati nell'immagine di destinazione. Per ulteriori informazioni, consulta Confronto dei volti nelle immagini.

Le varie operazioni di ricerca visualizzate nell'elenco seguente confrontano un volto (identificato da un `FaceId` o da un'immagine di input) con tutti i volti archiviati in una determinata raccolta di volti:

- SearchFaces
- SearchFacesByImage
- SearchUsers

- SearchUsersByImage

Casi d'uso che riguardano la sicurezza pubblica

Oltre alle raccomandazioni elencate in Best practice per sensori, immagini di input e video e Linee guida per l'indicizzazione dei volti in scenari comuni, dovresti adottare le seguenti best practice per la distribuzione dei sistemi di rilevamento e confronto dei volti per i casi d'uso che interessano la sicurezza pubblica. Innanzitutto, devi utilizzare soglie di confidenza del 99% o superiori in modo da ridurre errori e falsi positivi. In secondo luogo, è necessario prevedere l'intervento di revisori umani per verificare i risultati ricevuti dl sistema di confronto o di rilevamento. Non è consigliabile prendere decisioni basate sui risultati forniti dal sistema, senza inserire un elemento di controllo umano. I sistemi di rilevamento e confronto facciale devono costituire uno strumento utile per restringere il campo di ricerca e consentire alle persone di eseguire valutazioni rapide e prendere in considerazione tutte le opzioni disponibili. Infine, in questi casi è consigliabile adottare un utilizzo trasparente dei sistemi di rilevamento e confronto facciale, informando, quando possibile, gli utenti finali e i soggetti sull'uso di questi sistemi, ottenendo il loro consenso e offrendo loro un metodo per fornire opinioni e feedback su come migliorare il sistema.

Se sei un'agenzia di polizia che utilizza la funzionalità di confronto facciale Amazon Rekognition in relazione a indagini penali, devi rispettare i requisiti elencati nei Termini di servizio AWS. Questo include quanto segue:

- Gli esseri umani adeguatamente addestrati riesaminano tutte le decisioni di intraprendere azioni che potrebbero avere un impatto sulle libertà civili di una persona o sui diritti umani equivalenti.
- Addestrare il personale su un uso responsabile dei sistemi di riconoscimento facciale.
- Fornire informazioni pubbliche sull'uso dei sistemi di riconoscimento facciale da parte dell'utente.
- Non utilizzare Amazon Rekognition per la sorveglianza prolungata di una persona senza revisione indipendente o circostanze particolari.

In ogni caso, le corrispondenze per il confronto facciale devono essere considerate nel contesto di altre prove determinanti e non devono essere utilizzate come unica misura. Tuttavia, se il confronto facciale viene utilizzato per non-law-enforcement scenari (ad esempio, per sbloccare un telefono o autenticare l'identità di un dipendente per accedere a un edificio privato e sicuro), queste decisioni non richiederebbero un controllo manuale perché non influirebbero sulle libertà civili o sui diritti umani equivalenti di una persona.

Quando pianifichi l'utilizzo di un sistema di rilevamento o di confronto facciale per casi che interessano la sicurezza pubblica cerca di adottare sempre le best practice menzionate in precedenza. Inoltre, dovresti consultare le risorse pubblicate sull'uso del confronto facciale. Un documento utile è, ad esempio il Face Recognition Policy Development Template For Use In Criminal Intelligence and Investigative Activities pubblicato dal Bureau of Justice Assistance del Dipartimento di giustizia degli Stati Uniti. Il modello mette a disposizione una serie di risorse per il confronto facciale e relative alla biometrica ed è stato sviluppato per fornire alle forze di polizia e di pubblica sicurezza una struttura per definire policy di confronto facciale che risultino conformi alle normative vigenti, riducano i rischi di violazione della privacy e stabiliscano chiaramente le responsabilità degli enti. Altre risorse utili sono: Best Privacy Practices for Commercial Use of Facial Recognition, pubblicato dalla National Telecommunications and Information Administration, e Best Practices for Common Uses of Facial Recognition redatto dal personale della Federal Trade Commission. In futuro potrebbero essere pubblicate e rese disponibili ulteriori risorse ed è consigliabile essere sempre bene informati e aggiornati su questo argomento così importante.

Ricorda di utilizzare sempre i servizi AWS nel rispetto delle leggi vigenti. Non ti è consentito fare uso dei servizi AWS in modi che violino i diritti delle persone o che possano risultare pericolosi per altri utenti. Ciò significa che, per casi d'uso riguardanti la sicurezza, non è possibile utilizzare i servizi AWS in modi che rappresentino una discriminazione illegale di una persona o che violino i diritti, la privacy o le libertà civili di una persona. È consigliabile richiedere una consulenza legale per valutare tutte le implicazioni e le eventuali limitazioni relative al caso d'uso.

Utilizzo di Amazon Rekognition per favorire la pubblica sicurezza

Amazon Rekognition può essere d'aiuto negli scenari della pubblica sicurezza e delle forze dell'ordine, ad esempio nella ricerca di bambini smarriti, nella lotta alla tratta di esseri umani o nella prevenzione dei reati. Negli scenari della pubblica sicurezza e delle forze dell'ordine, occorre considerare quanto segue:

- Utilizza Amazon Rekognition come prima fase per trovare le possibili corrispondenze. Le risposte dalle operazioni relative ai volti di Amazon Rekognition consentono di ottenere rapidamente un set di potenziali corrispondenze da esaminare ulteriormente.
- Non usare le risposte di Amazon Rekognition per prendere decisioni autonome per gli scenari che richiedono l'analisi da parte di un essere umano. Se sei un'agenzia incaricata dell'applicazione della legge che aiuta Amazon Rekognition a identificare una persona in relazione a un'indagine penale, e le azioni saranno intraprese in base all'identificazione che potrebbe influire sulle libertà civili di tale persona o sui diritti umani equivalenti, la decisione di agire deve essere presa

da un persona adeguatamente addestrata sulla base dell'esame indipendente delle prove di identificazione.

- Utilizzare una soglia di somiglianza del 99% per scenari in cui sono necessarie corrispondenze di somiglianza dei volti estremamente accurate. Un esempio di questo è l'autenticazione dell'accesso a un edificio.
- Quando i diritti civili sono un problema, come i casi di utilizzo che comportano l'applicazione della legge, utilizza soglie di confidenza del 99% o superiori e usa la revisione umana delle previsioni di confronto facciale per garantire che i diritti civili di una persona non siano violati.
- Utilizzare una soglia di somiglianza inferiore al 99% per gli scenari che traggono vantaggio da un maggior numero di potenziali corrispondenze. Un esempio di questo è la ricerca di persone disperse. Se necessario, è possibile utilizzare l'attributo di risposta di somiglianza per determinare il livello di somiglianza di potenziali corrispondenze rispetto alla persona che si desidera riconoscere.
- Avere un piano per le corrispondenze di volti false positive restituite da Amazon Rekognition. Ad esempio, è possibile migliorare la corrispondenza utilizzando più immagini della stessa persona al momento della creazione dell'indice con l'operazione IndexFaces. Per ulteriori informazioni, consulta Linee guida per l'indicizzazione dei volti in scenari comuni.

Per altri casi d'uso (ad esempio nei social media), ti consigliamo di usare le tue capacità di giudizio per valutare se i risultati forniti da Amazon Rekognition possano richiedere una verifica da parte di una persona. Inoltre, a seconda dei requisiti dell'applicazione, la soglia di somiglianza può essere inferiore.

Creazione di una raccolta

È possibile utilizzare l'operazione CreateCollection per creare una raccolta.

Per ulteriori informazioni, consulta Gestire una raccolta.

Per creare una raccolta (SDK)

1. Se non lo hai già fatto:

   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

b. Installa e configura il e il. AWS CLI AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Utilizzare i seguenti esempi per richiamare l'operazione `CreateCollection`.

Java

L'esempio seguente crea una raccolta e visualizza il relativo nome della risorsa Amazon (ARN).

Modifica il valore di `collectionId` nel nome della raccolta che desideri creare.

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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
//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

Questo AWS CLI comando visualizza l'output JSON per l'operazione `create-collection` CLI.

Sostituisci il valore di `collection-id` con il nome della raccolta che desideri creare.

Sostituisci il valore di `profile_name` con il nome del tuo profilo di sviluppatore.

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

Python

L'esempio seguente crea una raccolta e visualizza il relativo nome della risorsa Amazon (ARN).

Modifica il valore di `collection_id` nel nome della raccolta che desideri creare. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

L'esempio seguente crea una raccolta e visualizza il relativo nome della risorsa Amazon (ARN).

Modifica il valore di `collectionId` nel nome della raccolta che desideri creare.

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

Nell'esempio seguente, sostituisci il valore di `region` con il nome della regione associata al tuo account e sostituisci il valore di `collectionName` con il nome desiderato della tua raccolta.

Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
//Copyright 2018 Amazon.com, Inc. or its 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 richiesta di operazione

L'input per `CreationCollection` è il nome della raccolta che si desidera creare.

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

CreateCollection risposta operativa

Amazon Rekognition crea la raccolta e restituisce il nome della risorsa Amazon (ARN) della nuova raccolta.

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

Assegnazione di tag alle raccolte

È possibile identificare, organizzare, cercare e filtrare le raccolte Amazon Rekognition utilizzando i tag. Ogni tag è un'etichetta composta da una chiave e da un valore definiti dall'utente.

Puoi anche utilizzare i tag per controllare l'accesso a una raccolta utilizzando Identity and Access Management (IAM). Per ulteriori informazioni, vedere Controllo dell'accesso alle AWS risorse mediante i tag delle risorse.

Argomenti

- Aggiunta di tag a una nuova raccolta
- Aggiunta di tag a una raccolta esistente
- Creazione dell'elenco dei tag di una raccolta
- Eliminazione dei tag da una raccolta

Aggiunta di tag a una nuova raccolta

Puoi aggiungere tag a una raccolta al momento della creazione utilizzando l'operazione `CreateCollection`. Specifica uno o più tag nel parametro di input dell'array `Tags`.

AWS CLI

Sostituisci il valore di `profile_name` con il nome del tuo profilo di sviluppatore.

```
aws rekognition create-collection --collection-id "collection-name" --tags
 "{"key1":"value1","key2":"value2"}" --profile profile-name
```

Per i dispositivi Windows:

```
aws rekognition create-collection --collection-id "collection-name" --tags
 "{\"key1\":\"value1\",\"key2\":\"value2\"}" --profile profile-name
```

Python

Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
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()
```

Aggiunta di tag a una raccolta esistente

Per aggiungere uno o più tag a una raccolta esistente, utilizza l'operazione `TagResource`. Specifica il nome della risorsa Amazon (ARN) (`ResourceArn`) della raccolta e i tag (`Tags`) che intendi aggiungere. L'esempio seguente mostra come aggiungere due tag.

AWS CLI

Sostituisci il valore di `profile_name` con il nome del tuo profilo di sviluppatore.

```
aws rekognition tag-resource --resource-arn collection-arn --tags
 "{"key1":"value1","key2":"value2"}" --profile profile-name
```

Per i dispositivi Windows:

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

Python

Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
# Copyright 2018 Amazon.com, Inc. or its 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**
>
> Se non conosci il nome della risorsa Amazon della raccolta, puoi utilizzare l'operazione `DescribeCollection`.

Creazione dell'elenco dei tag di una raccolta

Per elencare i tag associati a una raccolta, utilizza l'operazione `ListTagsForResource` e specifica l'ARN della raccolta (`ResourceArn`). La risposta è una mappa di chiavi e valori dei tag associati alla raccolta specificata.

AWS CLI

Sostituisci il valore di `profile_name` con il nome del tuo profilo di sviluppatore.

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

Python

Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
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()
```

L'output mostra un elenco di tag associati alla raccolta:

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

Eliminazione dei tag da una raccolta

Per rimuovere uno o più tag da una raccolta, utilizza l'operazione `UntagResource`. Specifica l'ARN del modello (`ResourceArn`) e le chiavi di tag (`Tag-Keys`) che desideri rimuovere.

AWS CLI

Sostituisci il valore di `profile_name` con il nome del tuo profilo di sviluppatore.

```
aws rekognition untag-resource --resource-arn resource-arn --profile profile-name --
tag-keys "key1" "key2"
```

In alternativa, puoi specificare le chiavi di tag in questo formato:

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

Python

Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
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()
```

Creazione dell'elenco delle raccolte

È possibile utilizzare l'operazione ListCollections per elencare le raccolte nella regione che si sta utilizzando.

Per ulteriori informazioni, consulta Gestire una raccolta.

Per elencare le raccolte (SDK)

1. Se non lo hai già fatto:
   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Utilizzare i seguenti esempi per richiamare l'operazione `ListCollections`.

   Java

   Il seguente esempio elenca le raccolte nella regione corrente.

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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
//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

Questo AWS CLI comando visualizza l'output JSON per l'operazione `list-collections` CLI. Sostituisci il valore di `profile_name` con il nome del tuo profilo di sviluppatore.

```
aws rekognition list-collections --profile profile-name
```

Python

Il seguente esempio elenca le raccolte nella regione corrente.

Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

Il seguente esempio elenca le raccolte nella regione corrente.

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

Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
//Copyright 2018 Amazon.com, Inc. or its 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 richiesta di operazione

L'input per `ListCollections` è il numero massimo di raccolte da restituire.

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

```
}
```

Se la risposta ha più raccolte di quante richieste da `MaxResults`, viene restituito un token che è possibile utilizzare per ottenere il successivo set di risultati, in una chiamata successiva a `ListCollections`. Per esempio:

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

ListCollections risposta operativa

Amazon Rekognition restituisce una matrice di raccolte (`CollectionIds`). Una matrice separata (`FaceModelVersions`) fornisce la versione del modello di volti utilizzato per analizzare i volti in ciascuna raccolta. Ad esempio, nella seguente risposta JSON, la raccolta `MyCollection` analizza i volti utilizzando la versione 2.0 del modello di volti. La raccolta `AnotherCollection` utilizza la versione 3.0 del modello di volti. Per ulteriori informazioni, consulta Comprendere il controllo delle versioni dei modelli.

`NextToken` è il token utilizzato per ottenere il successivo set di risultati, in una chiamata successiva a `ListCollections`.

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

Descrizione di una raccolta

È possibile utilizzare l'operazione DescribeCollection per ottenere le seguenti informazioni su una raccolta:

- Il numero di volti indicizzati nella raccolta.
- La versione del modello utilizzato con la raccolta. Per ulteriori informazioni, consulta the section called “Comprendere il controllo delle versioni dei modelli”.
- Il nome della risorsa Amazon (ARN) della raccolta.
- Data e ora di creazione della raccolta.

Per descrivere una raccolta (SDK)

1. Se non lo hai già fatto:
   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Utilizzare i seguenti esempi per richiamare l'operazione `DescribeCollection`.

   Java

   In questo esempio viene descritta una raccolta.

   Modifica il valore di `collectionId` nell'ID della raccolta desiderata.

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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
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

Questo AWS CLI comando visualizza l'output JSON per l'operazione `describe-collection` CLI. Modifica il valore di `collection-id` nell'ID della raccolta desiderata. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
aws rekognition describe-collection --collection-id collection-name --profile
 profile-name
```

Python

In questo esempio viene descritta una raccolta.

Modifica il valore di `collection_id` nell'ID della raccolta desiderata. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
# Copyright 2018 Amazon.com, Inc. or its 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 questo esempio viene descritta una raccolta.

Modifica il valore di `collectionId` nell'ID della raccolta desiderata.

```
//Copyright 2018 Amazon.com, Inc. or its 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 richiesta di operazione

L'input per `DescribeCollection` è l'ID della raccolta desiderata, come mostrato nel seguente esempio JSON.

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

DescribeCollectionrisposta operativa

La risposta include:

- Il numero di volti indicizzati nella raccolta, `FaceCount`.
- La versione del modello di volto utilizzato con la raccolta, `FaceModelVersion`. Per ulteriori informazioni, consulta the section called “Comprendere il controllo delle versioni dei modelli”.
- Il nome della risorsa Amazon della raccolta, `CollectionARN`.
- La data e l'ora di creazione della raccolta, `CreationTimestamp`. Il valore di `CreationTimestamp` è il numero di millisecondi dal tempo epoca Unix fino alla creazione della raccolta. Il tempo epoca Unix è 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970. Per ulteriori informazioni, consulta Tempo (Unix).

```
{
    "CollectionARN": "arn:aws:rekognition:us-east-1:nnnnnnnnnnn:collection/
MyCollection",
    "CreationTimestamp": 1.533422155042E9,
    "FaceCount": 200,
    "UserCount" : 20,
    "FaceModelVersion": "1.0"
}
```

Eliminazione di una raccolta

È possibile utilizzare l'operazione DeleteCollection per eliminare una raccolta.

Per ulteriori informazioni, consulta Gestire una raccolta.

Per eliminare una raccolta (SDK)

1. Se non lo hai già fatto:
   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Utilizzare i seguenti esempi per richiamare l'operazione `DeleteCollection`.

   Java

   In questo esempio viene eliminata una raccolta.

   Modifica il valore `collectionId` nella raccolta che desideri eliminare.

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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
// 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

Questo AWS CLI comando visualizza l'output JSON per l'operazione `delete-collection` CLI. Sostituisci il valore di `collection-id` con il nome della raccolta che desideri eliminare. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
aws rekognition delete-collection --collection-id collection-name --profile
 profile-name
```

Python

In questo esempio viene eliminata una raccolta.

Modifica il valore `collection_id` nella raccolta che desideri eliminare. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
# Copyright 2018 Amazon.com, Inc. or its 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 questo esempio viene eliminata una raccolta.

Modifica il valore `collectionId` nella raccolta che desideri eliminare.

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

Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
//Copyright 2018 Amazon.com, Inc. or its 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 richiesta di operazione

L'input per `DeleteCollection` è l'ID della raccolta da eliminare, come mostrato nel seguente esempio JSON.

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

DeleteCollection risposta operativa

La risposta `DeleteCollection` contiene un codice di stato HTTP che indica l'esito positivo o negativo dell'operazione. `200` viene restituito in caso di eliminazione della raccolta.

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

Aggiunta di volti a una raccolta

È possibile utilizzare l'operazione IndexFaces per rilevare i volti in un'immagine e aggiungerli a una raccolta. Per ogni volto rilevato, Amazon Rekognition ne estrae le caratteristiche e ne archivia le informazioni in un database. Inoltre, il comando archivia i metadati per ogni volto rilevato nella raccolta specificata. Amazon Rekognition non archivia i byte di immagine effettivi.

Per informazioni sulla fornitura di volti idonei per l'indicizzazione, consulta Raccomandazioni per immagini di input di confronto di volti.

Per ogni volto, l'operazione `IndexFaces` rende persistenti le informazioni seguenti:

- Caratteristiche multidimensionali dei volti: `IndexFaces` utilizza l'analisi dei volti per estrarre informazioni multidimensionali sulle caratteristiche dei volti e archiviarle nella raccolta. Non è possibile accedere direttamente a queste informazioni. Tuttavia, Amazon Rekognition utilizza queste informazioni durante la ricerca di corrispondenze in una raccolta di volti.

- Metadati: i metadati per ogni faccia includono un riquadro di delimitazione, un livello di confidenza (che il riquadro di delimitazione contenga una faccia), assegnato da IDs Amazon Rekognition (face ID e image ID) e un ID immagine esterno (se lo hai fornito) nella richiesta. Queste informazioni vengono restituite in risposta alla chiamata API `IndexFaces`. Per un esempio, consulta l'elemento `face` nella seguente risposta di esempio.

Il servizio restituisce questi metadata in risposta alle seguenti chiamate API:

- `ListFaces`
- Operazioni di ricerca di volti - Le risposte per SearchFaces e SearchFacesByImage restituiscono il livello di affidabilità per ogni volto corrispondente, insieme a questi metadata del volto.

Il numero di volti indicizzati da `IndexFaces` dipende della versione del modello di rilevamento volti associato alla raccolta di input. Per ulteriori informazioni, consulta Comprendere il controllo delle versioni dei modelli.

Informazioni su come i volti indicizzati vengono restituiti in una serie di oggetti FaceRecord.

È possibile associare volti indicizzati all'immagine in cui sono stati rilevati. Ad esempio, è possibile mantenere un indice lato client di immagini e volti nelle immagini. Per associare volti a un'immagine, specificare un'ID immagine nel parametro di richiesta `ExternalImageId`. L'ID immagine può essere il nome del file o un altro ID creato.

Oltre alle informazioni precedenti trattenute dall'API nella raccolta di volti, l'API restituisce inoltre i dettagli dei volti non trattenuti nella raccolta. (Consulta l'elemento `faceDetail` nella seguente risposta di esempio).

> **Note**
>
> `DetectFaces` restituisce le stesse informazioni, perciò non è necessario chiamare sia `DetectFaces` che `IndexFaces` per la stessa immagine.

Filtraggio dei volti

L' IndexFaces operazione consente di filtrare i volti indicizzati da un'immagine. Con `IndexFaces` è possibile specificare un numero massimo di volti da indicizzare, oppure è possibile scegliere solo facce rilevate con un indice di alta qualità.

È possibile specificare il numero massimo di volti indicizzati, `IndexFaces` utilizzando il parametro di input `MaxFaces`. Questa funzione è utile quando si vogliono indicizzare i volti più grandi in un'immagine e non le facce più piccole, come quelle di persone che sono sullo sfondo.

Per impostazione predefinita, `IndexFaces` sceglie un livello di qualità utilizzato per filtrare i volti. È possibile utilizzare il parametro di input `QualityFilter` per impostare esplicitamente il livello di qualità. I valori sono:

- `AUTO`: Amazon Rekognition sceglie il livello di qualità utilizzato per filtrare i volti (valore predefinito).
- `LOW`: tutti i volti, tranne quelli di qualità più bassa, vengono indicizzati.
- `MEDIUM`
- `HIGH`: vengono indicizzati solo i volti di migliore qualità.
- `NONE`: non viene indicizzato alcun volto in base alla qualità.

`IndexFaces` filtra i volti per i seguenti motivi:

- Il volto è troppo piccolo rispetto alle dimensioni dell'immagine.
- Il volto è troppo sfocato.
- L'immagine è troppo scura.
- La faccia ha una posa esagerata.
- Il volto non ha abbastanza dettagli per essere adatto alla ricerca di volti.

> **Note**
>
> Per utilizzare il filtraggio della qualità, hai bisogno di una raccolta associata alla versione 3 o superiore del modello facciale. Per avere la versione del modello facciale associato a una raccolta contatta DescribeCollection.

Le informazioni sui volti che non sono indicizzati da `IndexFaces` vengono restituite in una serie di oggetti UnindexedFace. La serie `Reasons` contiene un elenco di motivi per cui un volto non è stato indicizzato. Ad esempio, il valore `EXCEEDS_MAX_FACES` rappresenta un volto non indicizzato poiché il numero di volti specificato da `MaxFaces` è già stato rilevato.

Per ulteriori informazioni, consulta Gestione dei volti in una raccolta.

Per aggiungere volti a una raccolta (SDK)

1. Se non lo hai già fatto:

a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

b. Installa e configura il AWS CLI e il. AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Carica un'immagine (contenente uno o più volti) nel bucket Amazon S3.

   Per le istruzioni, consulta Caricamento di oggetti in Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

3. Utilizzare i seguenti esempi per richiamare l'operazione `IndexFaces`.

   Java

   In questo esempio vengono visualizzati gli identificatori dei volti aggiunti alla raccolta.

   Modifica il valore di `collectionId` nel nome della raccolta a cui desideri aggiungere un volto. Sostituisci i valori `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2. Il parametro `.withMaxFaces(1)` limita il numero di facce indicizzate a 1. Rimuovi o modifica il valore in base alle tue esigenze.

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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
//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

Questo AWS CLI comando visualizza l'output JSON per l'operazione `index-faces` CLI.

Sostituisci il valore di `collection-id` con il nome della raccolta in cui desideri archiviare il volto. Sostituisci i valori `Bucket` e `Name` con il nome del bucket Amazon S3 e il nome del file di immagine utilizzato nella fase 2. Il parametro `max-faces` limita il numero di facce indicizzate a 1. Rimuovi o modifica il valore in base alle tue esigenze. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
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
```

Se accedi alla CLI da un dispositivo Windows, usa le virgolette doppie anziché le virgolette singole ed evita le virgolette doppie interne tramite barra rovesciata (ovvero, \) per risolvere eventuali errori del parser che potresti riscontrare. Per un esempio, consulta quanto segue:

```
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

In questo esempio vengono visualizzati gli identificatori dei volti aggiunti alla raccolta.

Modifica il valore di `collectionId` nel nome della raccolta a cui desideri aggiungere un volto. Sostituisci i valori `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2. Il parametro di input `MaxFaces` limita il numero di facce indicizzate a 1. Rimuovi o modifica il valore in base alle tue esigenze. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

In questo esempio vengono visualizzati gli identificatori dei volti aggiunti alla raccolta.

Modifica il valore di `collectionId` nel nome della raccolta a cui desideri aggiungere un volto. Sostituisci i valori `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2.

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

using System;
using System.Collections.Generic;
using Amazon.Rekognition;
```

```
using Amazon.Rekognition.Model;

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

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        Image image = new Image()
        {
            S3Object = new S3Object()
            {
                Bucket = bucket,
                Name = photo
            }
        };

        IndexFacesRequest indexFacesRequest = new IndexFacesRequest()
        {
            Image = image,
            CollectionId = collectionId,
            ExternalImageId = photo,
            DetectionAttributes = new List<String>(){ "ALL" }
        };

        IndexFacesResponse indexFacesResponse =
 rekognitionClient.IndexFaces(indexFacesRequest);

        Console.WriteLine(photo + " added");
        foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
            Console.WriteLine("Face detected: Faceid is " +
               faceRecord.Face.FaceId);
    }
}
```

IndexFaces richiesta di operazione

L'input per `IndexFaces` è l'immagine da indicizzare e la raccolta a cui aggiungere i volti.

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "ExternalImageId": "input.jpg",
    "DetectionAttributes": [
        "DEFAULT"
    ],
    "MaxFaces": 1,
    "QualityFilter": "AUTO"
}
```

IndexFaces risposta operativa

`IndexFaces` restituisce le informazioni sui volti rilevati nell'immagine. Ad esempio, la seguente risposta JSON include gli attributi di rilevamento predefiniti per i volti rilevati nell'immagine di input. L'esempio mostra anche volti non indicizzati, poiché il valore del parametro di input `MaxFaces` è stato superato: la matrice `Reasons` contiene EXCEEDS_MAX_FACES. Se una faccia non è indicizzata per motivi di qualità, `Reasons` contiene valori come LOW_SHARPNESS o LOW_BRIGHTNESS. Per ulteriori informazioni, consulta 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"
            ]
        }
    ]
}
```

Per ottenere tutte le informazioni sui volti, specificare 'ALL' per il parametro di richiesta `DetectionAttributes`. Ad esempio, nella seguente risposta, notare le informazioni aggiuntive nell'elemento `faceDetail`, non conservato nel server:

- 25 punti di riferimento del volto (rispetto a solo cinque nell'esempio precedente)
- Dieci attributi del volto (occhiali, barba, occlusione, direzione dello sguardo e così via)
- Emozioni (osserva l'elemento `emotion`)

L'elemento `face` fornisce i metadati resi persistenti nel server.

`FaceModelVersion` è la versione del modello di volto associato alla raccolta. Per ulteriori informazioni, consulta Comprendere il controllo delle versioni dei modelli.

`OrientationCorrection` è l'orientamento stimato dell'immagine. Le informazioni sulla correzione dell'orientamento non vengono restituite se utilizzi un modello di rilevamento facciale successivo alla versione 3. Per ulteriori informazioni, consulta Ottenere l'orientamento dell'immagine e le coordinate del riquadro di delimitazione.

Il seguente esempio di risposta mostra il JSON restituito quando si specifica ["ALL"]:

```
{
    "FaceModelVersion": "3.0",
    "FaceRecords": [
        {
```

```
            "Face": {
                "BoundingBox": {
                    "Height": 0.06333333253860474,
                    "Left": 0.17185185849666595,
                    "Top": 0.7366666793823242,
                    "Width": 0.11061728745698929
                },
                "Confidence": 99.99999237060547,
                "ExternalImageId": "input.jpg",
                "FaceId": "578e2e1b-d0b0-493c-aa39-ba476a421a34",
                "ImageId": "9ba38e68-35b6-5509-9d2e-fcffa75d1653"
            },
            "FaceDetail": {
                "AgeRange": {
                    "High": 25,
                    "Low": 15
                },
                "Beard": {
                    "Confidence": 99.98077392578125,
                    "Value": false
                },
                "BoundingBox": {
                    "Height": 0.06333333253860474,
                    "Left": 0.17185185849666595,
                    "Top": 0.7366666793823242,
                    "Width": 0.11061728745698929
                },
                "Confidence": 99.99999237060547,
                "Emotions": [
                    {
                        "Confidence": 95.40877532958984,
                        "Type": "HAPPY"
                    },
                    {
                        "Confidence": 6.6088080406188965,
                        "Type": "CALM"
                    },
                    {
                        "Confidence": 0.7385611534118652,
                        "Type": "SAD"
                    }
                ],
                 "EyeDirection": {
                    "yaw": 16.299732,
```

```
                    "pitch": -6.407457,
                    "confidence": 99.968704
                              }
                "Eyeglasses": {
                    "Confidence": 99.96795654296875,
                    "Value": false
                },
                "EyesOpen": {
                    "Confidence": 64.0671157836914,
                    "Value": true
                },
                "Gender": {
                    "Confidence": 100,
                    "Value": "Female"
                },
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.21361233294010162,
                        "Y": 0.757106363773346
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.2518567442893982,
                        "Y": 0.7599404454231262
                    },
                    {
                        "Type": "nose",
                        "X": 0.2262365221977234,
                        "Y": 0.7711842060089111
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.2050037682056427,
                        "Y": 0.7801263332366943
                    },
                    {
                        "Type": "mouthRight",
                        "X": 0.2430567592382431,
                        "Y": 0.7836716771125793
                    },
                    {
                        "Type": "leftPupil",
                        "X": 0.2161938101053238,
```

```
                    "Y": 0.756662905216217
                },
                {
                    "Type": "rightPupil",
                    "X": 0.2523181438446045,
                    "Y": 0.7603650689125061
                },
                {
                    "Type": "leftEyeBrowLeft",
                    "X": 0.20066319406032562,
                    "Y": 0.7501518130302429
                },
                {
                    "Type": "leftEyeBrowUp",
                    "X": 0.2130996286869049,
                    "Y": 0.7480520606040955
                },
                {
                    "Type": "leftEyeBrowRight",
                    "X": 0.22584207355976105,
                    "Y": 0.7504606246948242
                },
                {
                    "Type": "rightEyeBrowLeft",
                    "X": 0.24509544670581818,
                    "Y": 0.7526801824569702
                },
                {
                    "Type": "rightEyeBrowUp",
                    "X": 0.2582615911960602,
                    "Y": 0.7516844868659973
                },
                {
                    "Type": "rightEyeBrowRight",
                    "X": 0.26881539821624756,
                    "Y": 0.7554477453231812
                },
                {
                    "Type": "leftEyeLeft",
                    "X": 0.20624476671218872,
                    "Y": 0.7568746209144592
                },
                {
                    "Type": "leftEyeRight",
```

```
                    "X": 0.22105035185813904,
                    "Y": 0.7582521438598633
                },
                {
                    "Type": "leftEyeUp",
                    "X": 0.21401576697826385,
                    "Y": 0.7553104162216187
                },
                {
                    "Type": "leftEyeDown",
                    "X": 0.21317370235919952,
                    "Y": 0.7584449648857117
                },
                {
                    "Type": "rightEyeLeft",
                    "X": 0.24393919110298157,
                    "Y": 0.7600628137588501
                },
                {
                    "Type": "rightEyeRight",
                    "X": 0.2598416209220886,
                    "Y": 0.7605880498886108
                },
                {
                    "Type": "rightEyeUp",
                    "X": 0.2519053518772125,
                    "Y": 0.7582084536552429
                },
                {
                    "Type": "rightEyeDown",
                    "X": 0.25177454948425293,
                    "Y": 0.7612871527671814
                },
                {
                    "Type": "noseLeft",
                    "X": 0.2185886949300766,
                    "Y": 0.774715781211853
                },
                {
                    "Type": "noseRight",
                    "X": 0.23328955471515656,
                    "Y": 0.7759330868721008
                },
                {
```

```
                        "Type": "mouthUp",
                        "X": 0.22446128726005554,
                        "Y": 0.7805567383766174
                    },
                    {
                        "Type": "mouthDown",
                        "X": 0.22087252140045166,
                        "Y": 0.7891407608985901
                    }
                ],
                "MouthOpen": {
                    "Confidence": 95.87068939208984,
                    "Value": false
                },
                "Mustache": {
                    "Confidence": 99.9828109741211,
                    "Value": false
                },
                "Pose": {
                    "Pitch": -0.9409101605415344,
                    "Roll": 7.233824253082275,
                    "Yaw": -2.3602254390716553
                },
                "Quality": {
                    "Brightness": 32.01998519897461,
                    "Sharpness": 93.67259216308594
                },
                "Smile": {
                    "Confidence": 86.7142105102539,
                    "Value": true
                },
                "Sunglasses": {
                    "Confidence": 97.38925170898438,
                    "Value": false
                }
            }
        }
    ],
    "OrientationCorrection": "ROTATE_0"
    "UnindexedFaces": []
}
```

Creazione dell'elenco dei volti e degli utenti associati in una raccolta

È possibile utilizzare l'ListFacesoperazione per elencare i volti e gli utenti associati in una raccolta.

Per ulteriori informazioni, consulta Gestione dei volti in una raccolta.

Per elencare i volti in una raccolta (SDK)

1. Se non lo hai già fatto:
   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Utilizzare i seguenti esempi per richiamare l'operazione `ListFaces`.

   Java

   In questo esempio viene visualizzato un elenco di volti in una raccolta.

   Modifica il valore di `collectionId` nella raccolta desiderata.

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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
// 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

Questo AWS CLI comando visualizza l'output JSON per l'operazione `list-faces` CLI. Sostituisci il valore di `collection-id` con il nome della raccolta che desideri elencare. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
aws rekognition list-faces --collection-id "collection-id"  --profile profile-
name
```

Python

In questo esempio viene visualizzato un elenco di volti in una raccolta.

Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

In questo esempio viene visualizzato un elenco di volti in una raccolta.

Modifica il valore di `collectionId` nella raccolta desiderata.

```
//Copyright 2018 Amazon.com, Inc. or its 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 richiesta di operazione

L'input to `ListFaces` è l'ID della raccolta per cui desideri elencare i volti. `MaxResults`è il numero massimo di facce da restituire. ListFaces inserisce anche un elenco di volti con IDs cui filtrare i risultati e un ID utente fornito per elencare solo i volti associati all'utente specificato.

```
{
    "CollectionId": "MyCollection",
    "MaxResults": 1
}
```

Se la risposta ha più volti di quanti richiesti da `MaxResults`, viene restituito un token che è possibile utilizzare per ottenere il successivo set di risultati, in una chiamata successiva a `ListFaces`. Per esempio:

```
{
    "CollectionId": "MyCollection",
    "NextToken": "sm+5ythT3aeEVIR4WA....",
    "MaxResults": 1
}
```

ListFaces risposta all'operazione

La risposta di `ListFaces` contiene le informazioni relative ai metadati dei volti memorizzati nella raccolta specificata.

- FaceModelVersion— La versione del modello facciale associata alla collezione. Per ulteriori informazioni, consulta Comprendere il controllo delle versioni dei modelli.
- Faces: le informazioni relative ai volti nella raccolta. Sono disponibili informazioni relative a BoundingBox, affidabilità, identificatori di immagine e ID volto. Per ulteriori informazioni, consulta la sezione Volto.
- NextToken— Il token utilizzato per ottenere il prossimo set di risultati.

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

Eliminazione dei volti da una raccolta

È possibile utilizzare l'operazione DeleteFaces per eliminare i volti da una raccolta. Per ulteriori informazioni, consulta Gestione dei volti in una raccolta.

Per eliminare i volti da una raccolta

1. Se non lo hai già fatto:
   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Utilizzare i seguenti esempi per richiamare l'operazione `DeleteFaces`.

   Java

   Questo esempio elimina un singolo volto da una raccolta.

   Modifica il valore di `collectionId` nel nome della raccolta che contiene il volto che desideri eliminare. Modifica il valore di `faces`nell'ID del volto che desideri eliminare. Per eliminare più facce, aggiungete la faccia IDs all'`faces`array.

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

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.DeleteFacesRequest;
import com.amazonaws.services.rekognition.model.DeleteFacesResult;

import java.util.List;


public class DeleteFacesFromCollection {
   public static final String collectionId = "MyCollection";
```

```
   public static final String faces[] = {"xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx"};

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

      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();


      DeleteFacesRequest deleteFacesRequest = new DeleteFacesRequest()
              .withCollectionId(collectionId)
              .withFaceIds(faces);

      DeleteFacesResult
 deleteFacesResult=rekognitionClient.deleteFaces(deleteFacesRequest);


      List < String > faceRecords = deleteFacesResult.getDeletedFaces();
      System.out.println(Integer.toString(faceRecords.size()) + " face(s)
 deleted:");
      for (String face: faceRecords) {
         System.out.println("FaceID: " + face);
      }
   }
}
```

Java V2

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
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

Questo AWS CLI comando visualizza l'output JSON per l'operazione `delete-faces` CLI. Sostituisci il valore di `collection-id` con il nome della raccolta che contiene il volto che desideri eliminare. Sostituisci il valore di `face-ids` con una matrice di volti IDs che desideri eliminare. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
aws rekognition delete-faces --collection-id "collection-id" --face-ids "faceid"
 --profile profile-name
```

Python

Questo esempio elimina un singolo volto da una raccolta.

Modifica il valore di `collectionId` nel nome della raccolta che contiene il volto che desideri eliminare. Modifica il valore di `faces`nell'ID del volto che desideri eliminare. Per eliminare più facce, aggiungete la faccia IDs all'`faces`array. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

Questo esempio elimina un singolo volto da una raccolta.

Modifica il valore di `collectionId` nel nome della raccolta che contiene il volto che desideri eliminare. Modifica il valore di `faces`nell'ID del volto che desideri eliminare. Per eliminare più facce, aggiungete la faccia IDs all'`faces`elenco.

```
//Copyright 2018 Amazon.com, Inc. or its 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 richiesta di operazione

L'input to `DeleteFaces` è l'ID della raccolta che contiene i volti e una matrice di volti IDs per i volti da eliminare.

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

DeleteFaces risposta all'operazione

La `DeleteFaces` risposta contiene una serie di volti IDs per i volti che sono stati eliminati.

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

Se i volti IDs forniti nell'input sono attualmente associati a un utente, verranno restituiti UnsuccessfulFaceDeletions con validi motivi.

```
{
   "DeletedFaces": [
        "daf29cac-f910-41e9-851f-6eeb0e08f973"
    ],
     "UnsuccessfulFaceDeletions" : [
        {
           "FaceId" : "0b683aed-a0f1-48b2-9b5e-139e9cc2a757",
           "UserId" : "demoUser1",
           "Reason" : ["ASSOCIATED_TO_AN_EXISTING_USER"]
        }
     ]
}
```

Creazione di un utente

È possibile utilizzare l'CreateUseroperazione per creare un nuovo utente in una raccolta utilizzando un ID utente univoco fornito dall'utente. È quindi possibile associare più volti all'utente appena creato.

Per creare un utente (SDK)

1. Se non lo hai già fatto:
   a. Crea o aggiorna un account utente IAM con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Utilizzare i seguenti esempi per richiamare l'operazione `CreateUser`.

Java

Questo esempio di codice Java crea un utente.

```
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

Questo AWS CLI comando crea un utente utilizzando l'operazione `create-user` CLI.

```
aws rekognition create-user --user-id user-id --collection-id collection-name --
region region-name
```

```
--client-request-token request-token
```

Python

Questo esempio di codice Python crea un utente.

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

Questo esempio di codice Go utilizza AWS Go SDK V2 e crea un utente.

```
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")

}
```

Eliminazione di un utente

È possibile utilizzare l'DeleteUseroperazione per eliminare un utente da una raccolta, in base allo UserID fornito. Tieni presente che tutti i volti associati allo UserID vengono dissociate dallo UserID prima che lo UserID specificato venga eliminato.

Per eliminare un utente (SDK)

1. Se non lo hai già fatto:

   a. Crea o aggiorna un account utente IAM con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

   b. Installa e configura il AWS CLI e il. AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Utilizzare i seguenti esempi per richiamare l'operazione `DeleteUser`.

   Java

   Questo esempio di codice Java elimina un utente.

```
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

Questo AWS CLI comando elimina un utente utilizzando l'operazione `create-user` CLI.

```
aws rekognition delete-user --collection-id MyCollection
--user-id user-id --collection-id collection-name --region region-name
```

Python

Questo esempio di codice Python elimina un utente.

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

Associazione di volti a un utente

È possibile utilizzare l'AssociateFacesoperazione per associare più volti singoli a un singolo utente. Per associare un volto a un utente, devi prima creare una raccolta e un utente. Nota che i vettori di volti devono risiedere nella stessa raccolta in cui risiede il vettore utente.

Per associare volti (SDK)

1. Se non lo hai già fatto:

a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Utilizzare i seguenti esempi per richiamare l'operazione `AssociateFaces`.

Java

Questo esempio di codice Java associa un volto a un utente.

```
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

Questo AWS CLI comando associa un volto a un utente, utilizzando l'operazione `associate-faces` CLI.

```
aws rekognition associate-faces --user-id user-id --face-ids face-id-1 face-id-2
--collection-id collection-name
--region region-name
```

Python

Questo esempio di codice Python associa un volto a un utente.

```
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 risposta all'operazione

La risposta per `AssociateFaces` include lo `UserStatus`, che è lo stato della richiesta di dissociazione, oltre a un elenco di `FaceIds` da associare. Viene inoltre restituito un elenco di `UnsuccessfulFaceAssociations`. Dopo aver inviato una richiesta di `AssociateFaces`, il completamento dell'operazione potrebbe richiedere circa un minuto.

Per questo motivo, UserStatus viene restituito, che può avere i seguenti valori:

- CREATO: indica che l'"Utente" è stato creato correttamente e che attualmente non ha alcun volto associato. L' 'utente' sarà in questo stato prima che venga effettuata una chiamata AssociateFaces " riuscita.
- AGGIORNAMENTO: indica che l'"Utente" sta venendo aggiornato per riflettere i volti appena associati/dissociati e diventerà ATTIVO in pochi secondi. I risultati di ricerca possono contenere "Utente" in questo stato e i clienti possono scegliere di ignorarlo nei risultati restituiti.
- ATTIVO: indica che l'"Utente" è aggiornato in modo da riflettere tutti i volti associati/dissociati ed è in uno stato ricercabile.

```
{
    "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"
        }
```

```
    ]
}
```

Dissociazione dei volti da un utente

È possibile utilizzare l'DisassociateFacesoperazione per rimuovere l'associazione tra un ID utente e un Face ID.

Per associare volti (SDK)

1. Se non lo hai già fatto:
   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Utilizzare i seguenti esempi per richiamare l'operazione `DisassociateFaces`.

   Java

   Questo esempio di Java rimuove l'associazione tra un FaceID e uno UserID con l'operazione `DisassociateFaces`.

```
import java.util.Arrays;
import java.util.List;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.DisassociateFacesRequest;
import com.amazonaws.services.rekognition.model.DisassociateFacesResult;


public class DisassociateFaces {

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


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

```
        /* Replace the below configurations to allow you successfully run the
 example

           @collectionId: The collection where user and faces are stored
           @userId: The user which faces will get disassociated from
           @faceIds: The list of face IDs that will get disassociated from the
 given user
         */

        String collectionId = "MyCollection";
        String userId = "demoUser";
        String faceId1 = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";
        String faceId2 = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";
        List<String> faceIds = Arrays.asList(faceid1,faceid2);

        System.out.println("Disassociating faces from existing user: " +
                userId);

        DisassociateFacesRequest request = new DisassociateFacesRequest()
                .withCollectionId(collectionId)
                .withUserId(userId)
                .withFaceIds(faceIds)

        DisassociateFacesResult result =
 rekognitionClient.disassociateFaces(request);

        System.out.println("Successful face disassociations: " +
 result.getDisassociatedFaces().size());
        System.out.println("Unsuccessful face disassociations: " +
 result.getUnsuccessfulFaceDisassociations().size());
    }

}
```

AWS CLI

Questo AWS CLI comando rimuove l'associazione tra un FaceID e un UserID con l'operazione. `DisassociateFaces`

```
aws rekognition disassociate-faces --face-ids list-of-face-ids
--user-id user-id --collection-id collection-name --region region-name
```

Python

L'esempio seguente rimuove l'associazione tra un FaceID e uno UserID con l'operazione `DisassociateFaces`.

```
from botocore.exceptions import ClientError
import boto3
import logging

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def disassociate_faces(collection_id, user_id, face_ids):
    """
    Disassociate stored faces within collection to the given user

    :param collection_id: The ID of the collection where user and faces are
 stored.
    :param user_id: The ID of the user that we want to disassociate faces from
    :param face_ids: The list of face IDs to be disassociated from the given
 user

    :return: response of AssociateFaces API
    """
    logger.info(f'Disssociating faces from user: {user_id}, {face_ids}')
    try:
        response = client.disassociate_faces(
            CollectionId=collection_id,
            UserId=user_id,
            FaceIds=face_ids
        )
        print(f'- disassociated {len(response["DisassociatedFaces"])} faces')
    except ClientError:
        logger.exception("Failed to disassociate faces from the given user")
        raise
    else:
        print(response)
        return response

def main():
    face_ids = ["faceId1", "faceId2"]
    collection_id = "collection-id"
```

```
    user_id = "user-id"
    disassociate_faces(collection_id, user_id, face_ids)

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

DisassociateFaces risposta all'operazione

La risposta per `DisassociateFaces` include lo `UserStatus`, che è lo stato della richiesta di dissociazione, oltre a un elenco di `FaceIds` da disassociare. Viene inoltre restituito un elenco di `UnsuccessfulFaceDisassociations`. Dopo aver inviato una richiesta a DisassociateFaces, il completamento dell'operazione potrebbe richiedere circa un minuto. Per questo motivo, UserStatus viene restituito il, che può avere i seguenti valori:

- CREATO: indica che l'"Utente" è stato creato correttamente e che attualmente non ha alcun volto associato. L' 'utente' sarà in questo stato prima che venga effettuata una chiamata AssociateFaces " riuscita.
- AGGIORNAMENTO: indica che l'"Utente" sta venendo aggiornato per riflettere i volti appena associati/dissociati e diventerà ATTIVO in pochi secondi. I risultati di ricerca possono contenere "Utente" in questo stato e i clienti possono scegliere di ignorarlo nei risultati restituiti.
- ATTIVO: indica che l'"Utente" è aggiornato in modo da riflettere tutti i volti associati/dissociati ed è in uno stato ricercabile.

```
{
    "UserStatus": "UPDATING",
    "DisassociatedFaces": [
        {
            "FaceId": "c92265d4-5f9c-43af-a58e-12be0ce02bc3"
        }
    ],
    "UnsuccessfulFaceDisassociations": [
        {
            "Reasons": [
                "ASSOCIATED_TO_A_DIFFERENT_IDENTITY"
            ],
            "FaceId": "f5817d37-94f6-4335-bfee-6cf79a3d806e",
            "UserId": "demoUser1"
        }
```

```
    ]
}
```

Creazione dell'elenco degli utenti in una raccolta

È possibile utilizzare l'ListUsersoperazione per elencare UserIds e il. UserStatus Per vedere IDs i volti associati a un UserID, usa l'operazione. ListFaces

Per elencare gli utenti (SDK)

1. Se non lo hai già fatto:
   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
   b. Installa e configura il AWS CLI e il. AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Utilizzare i seguenti esempi per richiamare l'operazione `ListUsers`.

   Java

   Questo esempio di Java elenca gli utenti di una raccolta utilizzando l'operazione `ListUsers`.

```
import java.util.List;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.ListUsersRequest;
import com.amazonaws.services.rekognition.model.ListUsersResult;
import com.amazonaws.services.rekognition.model.User;

public class ListUsers {

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


        AmazonRekognition amazonRekognition =
 AmazonRekognitionClientBuilder.defaultClient();


        System.out.println("Listing users");
        int limit = 10;
```

```
        ListUsersResult listUsersResult = null;
        String paginationToken = null;
        do {
            if (listUsersResult != null) {
                paginationToken = listUsersResult.getNextToken();
            }
            ListUsersRequest request = new ListUsersRequest()
                    .withCollectionId(collectionId)
                    .withMaxResults(limit)
                    .withNextToken(paginationToken);
            listUsersResult = amazonRekognition.listUsers(request);

            List<User> users = listUsersResult.getUsers();
            for (User currentUser: users) {
                System.out.println(currentUser.getUserId() + " : " +
 currentUser.getUserStatus());
            }
        } while (listUsersResult.getNextToken() != null);
    }
}
```

AWS CLI

Questo AWS CLI comando elenca gli utenti di una raccolta con l'`ListUsers`operazione.

```
aws rekognition list-users --collection-id collection-id --max-results number-
of-max-results
```

Python

L'esempio seguente elenca gli utenti di una raccolta con l'operazione `ListUsers`.

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

import boto3
from botocore.exceptions import ClientError
import logging
from pprint import pprint

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
```

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


def list_users(collection_id):
    """
    List all users from the given collection

    :param collection_id: The ID of the collection where user is stored.

    :return: response that contains list of Users found within given collection
    """
    logger.info(f'Listing the users in collection: {collection_id}')
    try:
        response = client.list_users(
            CollectionId=collection_id
        )
        pprint(response["Users"])
    except ClientError:
        logger.exception(f'Failed to list all user from given collection:
 {collection_id}')
        raise
    else:
        return response

def main():
    collection_id = "collection-id"
    list_users(collection_id)

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

ListUsers risposta all'operazione

La risposta a una richiesta ListUsers include un elenco degli elementi `Users` presenti nella raccolta insieme all'`UsedId`indirizzo e `UserStatus` dell'utente.

```
{
    "NextToken": "B1asJT3bAb/ttuGgPFV8BZoBZyGQzlUHXbuTNLh48a6enU7kXKw43hpOwizW7LOk/
Gk7Em09lznoq6+FcDCcSq2olrn7A98BLkt5keu+ZRVRUTyrXtT6J7Hmp
+ieQ2an6Zu0qzPfcdPeaJ9eAxG2d0WNrzJgi5hvmjoiSTTfKX3MQz1sduWQkvAAs4hZfhZoKFahFlqWofshCXa/
FHAAY3PL1PjxXbkNeSSMq8V7i1MlKCdrPVykCv9MokpPt7jtNvKPEZGUhxgBTFMxNWLEcFnzAiCWDg91dFy/
```

```
LalshPjXA9UVc5Gx9vIJNQ/
eO3cQRghAkCT3FOAiXsLAnA015ODTomZpWWVpqB21wKpI3LYmfAVFrDPGzpbTVlRmLsJm41bkmnBBBw9+DHz1Jn7zW
+qc5Fs3yaHu0f51Xg==",
    "Users": [
        {
            "UserId": "demoUser4",
            "UserStatus": "CREATED"
        },
        {
            "UserId": "demoUser2",
            "UserStatus": "CREATED"
        }
    ]
}
```

Ricerca di un volto con un ID volto

È possibile utilizzare l'SearchFacesoperazione per cercare gli utenti in una raccolta che corrispondano al volto più grande dell'immagine fornita.

L'ID volto viene restituito nella risposta dell'operazione IndexFaces quando il volto viene identificato e aggiunto a una raccolta. Per ulteriori informazioni, consulta Gestione dei volti in una raccolta.

Per cercare un volto in una raccolta utilizzando il relativo ID (SDK)

1. Se non lo hai già fatto:
   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Utilizzare i seguenti esempi per richiamare l'operazione `SearchFaces`.

   Java

   In questo esempio vengono visualizzate le informazioni relative ai volti che corrispondono a un volto identificato dal relativo ID.

Modifica il valore di `collectionID` nella raccolta che contiene il volto richiesto. Modifica il valore di `faceId` nell'identificatore del volto che desideri trovare.

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

}
```

Esegui il codice di esempio. Vengono visualizzate le informazioni relative ai volti corrispondenti.

Java V2

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
// 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

Questo AWS CLI comando visualizza l'output JSON per l'operazione `search-faces` CLI. Sostituisci il valore di `face-id` con l'identificatore del volto da ricercare e sostituisci il valore di `collection-id` con la raccolta in cui desideri eseguire la ricerca. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
aws rekognition search-faces --face-id face-id --collection-id "collection-id"
 --profile profile-name
```

Python

In questo esempio vengono visualizzate le informazioni relative ai volti che corrispondono a un volto identificato dal relativo ID.

Modifica il valore di `collectionID` nella raccolta che contiene il volto richiesto. Modifica il valore di `faceId` nell'identificatore del volto che desideri trovare. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

In questo esempio vengono visualizzate le informazioni relative ai volti che corrispondono a un volto identificato dal relativo ID.

Modifica il valore di `collectionID` nella raccolta che contiene il volto richiesto. Modifica il valore di `faceId` nell'identificatore del volto che desideri trovare.

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

Esegui il codice di esempio. Vengono visualizzate le informazioni relative ai volti corrispondenti.

SearchFaces richiesta di operazione

Per un ID volto specificato (ogni volto archiviato nella raccolta include un ID), `SearchFaces` cerca volti simili nella raccolta specificata. La risposta non include il volto che stai cercando. Include solo volti simili. Per impostazione predefinita, `SearchFaces` restituisce volti per cui l'algoritmo rileva una somiglianza superiore all'80%. La somiglianza indica il livello di corrispondenza del volto con il volto di input. È possibile utilizzare `FaceMatchThreshold` per specificare un valore diverso.

```
{
    "CollectionId": "MyCollection",
    "FaceId": "0b683aed-a0f1-48b2-9b5e-139e9cc2a757",
    "MaxFaces": 2,
    "FaceMatchThreshold": 99
}
```

SearchFaces risposta operativa

L'operazione restituisce una matrice di corrispondenze trovate e l'ID volto fornito come input.

```
{
    "SearchedFaceId": "7ecf8c19-5274-5917-9c91-1db9ae0449e2",
    "FaceMatches": [ list of face matches found ]
}
```

Per ogni corrispondenza di volto trovata, la risposta include la somiglianza e i metadati del volto, come nella risposta di esempio seguente:

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

Ricerca di un volto tramite immagine

È possibile utilizzare l'operazione SearchFacesByImage per cercare i volti in una raccolta che corrispondono al volto più grande in un'immagine fornita.

Per ulteriori informazioni, consulta Ricerca di volti e utenti all'interno di una raccolta.

Per cercare un volto in una raccolta utilizzando un'immagine (SDK)

1. Se non lo hai già fatto:

   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Carica un'immagine (contenente uno o più volti) nel bucket S3.

   Per le istruzioni, consulta Caricamento di oggetti in Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

3. Utilizzare i seguenti esempi per richiamare l'operazione `SearchFacesByImage`.

   Java

   In questo esempio vengono visualizzate le informazioni relative ai volti che corrispondono al volto più grande in un'immagine. L'esempio di codice specifica i parametri `FaceMatchThreshold` e `MaxFaces` per limitare i risultati restituiti nella risposta.

   Nel seguente esempio, modifica quanto segue: il valore di `collectionId` per la raccolta da ricercare, il valore di `bucket` nel bucket contenente l'immagine di input e il valore di `photo` nell'immagine di input.

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

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.FaceMatch;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.SearchFacesByImageRequest;
import com.amazonaws.services.rekognition.model.SearchFacesByImageResult;
import java.util.List;
import com.fasterxml.jackson.databind.ObjectMapper;


public class SearchFaceMatchingImageCollection {
    public static final String collectionId = "MyCollection";
    public static final String bucket = "bucket";
    public static final String photo = "input.jpg";

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

       AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

      ObjectMapper objectMapper = new ObjectMapper();

       // Get an image object from S3 bucket.
      Image image=new Image()
              .withS3Object(new S3Object()
```

```
                    .withBucket(bucket)
                    .withName(photo));

     // Search collection for faces similar to the largest face in the image.
     SearchFacesByImageRequest searchFacesByImageRequest = new
 SearchFacesByImageRequest()
             .withCollectionId(collectionId)
             .withImage(image)
             .withFaceMatchThreshold(70F)
             .withMaxFaces(2);

      SearchFacesByImageResult searchFacesByImageResult =
              rekognitionClient.searchFacesByImage(searchFacesByImageRequest);

      System.out.println("Faces matching largest face in image from" + photo);
     List < FaceMatch > faceImageMatches =
 searchFacesByImageResult.getFaceMatches();
     for (FaceMatch face: faceImageMatches) {
         System.out.println(objectMapper.writerWithDefaultPrettyPrinter()
                 .writeValueAsString(face));
        System.out.println();
     }
   }
}
```

Java V2

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
// 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

Questo AWS CLI comando visualizza l'output JSON per l'operazione `search-faces-by-image` CLI. Sostituisci il valore di `Bucket` con il bucket S3 utilizzato nella fase 2. Sostituisci il valore di `Name` con il nome del file delle immagini utilizzato nella fase 2. Sostituisci il valore di `collection-id` con il nome della raccolta in cui eseguire la ricerca. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

Se accedi alla CLI da un dispositivo Windows, usa le virgolette doppie anziché le virgolette singole ed evita le virgolette doppie interne tramite barra rovesciata (ovvero, \) per risolvere eventuali errori del parser che potresti riscontrare. Per un esempio, consulta quanto segue:

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

Python

In questo esempio vengono visualizzate le informazioni relative ai volti che corrispondono al volto più grande in un'immagine. L'esempio di codice specifica i parametri `FaceMatchThreshold` e `MaxFaces` per limitare i risultati restituiti nella risposta.

Nel seguente esempio, effettua queste operazioni: modifica il valore di `collectionId` per la raccolta da ricercare e sostituisci i valori di `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

In questo esempio vengono visualizzate le informazioni relative ai volti che corrispondono al volto più grande in un'immagine. L'esempio di codice specifica i parametri `FaceMatchThreshold` e `MaxFaces` per limitare i risultati restituiti nella risposta.

Nel seguente esempio, effettua queste operazioni: modifica il valore di `collectionId` per la raccolta da ricercare e sostituisci i valori di `bucket` e `photo` con i nomi del bucket Amazon S3 e dell'immagine utilizzati nella fase 2.

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

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

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

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        // Get an image object from S3 bucket.
        Image image = new Image()
        {
            S3Object = new S3Object()
            {
                Bucket = bucket,
                Name = photo
            }
        };

        SearchFacesByImageRequest searchFacesByImageRequest = new
 SearchFacesByImageRequest()
        {
            CollectionId = collectionId,
            Image = image,
            FaceMatchThreshold = 70F,
            MaxFaces = 2
        };

        SearchFacesByImageResponse searchFacesByImageResponse =
 rekognitionClient.SearchFacesByImage(searchFacesByImageRequest);

        Console.WriteLine("Faces matching largest face in image from " + photo);
        foreach (FaceMatch face in searchFacesByImageResponse.FaceMatches)
            Console.WriteLine("FaceId: " + face.Face.FaceId + ", Similarity: " +
 face.Similarity);
    }
}
```

SearchFacesByImage richiesta di operazione

I parametri di input per `SearchFacesImageByImage` sono la raccolta in cui eseguire la ricerca e l'ubicazione dell'immagine di origine. In questo esempio, l'immagine di origine è archiviata in un bucket Amazon S3 (`S3Object`). Sono specificati anche il numero massimo di volti da restituire (`Maxfaces`) e l'affidabilità minima che deve corrispondere a un volto affinché venga restituito (`FaceMatchThreshold`).

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "MaxFaces": 2,
    "FaceMatchThreshold": 99
}
```

SearchFacesByImage risposta operativa

Fornita un'immagine di input (.jpeg or .png), l'operazione rileva prima il volto nell'immagine di input, quindi ricerca nella raccolta specificata i volti simili.

> **Note**
>
> Se il servizio rileva più volti nell'immagine di input, per la ricerca nella raccolta utilizza il volto più grande rilevato.

L'operazione restituisce una matrice di corrispondenze di volti e le informazioni relative al volto di input. Le informazioni includono il riquadro di delimitazione, insieme al valore di affidabilità, che indica il livello di affidabilità relativo al fatto che il riquadro di delimitazione contenga un volto.

Per impostazione predefinita, `SearchFacesByImage` restituisce volti per cui l'algoritmo rileva una somiglianza superiore all'80%. La somiglianza indica il livello di corrispondenza del volto con il volto di input. È possibile utilizzare `FaceMatchThreshold` per specificare un valore diverso. Per ogni corrispondenza di volto trovata, la risposta include la somiglianza e i metadati del volto, come indicato nella risposta di esempio seguente:

```
{
    "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
}
```


Ricerca di utenti (ID volto/ ID utente)

È possibile utilizzare l'SearchUsersoperazione per cercare utenti in una raccolta specificata che corrispondono a un face ID o a un ID utente fornito. L'operazione elenca i risultati restituiti `UserIds` classificati in base al punteggio di somiglianza più alto rispetto a quello richiesto. UserMatchThreshold L'ID utente viene creato durante l' CreateUsers operazione. Per ulteriori informazioni, consulta Gestione degli utenti in una raccolta.

Per cercare utenti (SDK)

1. Se non lo hai già fatto:

a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Utilizzare i seguenti esempi per richiamare l'operazione `SearchUsers`.

Java

Questo esempio di Java cerca gli utenti di una raccolta utilizzando l'operazione `SearchUsers`.

```
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.UserMatch;
import com.amazonaws.services.rekognition.model.SearchUsersRequest;
import com.amazonaws.services.rekognition.model.SearchUsersResult;
import com.amazonaws.services.rekognition.model.UserMatch;

public class SearchUsers {
    //Replace collectionId and faceId with the values you want to use.

    public static final String collectionId = "MyCollection";
    public static final String faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";

    public static final String userd = 'demo-user';

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

        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

        // Search collection for faces matching the user id.
        SearchUsersRequest request = new SearchUsersRequest()
                .withCollectionId(collectionId)
                .withUserId(userId);

        SearchUsersResult result =
                rekognitionClient.searchUsers(request);
```

```
        System.out.println("Printing first search result with matched user and
similarity score");
        for (UserMatch match: result.getUserMatches()) {
            System.out.println(match.getUser().getUserId() + " with similarity
score " + match.getSimilarity());
        }

        // Search collection for faces matching the face id.
        SearchUsersRequest request1 = new SearchUsersRequest()
                .withCollectionId(collectionId)
                .withFaceId(faceId);

        SearchUsersResult result1 =
                rekognitionClient.searchUsers(request1);

        System.out.println("Printing second search result with matched user and
similarity score");
        for (UserMatch match: result1.getUserMatches()) {
            System.out.println(match.getUser().getUserId() + " with similarity
score " + match.getSimilarity());
        }
    }
```

AWS CLI

Questo AWS CLI comando cerca gli utenti in una raccolta con l'`SearchUsers`operazione.

```
aws rekognition search-users --face-id face-id --collection-id collection-id --
region region-name
```

Python

L'esempio seguente cerca gli utenti di una raccolta con l'operazione `SearchUsers`.

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

import boto3
from botocore.exceptions import ClientError
import logging

logger = logging.getLogger(__name__)
```

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

def search_users_by_face_id(collection_id, face_id):
    """
    SearchUsers operation with face ID provided as the search source

    :param collection_id: The ID of the collection where user and faces are
 stored.
    :param face_id: The ID of the face in the collection to search for.

    :return: response of SearchUsers API
    """
    logger.info(f'Searching for users using a face-id: {face_id}')
    try:
        response = client.search_users(
            CollectionId=collection_id,
            FaceId=face_id
        )
        print(f'- found {len(response["UserMatches"])} matches')
        print([f'- {x["User"]["UserId"]} - {x["Similarity"]}%' for x in
 response["UserMatches"]])
    except ClientError:
        logger.exception(f'Failed to perform SearchUsers with given face id:
 {face_id}')
        raise
    else:
        print(response)
        return response

def search_users_by_user_id(collection_id, user_id):
    """
    SearchUsers operation with user ID provided as the search source

    :param collection_id: The ID of the collection where user and faces are
 stored.
    :param user_id: The ID of the user in the collection to search for.

    :return: response of SearchUsers API
    """
    logger.info(f'Searching for users using a user-id: {user_id}')
    try:
        response = client.search_users(
            CollectionId=collection_id,
```

```
            UserId=user_id
        )
        print(f'- found {len(response["UserMatches"])} matches')
        print([f'- {x["User"]["UserId"]} - {x["Similarity"]}%' for x in
 response["UserMatches"]])
    except ClientError:
        logger.exception(f'Failed to perform SearchUsers with given face id:
 {user_id}')
        raise
    else:
        print(response)
        return response

def main():
    collection_id = "collection-id"
    user_id = "user-id"
    face_id = "face_id"
    search_users_by_face_id(collection_id, face_id)
    search_users_by_user_id(collection_id, user_id)

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

SearchUsers richiesta di operazione

Dato un FaceID o UserID, SearchUsers cerca le corrispondenze degli utenti nel CollectionID specificato. Per impostazione predefinita, SearchUsers restituisce l'utente IDs per il quale il punteggio di somiglianza è superiore all'80%. La somiglianza indica quanto lo UserID corrisponda al FaceID o allo UserID forniti. Se IDs vengono restituiti più utenti, questi vengono elencati in ordine dal punteggio di somiglianza più alto al più basso. Facoltativamente, è possibile utilizzare il UserMatchThreshold per specificare un valore diverso. Per ulteriori informazioni, consulta Gestione degli utenti in una raccolta.

Di seguito è riportato un esempio di SearchUsers richiesta che utilizza`UserId`:

```
{
   "CollectionId": "MyCollection",
   "UserId": "demoUser1",
   "MaxUsers": 2,
   "UserMatchThreshold": 99
```

```
}
```

Di seguito è riportato un esempio di SearchUsers richiesta che utilizza`FaceId`:

```
{
    "CollectionId": "MyCollection",
    "FaceId": "bff43c40-cfa7-4b94-bed8-8a08b2205107",
    "MaxUsers": 2,
    "UserMatchThreshold": 99
}
```

SearchUsers risposta all'operazione

Se si esegue la ricerca con a`FaceId`, la risposta `FaceId` per SearchUsers include il for`SearchedFace`, nonché un elenco di `UserMatches` `UserId` e `UserStatus` per ogni utente.

```
{
    "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"
}
```

Se si esegue una ricerca con un`UserId`, la risposta per SearchUsers include l'elemento `UserId` for the`SearchedUser`, oltre agli altri elementi di risposta.

```
{
    "SearchedUser": {
        "UserId": "demoUser1"
    },
    "UserMatches": [
        {
            "User": {
                "UserId": "demoUser2",
                "UserStatus": "ACTIVE"
            },
            "Similarity": 99.97946166992188
        }
    ],
    "FaceModelVersion": "6"
}
```

Ricerca di utenti (immagine)

`SearchUsersByImage` ricerca nel CollectionID specificato gli utenti in una raccolta che corrispondano al volto più grande rilevato in un'immagine fornita. Per impostazione predefinita, SearchUsersByImage restituisce Utente IDs per il quale il punteggio di somiglianza è superiore all'80%. La somiglianza indica quanto lo UserID corrisponda al volto più grande rilevato nell'immagine fornita. Se IDs vengono restituiti più utenti, questi vengono elencati in ordine dal punteggio di somiglianza più alto al più basso. Facoltativamente, è possibile utilizzare il UserMatchThreshold per specificare un valore diverso. Per ulteriori informazioni, consulta the section called “Gestione degli utenti in una raccolta”.

Per cercare gli utenti per immagine (SDK)

1. Se non lo hai già fatto:

a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Utilizzare i seguenti esempi per richiamare l'operazione `SearchUsersByImage`.

Java

Questo esempio di Java cerca gli utenti di una raccolta in base a un'immagine di input, utilizzando l'operazione `SearchUsersByImage`.

```
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.SearchUsersByImageRequest;
import com.amazonaws.services.rekognition.model.SearchUsersByImageResult;
import com.amazonaws.services.rekognition.model.UserMatch;


public class SearchUsersByImage {
    //Replace bucket, collectionId and photo with your values.
    public static final String collectionId = "MyCollection";
    public static final String s3Bucket = "bucket";
    public static final String s3PhotoFileKey = "input.jpg";

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

        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();


        // Get an image object from S3 bucket.
        Image image = new Image()
                .withS3Object(new S3Object()
                        .withBucket(s3Bucket)
                        .withName(s3PhotoFileKey));

        // Search collection for users similar to the largest face in the image.
        SearchUsersByImageRequest request = new SearchUsersByImageRequest()
```

```
                .withCollectionId(collectionId)
                .withImage(image)
                .withUserMatchThreshold(70F)
                .withMaxUsers(2);

        SearchUsersByImageResult result =
                rekognitionClient.searchUsersByImage(request);

        System.out.println("Printing search result with matched user and
 similarity score");
        for (UserMatch match: result.getUserMatches()) {
            System.out.println(match.getUser().getUserId() + " with similarity
 score " + match.getSimilarity());
        }
    }
}
```

AWS CLI

Questo AWS CLI comando cerca gli utenti in una raccolta in base a un'immagine di input, con l'`SearchUsersByImage`operazione.

```
aws rekognition search-users-by-image --image '{"S3Object":
{"Bucket":"s3BucketName","Name":"file-name"}}' --collection-id MyCollectionId --
region region-name
```

Python

L'esempio seguente cerca gli utenti di una raccolta in base a un'immagine di input, con l'operazione `SearchUsersByImage`.

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

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

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
```

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

def load_image(file_name):
    """
    helper function to load the image for indexFaces call from local disk

    :param image_file_name: The image file location that will be used by
 indexFaces call.
    :return: The Image in bytes
    """
    print(f'- loading image: {file_name}')
    with open(file_name, 'rb') as file:
        return {'Bytes': file.read()}

def search_users_by_image(collection_id, image_file):
    """
    SearchUsersByImage operation with user ID provided as the search source

    :param collection_id: The ID of the collection where user and faces are
 stored.
    :param image_file: The image that contains the reference face to search
 for.

    :return: response of SearchUsersByImage API
    """
    logger.info(f'Searching for users using an image: {image_file}')
    try:
        response = client.search_users_by_image(
            CollectionId=collection_id,
            Image=load_image(image_file)
        )
        print(f'- found {len(response["UserMatches"])} matches')
        print([f'- {x["User"]["UserId"]} - {x["Similarity"]}%' for x in
 response["UserMatches"]])
    except ClientError:
        logger.exception(f'Failed to perform SearchUsersByImage with given
 image: {image_file}')
        raise
    else:
        print(response)
        return response

def main():
    collection_id = "collection-id"
```

```
    IMAGE_SEARCH_SOURCE = os.getcwd() + '/image_path'
    search_users_by_image(collection_id, IMAGE_SEARCH_SOURCE)

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

SearchUsersByImage richiesta di operazione

La richiesta per `SearchUsersByImage` include la raccolta in cui eseguire la ricerca e l'ubicazione dell'immagine di origine. In questo esempio, l'immagine di origine è archiviata in un bucket Amazon S3 (`S3Object`). Sono specificati anche il numero massimo di utenti da restituire (`MaxUsers`) e la confidenza minima da soddisfare affinché un volto venga restituito (`UserMatchThreshold`).

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "MaxUsers": 2,
    "UserMatchThreshold": 99
}
```

SearchUsersByImage risposta operativa

La risposta per `SearchUsersByImage` include un `FaceDetail` oggetto per`SearchedFace`, oltre a un elenco di UserMatches con `UserIdSimilarity`, e `UserStatus` per ciascuno. Se l'immagine di input conteneva più di un volto, UnsearchedFaces verrà restituito anche un elenco di.

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

Ricerca di volti in video archiviati

È possibile cercare volti in una raccolta che corrispondano ai volti delle persone rilevate in un video archiviato o un video in streaming. Questa sezione descrive la ricerca di volti in un video archiviato. Per informazioni sulla ricerca di volti in un video in streaming, consulta Utilizzo degli eventi video in streaming.

I volti cercati devono prima essere indicizzati in una raccolta utilizzando IndexFaces. Per ulteriori informazioni, consulta Aggiunta di volti a una raccolta.

La ricerca di volti di Video Amazon Rekognition segue lo stesso flusso di lavoro asincrono di altre operazioni Video Amazon Rekognition che analizzano i video archiviati in un bucket Amazon S3. Per iniziare a cercare i volti in un video archiviato, chiama StartFaceSearche fornisci l'ID della raccolta che desideri cercare. Amazon Rekognition per video pubblica lo stato di completamento dell'analisi video in un argomento Amazon Simple Notification Service (Amazon SNS). Se l'analisi video ha esito positivo, effettua la chiamata a GetFaceSearch per ottenere i risultati della ricerca. Per ulteriori informazioni su come avviare analisi video e ottenere i risultati, consultare Chiamata delle operazioni Video Amazon Rekognition.

La procedura seguente mostra come cercare volti in una raccolta che corrispondono ai volti di persone rilevate in un video. La procedura inoltre illustra come ottenere i dati di tracciamento per le persone con corrispondenza nel video. La procedura si espande nel codice in Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK), che utilizza una coda Amazon Simple Queue Service (Amazon SQS) per ottenere lo stato di completamento di una richiesta di analisi video.

Per cercare un video per la corrispondenza di volti (SDK)

1. Creare una raccolta.
2. Indicizzare un volto nella raccolta.

3. Eseguire Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK).
4. Aggiungere il seguente codice alla classe `VideoDetect` creata nella fase 3.

   Java

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

       //Face collection search in video
 ======================================================================
       private static void StartFaceSearchCollection(String bucket, String
 video, String collection) throws Exception{

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

        StartFaceSearchRequest req = new StartFaceSearchRequest()
                .withCollectionId(collection)
                .withVideo(new Video()
                        .withS3Object(new S3Object()
                                .withBucket(bucket)
                                .withName(video)))
                .withNotificationChannel(channel);



        StartFaceSearchResult startPersonCollectionSearchResult =
 rek.startFaceSearch(req);
        startJobId=startPersonCollectionSearchResult.getJobId();

    }

    //Face collection search in video
 ======================================================================
    private static void GetFaceSearchCollectionResults() throws Exception{

       GetFaceSearchResult faceSearchResult=null;
       int maxResults=10;
       String paginationToken=null;

       do {
```

```
            if (faceSearchResult !=null){
                paginationToken = faceSearchResult.getNextToken();
            }


            faceSearchResult  = rek.getFaceSearch(
                    new GetFaceSearchRequest()
                    .withJobId(startJobId)
                    .withMaxResults(maxResults)
                    .withNextToken(paginationToken)
                    .withSortBy(FaceSearchSortBy.TIMESTAMP)
                    );


            VideoMetadata videoMetaData=faceSearchResult.getVideoMetadata();

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


            //Show search results
            List<PersonMatch> matches=
                    faceSearchResult.getPersons();

            for (PersonMatch match: matches) {
                long milliSeconds=match.getTimestamp();
                System.out.print("Timestamp: " + Long.toString(milliSeconds));
                System.out.println(" Person number: " +
match.getPerson().getIndex());
                List <FaceMatch> faceMatches = match.getFaceMatches();
                if (faceMatches != null) {
                    System.out.println("Matches in collection...");
                    for (FaceMatch faceMatch: faceMatches){
                        Face face=faceMatch.getFace();
                        System.out.println("Face Id: "+ face.getFaceId());
                        System.out.println("Similarity: " +
faceMatch.getSimilarity().toString());
                        System.out.println();
                    }
```

```
                }
                System.out.println();
            }

            System.out.println();

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

    }
```

Nella funzione `main`, sostituisci le righe:

```
        StartLabelDetection(bucket, video);

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

con:

```
        String collection="collection";
        StartFaceSearchCollection(bucket, video, collection);

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

Java V2

Questo codice è tratto dall' GitHub archivio degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
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()
```

Nella funzione `main`, sostituisci le righe:

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

con:

```
    collection='tests'
    analyzer.StartFaceSearchCollection(collection)

    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetFaceSearchCollectionResults()
```

Se hai già eseguito un video di esempio diverso da Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK), il codice da sostituire potrebbe essere diverso.

5. Modificare il valore di `collection` nel nome della raccolta creata nella fase 1.

6. Eseguire il codice. Viene visualizzato un elenco di persone nel video i cui volti corrispondono a quelli della raccolta di input. Vengono inoltre visualizzati i dati di tracciamento per ogni persona con corrispondenza.

GetFaceSearch risposta operativa

Di seguito è riportata una risposta di esempio JSON da `GetFaceSearch`.

La risposta include una matrice di persone (`Persons`) rilevate nel video i cui volti corrispondono a un volto nella raccolta di input. Esiste un elemento della matrice, PersonMatch, per ogni corrispondenza della persona nel video. Ogni elemento `PersonMatch` include una matrice di corrispondenze di volti dalla raccolta di input, FaceMatch, informazioni sulla persona con corrispondenza, PersonDetail, e l'ora in cui è stata rilevata la corrispondenza della persona nel video.


```
{
    "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
}
```

```
}
```

Ricerca di volti in una raccolta in streaming video

È possibile utilizzare Video Amazon Rekognition per rilevare e riconoscere i volti da una raccolta in streaming video. Con Amazon Rekognition Video puoi creare uno stream processor CreateStreamProcessor() per avviare e gestire l'analisi dei video in streaming.

Per rilevare un volto noto in un flusso video (ricerca di volti), Video Amazon Rekognition utilizza Flusso di video Amazon Kinesis per ricevere ed elaborare un flusso video. I risultati delle analisi vengono generati da Video Amazon Rekognition a un flusso di dati Kinesis e quindi letti dall'applicazione client.

Per usare Video Amazon Rekognition con un video in streaming, l'applicazione deve implementare quanto segue:

- Un flusso video Kinesis per l'invio di video in streaming ad Video Amazon Rekognition. Per ulteriori informazioni, consulta la Guida per gli sviluppatori di Flusso di video Amazon Kinesis.
- Un elaboratore di flussi Video Amazon Rekognition per gestire l'analisi del video in streaming. Per ulteriori informazioni, consulta Panoramica delle operazioni del processore di streaming Video Amazon Rekognition.
- Un utente di Kinesis Data Stream per leggere i risultati dell'analisi che Video Amazon Rekognition invia al flusso di dati Kinesis. Per ulteriori informazioni, consulta Utenti di Kinesis Data Streams.

Questa sezione contiene informazioni sulla scrittura di un'applicazione che crea lo il flusso video Kinesis e altre risorse necessarie, trasmette video in Video Amazon Rekognition e riceve i risultati dell'analisi.

Argomenti

- Configurazione delle risorse Video Amazon Rekognition e Amazon Kinesis
- Ricerca di volti in un video in streaming
- Streaming tramite un GStreamer plugin
- Risoluzione dei problemi dello streaming di video

Configurazione delle risorse Video Amazon Rekognition e Amazon Kinesis

Le seguenti procedure descrivono i passaggi da seguire per effettuare il provisioning del flusso video Kinesis e altre risorse utilizzate per riconoscere i volti in un video in streaming.

Prerequisiti

Per eseguire questa procedura, è necessario aver installato il. AWS SDK per Java Per ulteriori informazioni, consulta Nozioni di base su Amazon Rekognition. Il dispositivo Account AWS che usi deve disporre delle autorizzazioni di accesso all'API Amazon Rekognition. Per ulteriori informazioni, consulta la sezione Operazioni definite da Amazon Rekognition nella Guida per l'utente IAM.

Per riconoscere volti in un flusso video (SDK AWS)

1. Se non l'hai già fatto, crea un ruolo di servizio IAM per consentire ad Video Amazon Rekognition di accedere ai tuoi flussi video Kinesis e ai tuoi flussi di dati Kinesis. Prendi nota dell'ARN. Per ulteriori informazioni, consulta Concedere l'accesso agli stream utilizzando AmazonRekognitionServiceRole.
2. Creare una raccolta e prendere nota dell'identificatore di raccolta utilizzato.
3. Indicizzare i volti che si desidera cercare nella raccolta creata alla fase 2.
4. Crea un flusso video Kinesis e prendi nota del relativo nome della risorsa Amazon (ARN).
5. Crea un flusso di dati Kinesis. Anteponi il nome dello stream con AmazonRekognitione annota l'ARN dello stream.

Puoi quindi creare l'elaboratore di flussi per la ricerca di volti e avviarlo usando il nome dell'elaboratore di flussi che hai scelto.

> **Note**
>
> È consigliabile avviare l'elaboratore di flussi solo dopo aver verificato la possibilità di importare contenuti multimediali nel flusso video Kinesis.

Streaming di video in Video Amazon Rekognition

Per lo streaming di video in Video Amazon Rekognition, utilizzerai l'SDK di Flusso di video Amazon Kinesis per creare e utilizzare un flusso video Kinesis. L'operazione `PutMedia` scrive

frammenti di dati video in un flusso video Kinesis utilizzato da Video Amazon Rekognition. Ogni frammento di dati video dura in genere da 2 a 10 secondi e contiene una sequenza compatta di fotogrammi video. Video Amazon Rekognition supporta video con codifica H.264, che possono avere tre tipi di fotogrammi (I, B e P). Per ulteriori informazioni, consulta Interframe. Il primo fotogramma nel frammento deve essere un fotogramma I. Un fotogramma I può essere decodificato indipendentemente da qualsiasi altro fotogramma.

Man mano che i dati video giungono nel flusso video Kinesis, Kinesis Video Streams assegna un numero univoco al frammento. Per un esempio, vedi PutMedia Esempio di API.

- Se stai eseguendo lo streaming da una sorgente codificata Matroska (MKV), usa l'PutMediaoperazione per trasmettere il video sorgente nel flusso video Kinesis che hai creato. Per ulteriori informazioni, consulta Esempio di API. PutMedia
- Se stai eseguendo lo streaming dalla fotocamera di un dispositivo, consulta Streaming tramite un GStreamer plugin.

Concessione ad Video Amazon Rekognition dell'accesso alle risorse

Utilizzi un ruolo di servizio AWS Identity and Access Management (IAM) per consentire ad Amazon Rekognition Video l'accesso in lettura ai flussi video Kinesis. Se utilizzi un elaboratore di flussi per la ricerca di volti, utilizzi un ruolo di servizio IAM per consentire ad Video Amazon Rekognition l'accesso in scrittura ai flussi di dati Kinesis. Se utilizzi un elaboratore di flussi per il monitoraggio della sicurezza, utilizzi i ruoli IAM per consentire ad Video Amazon Rekognition di accedere al tuo bucket Amazon S3 e a un argomento di Amazon SNS.

Concessione dell'accesso agli elaboratori di flussi per la ricerca di volti

Puoi creare una policy di autorizzazione che consenta ad Video Amazon Rekognition di accedere ai singoli flussi video Kinesis e ai flussi di dati Kinesis.

Per consentire ad Video Amazon Rekognition di accedere a un elaboratore di flussi per la ricerca di volti

1. Crea una nuova policy di autorizzazione con l'editor delle policy JSON IAM e utilizza la policy seguente. Sostituisci `video-arn` con l'ARN del flusso video Kinesis desiderato. Se utilizzi un elaboratore di flussi per la ricerca di volti, sostituisci `data-arn` con l'ARN del flusso di dati Kinesis desiderato.

   ```
   {
   ```

```
    "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. Crea un ruolo di servizio IAM o aggiornane uno esistente. Per creare il ruolo di servizio IAM, utilizza le informazioni seguenti:

    1. Scegli Rekognition come nome del servizio.
    2. Scegli Rekognition come caso d'uso del ruolo del servizio.
    3. Collega la policy di autorizzazione creata nella fase 1.

3. Prendere nota dell'ARN del ruolo del servizio. necessario per avviare le operazioni di analisi video.

Concedere l'accesso agli stream utilizzando AmazonRekognitionServiceRole

Come opzione alternativa per configurare l'accesso ai flussi video e ai flussi di dati di Kinesis, puoi utilizzare la policy di autorizzazione `AmazonRekognitionServiceRole`. IAM fornisce il caso d'uso del ruolo di servizio Rekognition che, se usato con la policy di autorizzazione `AmazonRekognitionServiceRole`, è in grado di scrivere su più flussi di dati Kinesis e leggere da tutti i flussi video Kinesis. Per consentire ad Amazon Rekognition Video l'accesso in scrittura a più flussi di dati Kinesis, puoi anteporre ai nomi dei flussi di dati Kinesis —ad esempio,. AmazonRekognition`AmazonRekognitionMyDataStreamName`

Per consentire ad Video Amazon Rekognition di accedere al flusso video Kinesis e al flusso di dati Kinesis

1. Crea un ruolo di servizio IAM. Per creare il ruolo di servizio IAM, utilizza le informazioni seguenti:
    1. Scegli Rekognition come nome del servizio.
    2. Scegli Rekognition come caso d'uso del ruolo del servizio.
    3. Scegli la politica di AmazonRekognitionServiceRoleautorizzazione, che consente ad Amazon Rekognition Video di accedere in scrittura ai flussi di dati Kinesis con AmazonRekognitionprefisso e accesso in lettura a tutti i tuoi flussi video Kinesis.
2. Per garantire la tua Account AWS sicurezza, limita l'ambito di accesso di Rekognition alle sole risorse che stai utilizzando. Questo può essere fatto associando una policy di attendibilità al ruolo di servizio IAM. Per informazioni su come fare, consulta Prevenzione del problema "confused deputy" tra servizi.
3. Prendere nota del valore nome della risorsa Amazon (ARN) del ruolo di servizio, necessario per avviare le operazioni di analisi video.

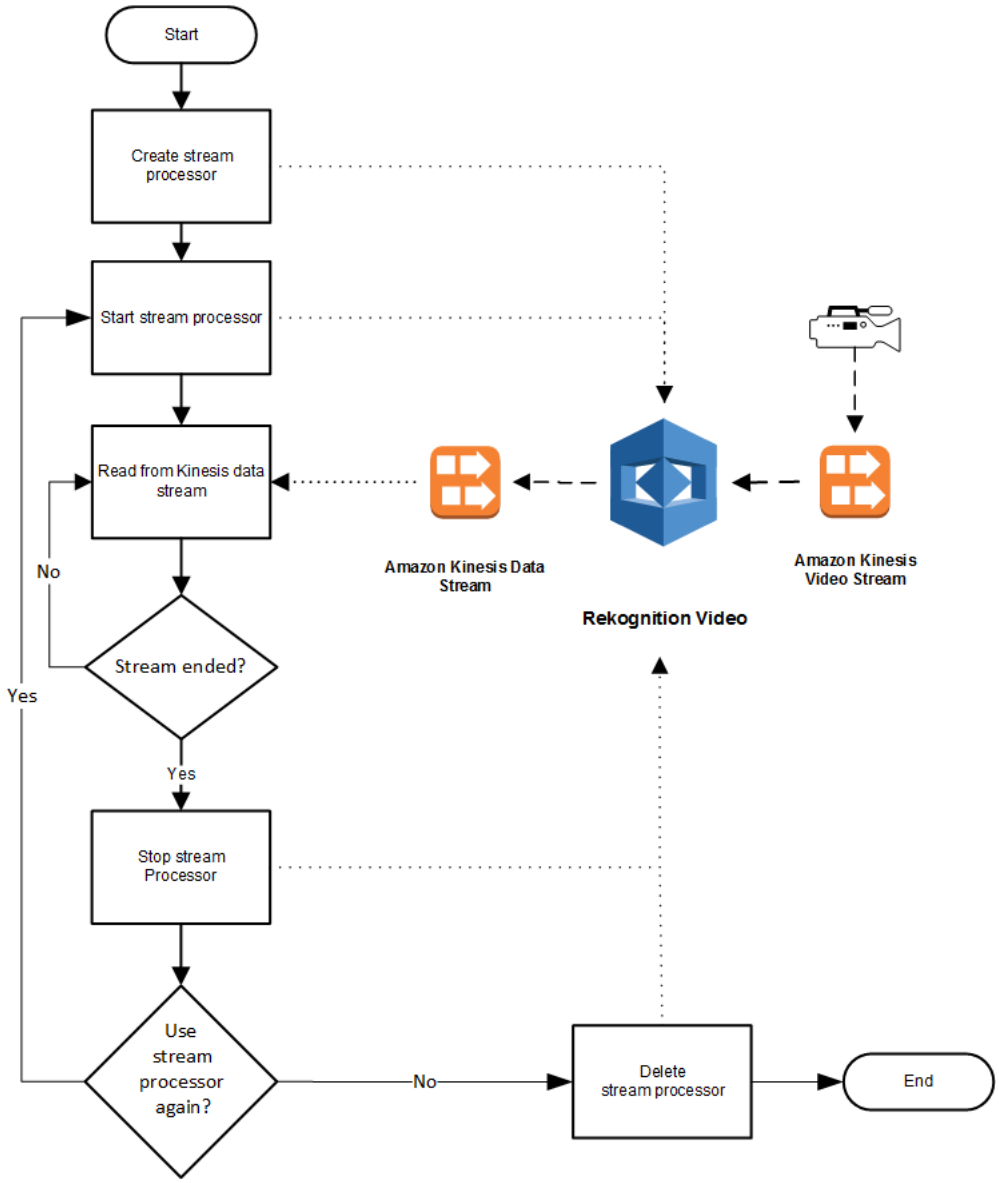
Ricerca di volti in un video in streaming

Video Amazon Rekognition può cercare i volti in una raccolta che corrispondono a quelli rilevati in un video in streaming. Per ulteriori informazioni sulle raccolte, consulta Ricerca di volti in una raccolta.

Argomenti

- Creazione dell'elaboratore di flussi per la ricerca di volti Video Amazon Rekognition
- Avvio dell'elaboratore di flussi per la ricerca di volti Video Amazon Rekognition
- Utilizzo dell'elaboratore di flussi per la ricerca di volti (esempio Java V2)
- Utilizzo dell'elaboratore di flussi per la ricerca di volti (esempio Java V1)
- Lettura dei risultati delle analisi di video in streaming
- Visualizzazione locale dei risultati di Rekognition con Kinesis Video Streams
- Comprendere il frame record JSON di riconoscimento facciale Kinesis

Il seguente diagramma mostra in che modo Video Amazon Rekognition rileva e riconosce i volti in un video in streaming.



Creazione dell'elaboratore di flussi per la ricerca di volti Video Amazon Rekognition

Prima di poter analizzare un video in streaming, devi creare uno stream processor Amazon Rekognition Video (). [CreateStreamProcessor](#) L'elaboratore di flussi contiene informazioni sul flusso di dati Kinesis e sul flusso video Kinesis. Contiene inoltre l'identificatore della raccolta che include i volti da riconoscere nel video in streaming di input. È inoltre necessario specificare un nome per l'elaboratore di flussi. Di seguito è riportato un esempio di JSON per la richiesta `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
                }
        }
}
```

Di seguito è riportata una risposta di esempio da `CreateStreamProcessor`.

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

Avvio dell'elaboratore di flussi per la ricerca di volti Video Amazon Rekognition

Per iniziare ad analizzare il video in streaming, chiamare StartStreamProcessor con il nome dell'elaboratore di flussi specificato in `CreateStreamProcessor`. Di seguito è riportato un esempio di JSON per la richiesta `StartStreamProcessor`.

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

All'avvio dell'elaboratore di flussi viene restituita una risposta HTTP 200, insieme a un corpo JSON vuoto.

Utilizzo dell'elaboratore di flussi per la ricerca di volti (esempio Java V2)

Il codice di esempio seguente mostra come chiamare varie operazioni dello stream processor, ad esempio CreateStreamProcessore StartStreamProcessor, utilizzando l' AWS SDK for Java versione 2.

Questo codice è tratto dal repository degli esempi GitHub di AWS Documentation SDK. Guarda l'esempio completo qui.

```
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.");
    }
}
```

Utilizzo dell'elaboratore di flussi per la ricerca di volti (esempio Java V1)

Il codice di esempio seguente mostra come chiamare varie operazioni dello stream processor, ad esempio CreateStreamProcessorand StartStreamProcessor, utilizzando Java V1. L'esempio include una classe stream processor manager (StreamManager) che fornisce metodi per chiamare le operazioni dello stream processor. La classe starter (Starter) crea un StreamManager oggetto e chiama varie operazioni.

Per configurare l'esempio:

1. Impostare i valori dei campi membro della classe Starter sui valori desiderati.
2. Nella funzione classe Starter `main`, rimuovere il commento dalla chiamata della funzione desiderata.

Classe Starter

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

// Starter class. Use to create a StreamManager class
// and call stream processor operations.
package com.amazonaws.samples;
import com.amazonaws.samples.*;

public class Starter {

 public static void main(String[] args) {


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

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

StreamManager classe

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

Lettura dei risultati delle analisi di video in streaming

È possibile utilizzare la libreria del client di Flusso di dati Amazon Kinesis per utilizzare i risultati delle analisi inviati al flusso di output di Flusso di dati Amazon Kinesis. Per ulteriori informazioni, consulta Lettura di dati da un flusso di dati Kinesis. Video Amazon Rekognition posiziona un record di fotogramma JSON per ogni fotogramma analizzato nel flusso di output di Kinesis. Video Amazon Rekognition non analizza tutti i fotogrammi che gli vengono trasmessi tramite il flusso video Kinesis.

Un record di fotogramma inviato a un flusso di dati Kinesis contiene informazioni sul frammento del flusso video Kinesis in cui si trova il fotogramma, la sua posizione nel frammento e i volti riconosciuti al suo interno. Include inoltre le informazioni sullo stato dell'elaboratore di flussi. Per ulteriori informazioni, consulta Comprendere il frame record JSON di riconoscimento facciale Kinesis.

La libreria parser di Flusso di video Amazon Kinesis contiene test di esempio che utilizzano i risultati di Video Amazon Rekognition e li integra con il flusso video Kinesis originale. Per ulteriori informazioni, consulta Visualizzazione locale dei risultati di Rekognition con Kinesis Video Streams.

Video Amazon Rekognition trasmette le informazioni delle analisi di Video Amazon Rekognition al flusso di dati Kinesis. Di seguito è riportato un esempio di JSON per un singolo record.

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

Nell'esempio JSON, tenere presente quanto segue:

- InputInformation— Informazioni sullo stream video Kinesis utilizzato per lo streaming di video in Amazon Rekognition Video. Per ulteriori informazioni, consulta InputInformation.

- StreamProcessorInformation— Informazioni sullo stato dello stream processor Amazon Rekognition Video. L'unico valore possibile per il campo `Status` è RUNNING. Per ulteriori informazioni, consulta StreamProcessorInformation.
- FaceSearchResponse— Contiene informazioni sui volti nel video in streaming che corrispondono ai volti nella raccolta di input. FaceSearchResponsecontiene un DetectedFace oggetto, ossia un volto rilevato nel fotogramma video analizzato. Per ogni volto rilevato, la matrice `MatchedFaces` contiene una serie di oggetti di volti corrispondenti (MatchedFace) individuati nella raccolta di input, oltre a un punteggio di somiglianza.

Mappatura del flusso video Kinesis al flusso di dati Kinesis

È possibile mappare i fotogrammi del flusso video Kinesis ai fotogrammi analizzati inviati al flusso di dati Kinesis. Ad esempio, durante la visualizzazione di un video in streaming, è possibile visualizzare riquadri intorno ai volti delle persone riconosciute. Le coordinate del riquadro di delimitazione vengono inviate come parte del record di riconoscimento del volto Kinesis al flusso di dati Kinesis. Per visualizzare correttamente il riquadro di delimitazione, è necessario mappare le informazioni di data e ora inviate al record di riconoscimento del volto Kinesis ai corrispondenti fotogrammi nel flusso video Kinesis di origine.

La tecnica utilizzata per mappare il flusso video Kinesis al flusso di dati Kinesis varia a seconda che si riproducano in streaming file multimediali live (ad esempio, un video in streaming live) oppure file multimediali archiviati (ad esempio, un video memorizzato).

Mappatura in caso di streaming di file multimediali live

Per mappare un fotogramma del flusso video Kinesis a un fotogramma del flusso di dati Kinesis

1. Imposta il parametro `FragmentTimeCodeType` di input dell'PutMediaoperazione su`RELATIVE`.
2. Chiamare `PutMedia` per trasmettere contenuti multimediali live nel flusso video Kinesis.
3. Quando si riceve un record di riconoscimento del volto Kinesis dal flusso di dati Kinesis, archiviare i valori di `ProducerTimestamp` e `FrameOffsetInSeconds` dal campo KinesisVideo.
4. Calcolare il timestamp corrispondente al fotogramma del flusso video Kinesis aggiungendo insieme i valori dei campi `ProducerTimestamp` e `FrameOffsetInSeconds`.

Mappatura in caso di streaming di file multimediali archiviati

Per mappare un fotogramma del flusso video Kinesis a un fotogramma del flusso di dati Kinesis

1. Chiama PutMediaper inviare contenuti multimediali archiviati nel flusso video Kinesis.
2. Quando si riceve un oggetto `Acknowledgement` dalla risposta dell'operazione `PutMedia`, archiviare il valore del campo `FragmentNumber` dal campo Payload. `FragmentNumber` è il numero di frammento per il cluster MKV.
3. Quando si riceve un record di riconoscimento del volto Kinesis dal flusso di dati Kinesis, archiviare il valore del campo `FrameOffsetInSeconds` dal campo KinesisVideo.
4. Calcolare la mappatura utilizzando i valori `FrameOffsetInSeconds` e `FragmentNumber` archiviati nelle fasi 2 e 3. `FrameOffsetInSeconds` è l'offset nel frammento con il valore `FragmentNumber` specifico inviato al flusso di dati Amazon Kinesis. Per ulteriori informazioni su come ottenere i fotogrammi video per un numero di frammento specifico, consulta File multimediali archiviati di Flusso di video Amazon Kinesis.

Visualizzazione locale dei risultati di Rekognition con Kinesis Video Streams

Puoi vedere i risultati di Amazon Rekognition Video visualizzati nel tuo feed da Amazon Kinesis Video Streams utilizzando i test di esempio di Amazon Kinesis Video Streams Parser Library forniti all'indirizzo - Rekognition Examples. KinesisVideo `KinesisVideoRekognitionIntegrationExample`Visualizza dei riquadri di delimitazione sui volti rilevati e rende il video localmente. JFrame Questo processo presuppone che tu abbia collegato correttamente un ingresso multimediale dalla fotocamera di un dispositivo a un flusso video Kinesis e avviato un elaboratore di flussi Amazon Rekognition. Per ulteriori informazioni, consulta Streaming tramite un GStreamer plugin.

Fase 1: installazione della libreria parser Kinesis Video Streams

Per creare una directory e scaricare il repository Github, esegui il seguente comando:

```
$ git clone https://github.com/aws/amazon-kinesis-video-streams-parser-library.git
```

Vai alla directory della libreria ed esegui il seguente comando Maven per eseguire un'installazione pulita:

```
$ mvn clean install
```

Fase 2: configurazione del test di esempio di integrazione tra Kinesis Video Streams e Rekognition

Apri il file `KinesisVideoRekognitionIntegrationExampleTest.java`. Rimuovi `@Ignore` subito dopo l'intestazione della classe. Compila i campi dati con le informazioni delle tue risorse Amazon Kinesis e Amazon Rekognition. Per ulteriori informazioni, consulta Configurazione delle risorse Video Amazon Rekognition e Amazon Kinesis. Se stai trasmettendo video in streaming al flusso video Kinesis, rimuovi il parametro `inputStream`.

Consulta l'esempio di codice seguente:

```
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();
```

Fase 3: esecuzione del test di esempio di integrazione tra Kinesis Video Streams e Rekognition

Assicurati che il tuo flusso video Kinesis riceva input multimediali se stai trasmettendo ad esso e inizia ad analizzare il flusso con un elaboratore di flussi Video Amazon Rekognition in esecuzione.

Per ulteriori informazioni, consulta Panoramica delle operazioni del processore di streaming Video Amazon Rekognition. Esegui la `KinesisVideoRekognitionIntegrationExampleTest` lezione come test. JUnit Dopo un breve ritardo, si apre una nuova finestra con un feed video del flusso video Kinesis con riquadri di delimitazione disegnati sui volti rilevati.

> **Note**
>
> I volti della raccolta utilizzata in questo esempio devono avere un ID immagine esterno (il nome del file) specificato in questo formato affinché le etichette dei riquadri di delimitazione visualizzino un testo significativo: PersonName 1-Trusted, PersonName 2-Intruder, 3-Neutral, ecc. PersonName Le etichette possono anche essere codificate a colori e sono personalizzabili nel file.java. FaceType

Comprendere il frame record JSON di riconoscimento facciale Kinesis

Video Amazon Rekognition può riconoscere i volti in un video in streaming. Per ogni fotogramma analizzato, Video Amazon Rekognition invia un record di fotogramma JSON a un flusso di dati Kinesis. Video Amazon Rekognition non analizza tutti i fotogrammi che gli vengono trasmessi tramite il flusso video Kinesis.

Il record di fotogramma JSON contiene informazioni sul flusso di input e output, sullo stato dell'elaboratore di flussi e informazioni sui volti riconosciuti nel fotogramma analizzato. Questa sezione contiene informazioni di riferimento per il record di fotogramma JSON.

Di seguito è riportata la sintassi JSON per un record di flusso di dati Kinesis. Per ulteriori informazioni, consulta Utilizzo degli eventi video in streaming.

> **Note**
>
> L'API Video Amazon Rekognition funziona confrontando i volti nel flusso di input con una raccolta di volti e restituendo le corrispondenze trovate più vicine, insieme a un punteggio di somiglianza.

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

Record JSON

Il record JSON include informazioni su un fotogramma elaborato da Video Amazon Rekognition. Il record include informazioni sul video in streaming, lo stato del fotogramma analizzato e informazioni sui volti riconosciuti nel fotogramma.

InputInformation

Informazioni sul flusso video Kinesis utilizzato per lo streaming di video in Video Amazon Rekognition.

Tipo: oggetto InputInformation

StreamProcessorInformation

Informazioni sull'elaboratore di flussi di Video Amazon Rekognition. Sono incluse le informazioni sullo stato attuale dell'elaboratore di flussi.

Tipo: oggetto StreamProcessorInformation

FaceSearchResponse

Informazioni sui volti rilevati in un fotogramma video in streaming e i volti corrispondenti trovati nella raccolta di input.

Tipo: array di oggetti FaceSearchResponse

InputInformation

Informazioni su un flusso video di origine utilizzato da Video Amazon Rekognition. Per ulteriori informazioni, consulta Utilizzo degli eventi video in streaming.

KinesisVideo

Tipo: oggetto KinesisVideo

KinesisVideo

Informazioni sul flusso video Kinesis che esegue lo streaming del video di origine in Video Amazon Rekognition. Per ulteriori informazioni, consulta Utilizzo degli eventi video in streaming.

StreamArn

Il nome della risorsa Amazon (ARN) del flusso video Kinesis.

Tipo: stringa

FragmentNumber

Frammento del video in streaming che contiene il fotogramma rappresentato da questo record.

Tipo: stringa

ProducerTimestamp

Time stamp Unix lato produttore del frammento. Per ulteriori informazioni, consulta PutMedia.

Tipo: numero

ServerTimestamp

Time stamp Unix lato server del frammento. Per ulteriori informazioni, consulta PutMedia.

Tipo: numero

FrameOffsetInSeconds

Offset del fotogramma (in secondi) all'interno del frammento.

Tipo: numero

StreamProcessorInformation

Informazioni di stato sull'elaboratore di flussi.

Stato

Stato corrente dell'elaboratore di flussi. Un valore possibile è RUNNING.

Tipo: stringa

FaceSearchResponse

Informazioni su un volto rilevato in un fotogramma video in streaming e sui volti in una raccolta che corrispondono al volto rilevato. La raccolta viene specificata in una chiamata a CreateStreamProcessor. Per ulteriori informazioni, consulta Utilizzo degli eventi video in streaming.

DetectedFace

I dettagli di un volto rilevato in un fotogramma video analizzato.

Tipo: oggetto DetectedFace

MatchedFaces

Una matrice di dettagli di volti per i volti in una raccolta corrispondente al volto rilevato in `DetectedFace`.

Tipo: array di oggetti MatchedFace

DetectedFace

Informazioni su un volto rilevato in un fotogramma di un video in streaming. I volti corrispondenti nella raccolta di input sono disponibili nel campo oggetto MatchedFace.

BoundingBox

Coordinate della cornice per un volto rilevato in un fotogramma di un video analizzato. L' BoundingBox oggetto ha le stesse proprietà dell' BoundingBox oggetto utilizzato per l'analisi delle immagini.

Tipo: oggetto BoundingBox

Confidence

Il livello di confidenza (1-100) che Video Amazon Rekognition ha sul fatto che il volto rilevato sia effettivamente un volto. 1 è la confidenza più bassa, 100 è la più alta.

Tipo: numero

Landmarks

Matrice di punti di riferimento del volto.

Tipo: matrice di oggetti Landmark

Posa

Indica la posa del volto definita da beccheggio, rollio e imbardata.

Tipo: oggetto Pose

Qualità

Identifica la luminosità e la nitidezza dell'immagine del volto.

Tipo: oggetto ImageQuality

MatchedFace

Informazioni su un volto che corrisponde a un volto rilevato in un fotogramma video analizzato.

Face

Informazioni di corrispondenza per un volto nella raccolta di input che corrisponde al volto rilevato nell'oggetto DetectedFace.

Tipo: oggetto Face

Somiglianza

Il livello di confidenza (1-100) sul fatto che i volti corrispondano. 1 è la confidenza più bassa, 100 è la più alta.

Tipo: numero

Streaming tramite un GStreamer plugin

Video Amazon Rekognition può analizzare un video in streaming live dalla fotocamera di un dispositivo. Per accedere all'input multimediale da una sorgente del dispositivo, è necessario installarlo GStreamer. GStreamer è un software framework multimediale di terze parti che collega fonti multimediali e strumenti di elaborazione nelle pipeline del flusso di lavoro. È inoltre necessario

installare il plugin Producer di Flusso di video Amazon Kinesis per Gstreamer. Questo processo presuppone che tu abbia configurato correttamente le risorse Video Amazon Rekognition e Amazon Kinesis. Per ulteriori informazioni, consulta Configurazione delle risorse Video Amazon Rekognition e Amazon Kinesis.

Fase 1: installare Gstreamer

Scarica e installa Gstreamer, un software di piattaforma multimediale di terze parti. Puoi utilizzare un software di gestione dei pacchetti come Homebrew (Gstreamer su Homebrew) o scaricarlo direttamente dal sito Web di Freedesktop.

Verifica la corretta installazione di Gstreamer lanciando un feed video con un'origine di test dal tuo terminale della riga di comando.

```
$ gst-launch-1.0 videotestsrc ! autovideosink
```

Fase 2: installare il plugin Producer di Kinesis Video Streams

In questa sezione, scaricherai la libreria Producer di Flusso di video Amazon Kinesis e installerai il plugin Gstreamer di Kinesis Video Streams.

Crea una directory e clona il codice di origine dal repository GitHub. Assicurati di includere il parametro `--recursive`.

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

Segui le istruzioni fornite dalla libreria per configurare e compilare il progetto. Assicurati di utilizzare i comandi specifici per la piattaforma per il sistema operativo in uso. Usa il parametro `-DBUILD_GSTREAMER_PLUGIN=ON` quando esegui `cmake` per installare il plugin Gstreamer di Kinesis Video Streams. Questo progetto richiede i seguenti pacchetti aggiuntivi inclusi nell'installazione: GCC o Clang, Curl, Openssl e Log4cplus. Se la compilazione non riesce a causa di un pacchetto mancante, verifica che il pacchetto sia installato e presente nel tuo PATH. Se riscontri l'errore "impossibile eseguire il programma compilato in C" durante la compilazione, esegui nuovamente il comando di compilazione. A volte, il compilatore C corretto non viene trovato.

Verifica l'installazione del plugin Kinesis Video Streams eseguendo il seguente comando.

```
$ gst-inspect-1.0 kvssink
```

Dovrebbero apparire le seguenti informazioni, come i dettagli di fabbrica e del plugin:

```
Factory Details:
  Rank                     primary + 10 (266)
  Long-name                KVS Sink
  Klass                    Sink/Video/Network
  Description              GStreamer AWS KVS plugin
  Author                   AWS KVS <kinesis-video-support@amazon.com>

Plugin Details:
  Name                     kvssink
  Description              GStreamer AWS KVS plugin
  Filename                 /Users/YOUR_USER/amazon-kinesis-video-streams-producer-sdk-
cpp/build/libgstkvssink.so
  Version                  1.0
  License                  Proprietary
  Source module            kvssinkpackage
  Binary package           GStreamer
  Origin URL               http://gstreamer.net/

  ...
```

Fase 3: esegui Gstreamer con il plugin Kinesis Video Streams

Prima di iniziare lo streaming dalla fotocamera di un dispositivo a Kinesis Video Streams, potrebbe essere necessario convertire l'origine multimediale in un codec accettabile per Kinesis Video Streams. Per determinare le specifiche e le funzionalità di formato dei dispositivi attualmente collegati al computer, esegui il seguente comando.

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

Per iniziare lo streaming, avvia Gstreamer con il seguente comando di esempio e aggiungi le tue credenziali e le informazioni di Flusso di video Amazon Kinesis. Dovresti utilizzare le chiavi di accesso e la regione del ruolo di servizio IAM che hai creato concedendo ad Amazon Rekognition l'accesso ai tuoi flussi Kinesis. Per ulteriori informazioni sulle chiavi di accesso, consulta Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM. Inoltre, puoi regolare i parametri degli argomenti del formato video in base alle tue esigenze di utilizzo e disponibili sul tuo dispositivo.

```
$ gst-launch-1.0 autovideosrc device=/dev/video0 ! videoconvert ! video/x-
raw,format=I420,width=640,height=480,framerate=30/1 !
                x264enc bframes=0 key-int-max=45 bitrate=500 ! video/x-h264,stream-
format=avc,alignment=au,profile=baseline !
                kvssink stream-name="YOUR_STREAM_NAME" storage-size=512 access-
key="YOUR_ACCESS_KEY" secret-key="YOUR_SECRET_ACCESS_KEY" aws-region="YOUR_AWS_REGION"
```

Per ulteriori comandi di avvio, consulta Esempi di comandi di GStreamer avvio.

> **Note**
>
> Se il comando di avvio termina con un errore di mancata negoziazione, controlla l'output dal monitor del dispositivo e assicurati che i valori del parametro `videoconvert` siano funzionalità valide del tuo dispositivo.

Dopo alcuni secondi, vedrai un feed video dalla fotocamera del tuo dispositivo sul flusso video di Kinesis. Per iniziare a rilevare e abbinare i volti con Amazon Rekognition, avvia l'elaboratore di flussi Video Amazon Rekognition. Per ulteriori informazioni, consulta Panoramica delle operazioni del processore di streaming Video Amazon Rekognition.

Risoluzione dei problemi dello streaming di video

In questo argomento sono disponibili le informazioni per l'uso di Video Amazon Rekognition con i video in streaming.

Argomenti

- Non so se il processore di streaming è stato creato
- Non so se il processore di streaming è stato configurato correttamente
- Il processore di streaming non restituisce risultati
- Lo stato del processore di streaming è FAILED
- Il processore di streaming non restituisce i risultati previsti

Non so se il processore di streaming è stato creato

Usa il AWS CLI comando seguente per ottenere un elenco degli stream processor e il loro stato attuale.

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

È possibile ottenere ulteriori dettagli utilizzando il AWS CLI comando seguente. Sostituisci `stream-processor-name` con il nome processore di streaming desiderato.

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

Non so se il processore di streaming è stato configurato correttamente

Se il codice non restituisce i risultati dell'analisi di Video Amazon Rekognition, è possibile che l'elaboratore di flussi non sia configurato correttamente. Esegui le operazioni seguenti per verificare che il processore di streaming sia configurato correttamente e in grado di produrre risultati.

Per determinare se la soluzione è stata configurata correttamente

1. Eseguire il seguente comando per verificare che il processore di streaming sia in esecuzione. Sostituire `stream-processor-name` con il nome del processore di streaming. Il processore di streaming è in esecuzione se il valore di `Status` è `RUNNING`. Se lo stato è `RUNNING` e non vengono restituiti risultati, consultare Il processore di streaming non restituisce risultati. Se lo stato è `FAILED`, consultare Lo stato del processore di streaming è FAILED.

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

2. Se lo stream processor è in esecuzione, esegui il seguente PowerShell comando o Bash per leggere i dati dal flusso di dati Kinesis in uscita.

   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. Utilizzare lo strumento di decodifica sul sito Web di Base64 Decode per decodificare l'output in una stringa leggibile. Per ulteriori informazioni consultare Fase 3: estrazione del record.
4. Se i comandi funzionano e i risultati del rilevamento facciale sono presenti nel flusso di dati Kinesis, la soluzione è configurata correttamente. Se il comando non riesce, verificare gli altri suggerimenti della risoluzione dei problemi e consultare Concessione ad Video Amazon Rekognition dell'accesso alle risorse.

In alternativa, puoi utilizzare il AWS Lambda modello "kinesis-process-record" per registrare i messaggi dal flusso di dati Kinesis CloudWatch per una visualizzazione continua. Ciò comporta costi aggiuntivi per e. AWS Lambda CloudWatch

Il processore di streaming non restituisce risultati

Il processore di streaming potrebbe non restituire risultati per diversi motivi.

Motivo 1: il processore di streaming non è configurato correttamente

Il processore di streaming potrebbe non essere configurato correttamente. Per ulteriori informazioni, consulta Non so se il processore di streaming è stato configurato correttamente.

Motivo 2: il processore di streaming non è nello stato RUNNING

Per risolvere il problema dello stato del processore di streaming

1. Controllate lo stato dello stream processor con il seguente AWS CLI comando.

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

2. Se il valore di `Status` è `STOPPED`, avviare del processore di streaming con il seguente comando:

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

3. Se il valore di `Status` è `FAILED`, consultare Lo stato del processore di streaming è FAILED.
4. Se il valore di `Status` è `STARTING`, attendere 2 minuti e controllare lo stato ripetendo la fase 1. Se il valore dello stato è ancora `STARTING`, procedere nel seguente modo:

    a. Eliminare il processore di streaming con il seguente comando.

```
aws rekognition delete-stream-processor --name stream-processor-name
```

   b. Creare un nuovo processore di streaming con la stessa configurazione. Per ulteriori informazioni, consulta Utilizzo degli eventi video in streaming.
   c. Se i problemi persistono, contatta l' AWS assistenza.
5. Se il valore di `Status` è `RUNNING`, consultare Motivo 3: non sono presenti dati attivi nel flusso video Kinesis.

Motivo 3: non sono presenti dati attivi nel flusso video Kinesis

Per controllare se sono presenti dati attivi nel flusso video Kinesis

1. Accedi a e apri AWS Management Console la console Amazon Kinesis Video https://console.aws.amazon.com/kinesisvideo/Streams all'indirizzo.
2. Seleziona il flusso video Kinesis che è l'input per l'elaboratore di flussi Amazon Rekognition.
3. Se l'anteprima indica Dati non presenti nel flusso, non sono presenti dati nel flusso di input per l'elaborazione di Video Amazon Rekognition.

Per informazioni su come produrre video con Kinesis Video Streams, consulta Librerie Producer di Kinesis Streams.

Lo stato del processore di streaming è FAILED

Puoi controllare lo stato di uno stream processor utilizzando il seguente AWS CLI comando.

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

Se il valore dello stato è FAILED, verifica le informazioni della risoluzione dei problemi per i seguenti messaggi di errore.

Errore di accesso negato al ruolo

Il ruolo IAM utilizzato dall'elaboratore di flussi non esiste o Video Amazon Rekognition non dispone dell'autorizzazione per assumere il ruolo.

Per risolvere i problemi relativi all'accesso al ruolo IAM

1. Accedi AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Dal riquadro di navigazione a sinistra, scegliere Ruoli e verificare che il ruolo esista.

3. Se il ruolo esiste, verifica che il ruolo abbia la politica delle AmazonRekognitionServiceRoleautorizzazioni.
4. Se il ruolo non esiste o non dispone delle autorizzazioni appropriate, vedere Concessione ad Video Amazon Rekognition dell'accesso alle risorse.
5. Avvia lo stream processor con il seguente AWS CLI comando.

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

Errore di accesso negato a Kinesis Video o a Kinesis Data

Il ruolo non dispone dell'accesso alle operazioni API Kinesis Video Streams `GetMedia` e `GetDataEndpoint`. È inoltre possibile che non abbia accesso alle operazioni API Kinesis Data Streams `PutRecord` e `PutRecords`.

Per risolvere i problemi relativi alle autorizzazioni delle API

1. Accedi AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Aprire il ruolo e verificare che sia collegata la seguente policy di autorizzazioni.

```
{
    "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. Se mancano le autorizzazioni, aggiornare la policy. Per ulteriori informazioni, consulta Concessione ad Video Amazon Rekognition dell'accesso alle risorse.

Errore: «Lo stream ***input-video-stream-name*** non esiste»

L'input del flusso video Kinesis per l'elaboratore di flussi non esiste o non è configurato correttamente.

Per risolvere i problemi relativi al flusso video Kinesis

1. Utilizzare il comando seguente per verificare che il flusso esista.

```
aws kinesisvideo list-streams
```

2. Se il flusso è presente, verificare quanto segue.

   - Il nome della risorsa Amazon (ARN) deve essere uguale all'ARN del flusso di input per il processore di streaming.
   - Il flusso video Kinesis è nella stessa regione dell'elaboratore di flussi.

   Se lo stream processor non è configurato correttamente, eliminalo con il seguente AWS CLI comando.

```
aws rekognition delete-stream-processor --name stream-processor-name
```

3. Creare un nuovo processore di streaming con il flusso video Kinesis desiderato. Per ulteriori informazioni, consulta Creazione dell'elaboratore di flussi per la ricerca di volti Video Amazon Rekognition.

Errore di raccolta non trovata

La raccolta Amazon Rekognition utilizzata dall'elaboratore di flussi per il riconoscimento dei volti non esiste oppure viene utilizzata una raccolta non corretta.

Per verificare la raccolta

1. Utilizzate il AWS CLI comando seguente per determinare se esiste la raccolta richiesta. Passa `region` alla AWS regione in cui stai utilizzando lo stream processor.

   ```
   aws rekognition list-collections --region region
   ```

   Se la raccolta non esiste, creare una nuova raccolta e aggiungere le informazioni per il riconoscimento facciale. Per ulteriori informazioni, consulta Ricerca di volti in una raccolta.
2. Nella chiamata a CreateStreamProcessor, controllare che il valore del parametro di input `CollectionId` sia corretto.
3. Avvia lo stream processor con il seguente AWS CLI comando.

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

Errore: «Stream ***output-kinesis-data-stream-name*** nell'account ***account-id*** non trovato»

Il flusso di dati Kinesis in uscita utilizzato dallo stream processor non esiste nella tua regione Account AWS o non si trova nella stessa AWS regione dello stream processor.

Per risolvere i problemi relativi al flusso di dati Kinesis

1. Usa il seguente AWS CLI comando per determinare se esiste il flusso di dati Kinesis. Passa `region` alla AWS regione in cui stai utilizzando lo stream processor.

   ```
   aws kinesis list-streams --region region
   ```
2. Se il flusso di dati Kinesis esiste, verificare che il nome del flusso di dati Kinesis sia uguale al nome del flusso di output utilizzato dal processore di streaming.
3. Se il flusso di dati Kinesis non esiste, potrebbe esistere in un'altra AWS regione. Il flusso di dati Kinesis deve essere nella stessa regione del processore di streaming.
4. Se necessario, creare un nuovo flusso di dati Kinesis.

   a. Creare un flusso di dati Kinesis con lo stesso nome del processore di flusso. Per ulteriori informazioni, consultare Fase 1: creazione di un flusso di dati.

   b. Avvia lo stream processor con il seguente AWS CLI comando.

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

Il processore di streaming non restituisce i risultati previsti

Se il processore di streaming non restituisce le corrispondenze previste per il riconoscimento facciale, utilizza le seguenti informazioni.

- Ricerca di volti in una raccolta
- Raccomandazioni per le impostazioni della fotocamera (video in streaming)

Rilevamento dei movimenti delle persone

> **Note**
>
> Avviso di fine del supporto: il 31 ottobre 2025, AWS interromperà il supporto per Amazon Rekognition People Pathing. Dopo il 31 ottobre 2025, non sarà più possibile utilizzare la funzionalità Rekognition People Pathing. Per ulteriori informazioni, consulta questo post del blog.

Video Amazon Rekognition è in grado di tracciare i movimenti delle persone nei video e fornire informazioni quali:

- Posizione della persona nel fotogramma video al momento in cui viene tracciato il suo movimento.
- Punti di riferimento del volto, ad esempio la posizione dell'occhio sinistro, quando rilevato.

Il rilevamento dei movimenti delle persone di Video Amazon Rekognition nei video archiviati è un'operazione asincrona. Per avviare il percorso delle persone durante le videochiamate StartPersonTracking. Amazon Rekognition per video pubblica lo stato di completamento dell'analisi video in un argomento Amazon Simple Notification Service. Se l'analisi video ha esito positivo, chiama GetPersonTrackingper ottenere i risultati dell'analisi video. Per ulteriori informazioni su come chiamare le operazioni API di Video Amazon Rekognition consultare Chiamata delle operazioni Video Amazon Rekognition.

La procedura seguente mostra come tracciare i movimenti delle persone attraverso un video archiviato in un bucket Amazon S3. L'esempio si espande nel codice in Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK), che utilizza una coda Amazon Simple Queue Service per ottenere lo stato di completamento di una richiesta di analisi video.

Per rilevare le persone in un video archiviato in un bucket Amazon S3 (SDK)

1. Eseguire Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK).
2. Aggiungere il seguente codice alla classe `VideoDetect` creata nella fase 1.

Java

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

       //
Persons=============================================================================
       private static void StartPersonDetection(String bucket, String video)
 throws Exception{


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

        StartPersonTrackingRequest req = new StartPersonTrackingRequest()
                .withVideo(new Video()
                        .withS3Object(new S3Object()
                            .withBucket(bucket)
                            .withName(video)))
                .withNotificationChannel(channel);



        StartPersonTrackingResult startPersonDetectionResult =
 rek.startPersonTracking(req);
        startJobId=startPersonDetectionResult.getJobId();

       }

       private static void GetPersonDetectionResults() throws Exception{
           int maxResults=10;
           String paginationToken=null;
           GetPersonTrackingResult personTrackingResult=null;

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

```
                personTrackingResult = rek.getPersonTracking(new
GetPersonTrackingRequest()
                     .withJobId(startJobId)
                     .withNextToken(paginationToken)
                     .withSortBy(PersonTrackingSortBy.TIMESTAMP)
                     .withMaxResults(maxResults));

                VideoMetadata
videoMetaData=personTrackingResult.getVideoMetadata();

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


                //Show persons, confidence and detection times
                List<PersonDetection> detectedPersons=
personTrackingResult.getPersons();

                for (PersonDetection detectedPerson: detectedPersons) {

                   long seconds=detectedPerson.getTimestamp()/1000;
                   System.out.print("Sec: " + Long.toString(seconds) + " ");
                   System.out.println("Person Identifier: "  +
detectedPerson.getPerson().getIndex());
                        System.out.println();
                }
           }  while (personTrackingResult !=null &&
personTrackingResult.getNextToken() != null);

       }
```

Nella funzione `main`, sostituisci le righe:

```
       StartLabelDetection(amzn-s3-demo-bucket, video);

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

con:

```
        StartPersonDetection(amzn-s3-demo-bucket, video);

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

Java V2

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
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
```

Nella funzione `main`, sostituisci le righe:

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

con:

```
    analyzer.StartPersonPathing()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetPersonPathingResults()
```

CLI

Esegui il AWS CLI comando seguente per avviare le persone a creare percorsi in un video.

```
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
```

Aggiorna i seguenti valori:

- Modifica `amzn-s3-demo-bucket` e `video-name` con il nome del bucket Amazon S3 e il nome del file specificati nella fase 2.
- Cambia `region-name` con la regione AWS che stai utilizzando.
- Sostituisci il valore di `profile-name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.
- Cambia `topic-ARN` con l'ARN dell'argomento Amazon SNS creato nella fase 3 di Configurazione di Video Amazon Rekognition.
- Modifica `role-ARN` con l'ARN del ruolo di servizio IAM creato nella fase 7 di Configurazione di Video Amazon Rekognition.

Se accedi alla CLI da un dispositivo Windows, usa le virgolette doppie anziché le virgolette singole ed evita le virgolette doppie interne tramite barra rovesciata (ovvero, \) per risolvere eventuali errori del parser che potresti riscontrare. Un esempio è fornito di seguito:

```
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
```

Dopo aver eseguito l'esempio di codice precedente, copia il `jobID` restituito e inseriscilo nel comando `GetPersonTracking` di seguito per ottenere i risultati, sostituendo `job-id-number` con il `jobID` ricevuto in precedenza:

```
aws rekognition get-person-tracking --job-id job-id-number
```

> **Note**
>
> Se hai già eseguito un video di esempio diverso da Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK), il codice da sostituire potrebbe essere diverso.

3. Eseguire il codice. Gli identificatori univoci per le persone localizzate vengono visualizzati insieme al tempo, in secondi, durante il quale i movimenti delle persone sono stati tracciati.

GetPersonTracking risposta all'operazione

`GetPersonTracking` restituisce una matrice, `Persons`, degli oggetti di PersonDetection contenenti dettagli sulle persone rilevate nel video e quando i loro movimenti vengono tracciati.

Puoi ordinare `Persons` utilizzando il parametro di input `SortBy`. Specifica `TIMESTAMP` per ordinare gli elementi in base al momento in cui i movimenti delle persone vengono tracciati nel video. Specifica

INDEX per ordinare in base alle persone localizzate nel video. All'interno di ogni set di risultati per una persona, gli elementi vengono disposti in ordine di affidabilità decrescente riguardo alla precisione del rilevamento dei movimenti. Per default, la matrice Persons restituita è ordinata in base al valore di TIMESTAMP. Di seguito è riportato un esempio di risposta JSON dell'operazione GetPersonDetection. I risultati sono ordinati in base al momento (in millisecondi) dall'inizio del video in cui i movimenti delle persone vengono tracciati nel video. Nella risposta, tenere presente quanto segue:

- Informazioni sulla persona – L'elemento matrice PersonDetection contiene informazioni sulla persona rilevata. Ad esempio, il momento in cui la persona è stata rilevata (Timestamp), la posizione della persona nel fotogramma video al momento in cui è stata rilevata (BoundingBox) e il livello di affidabilità di Video Amazon Rekognition riguardo al corretto rilevamento della persona (Confidence).

  Le caratteristiche del viso non vengono restituite per tutti i timestamp in corrispondenza dei quali il movimento della persona viene tracciato. Inoltre, se il corpo di una persona localizzata non è visibile, viene restituita solo la posizione del suo volto.
- Informazioni di paginazione – L'esempio mostra una pagina di informazioni di rilevamento delle persone. Puoi specificare il numero di elementi della persona da restituire nel parametro di input MaxResults per GetPersonTracking. Se esiste un numero di risultati maggiore di MaxResults, GetPersonTracking restituisce un token (NextToken) utilizzato per ottenere la pagina di risultati successiva. Per ulteriori informazioni, consulta Ottenere i risultati dell'analisi di Video Amazon Rekognition.
- Indice – Un identificatore univoco per identificare la persona in tutto il video.
- Informazioni video – La risposta include informazioni sul formato video (VideoMetadata) in ogni pagina di informazioni restituita da GetPersonDetection.

```
{
    "JobStatus": "SUCCEEDED",
    "NextToken": "AcDymG0fSSoaI6+BBYpka5wVlqttysSPP8VvWcujMDluj1QpFo/vf
+mrMoqBGk8eUEiFlllR6g==",
    "Persons": [
        {
            "Person": {
                "BoundingBox": {
                    "Height": 0.8787037134170532,
                    "Left": 0.00572916679084301,
```

```
                    "Top": 0.12129629403352737,
                    "Width": 0.21666666865348816
                },
                "Face": {
                    "BoundingBox": {
                        "Height": 0.20000000298023224,
                        "Left": 0.029999999329447746,
                        "Top": 0.2199999988079071,
                        "Width": 0.11249999701976776
                    },
                    "Confidence": 99.85971069335938,
                    "Landmarks": [
                        {
                            "Type": "eyeLeft",
                            "X": 0.06842322647571564,
                            "Y": 0.3010137975215912
                        },
                        {
                            "Type": "eyeRight",
                            "X": 0.10543643683195114,
                            "Y": 0.29697132110595703
                        },
                        {
                            "Type": "nose",
                            "X": 0.09569807350635529,
                            "Y": 0.33701086044311523
                        },
                        {
                            "Type": "mouthLeft",
                            "X": 0.0732642263174057,
                            "Y": 0.3757539987564087
                        },
                        {
                            "Type": "mouthRight",
                            "X": 0.10589495301246643,
                            "Y": 0.3722417950630188
                        }
                    ],
                    "Pose": {
                        "Pitch": -0.5589138865470886,
                        "Roll": -5.1093974113464355,
                        "Yaw": 18.69594955444336
                    },
                    "Quality": {
```

```
                    "Brightness": 43.052337646484375,
                    "Sharpness": 99.68138885498047
                }
            },
            "Index": 0
        },
        "Timestamp": 0
    },
    {
        "Person": {
            "BoundingBox": {
                "Height": 0.9074074029922485,
                "Left": 0.24791666865348816,
                "Top": 0.09259258955717087,
                "Width": 0.375
            },
            "Face": {
                "BoundingBox": {
                    "Height": 0.23000000417232513,
                    "Left": 0.42500001192092896,
                    "Top": 0.16333332657814026,
                    "Width": 0.12937499582767487
                },
                "Confidence": 99.97504425048828,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.46415066719055176,
                        "Y": 0.2572723925113678
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.5068183541297913,
                        "Y": 0.23705792427062988
                    },
                    {
                        "Type": "nose",
                        "X": 0.49765899777412415,
                        "Y": 0.28383663296699524
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.487221896648407,
                        "Y": 0.3452930748462677
```

```
                        },
                        {
                            "Type": "mouthRight",
                            "X": 0.5142884850502014,
                            "Y": 0.33167609572410583
                        }
                    ],
                    "Pose": {
                        "Pitch": 15.966927528381348,
                        "Roll": -15.547388076782227,
                        "Yaw": 11.34195613861084
                    },
                    "Quality": {
                        "Brightness": 44.80223083496094,
                        "Sharpness": 99.95819854736328
                    }
                },
                "Index": 1
            },
            "Timestamp": 0
        }.....

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

Rilevamento dei dispositivi di protezione individuale

Amazon Rekognition è in grado di rilevare i dispositivi di protezione individuale (DPI) indossati dalle persone in un'immagine. Puoi utilizzare queste informazioni per migliorare le pratiche di sicurezza sul lavoro. Ad esempio, è possibile utilizzare il rilevamento dei DPI per determinare se i lavoratori in un cantiere edile indossano copricapi o se gli operatori sanitari indossano protezioni per il volto e per le mani. L'immagine seguente mostra alcuni dei tipi di DPI che è possibile rilevare.

Per rilevare i PPE in un'immagine si chiama l'DetectProtectiveEquipmentAPI e si passa un'immagine di input. La risposta è una struttura JSON che include quanto segue.

- Le persone rilevate nell'immagine.
- Le parti del corpo in cui vengono indossati i DPI (viso, testa, mano sinistra e mano destra).
- I tipi di DPI rilevati sulle parti del corpo (protezione per il volto, protezione per le mani e copricapo).

- Per i DPI rilevati, un indicatore che indica se il DPI copre o meno la parte del corpo corrispondente.

Vengono restituiti i riquadri di delimitazione per indicare l'ubicazione delle persone e degli oggetti dei DPI rilevati nell'immagine.

Facoltativamente, è possibile richiedere un riepilogo degli articoli e delle persone DPI rilevati in un'immagine. Per ulteriori informazioni, consulta Riepilogo dei DPI rilevati in un'immagine.

> **Note**
>
> Il rilevamento DPI di Amazon Rekognition non esegue il riconoscimento o il confronto facciale e non è in grado di identificare le persone rilevate.

Tipi di DPI

DetectProtectiveEquipmentrileva i seguenti tipi di DPI. Se desideri rilevare altri tipi di DPI nelle immagini, prendi in considerazione l'utilizzo di Etichette personalizzate Amazon Rekognition per addestrare un modello personalizzato. Per maggiori informazioni, consulta Etichette personalizzate Amazon Rekognition.

Protezione per il volto

`DetectProtectiveEquipment` è in grado di rilevare le protezioni per il volto comuni come quelle chirurgiche, N95 e maschere in tessuto.

Guanti

`DetectProtectiveEquipment` è in grado di rilevare protezioni per le mani come guanti chirurgici e guanti di sicurezza.

Copricapo

`DetectProtectiveEquipment` è in grado di rilevare elmetti e caschi.

L'API indica che in un'immagine è stata rilevata una protezione per la testa, la mano o il viso. L'API non restituisce informazioni sul tipo di protezione specifica. Ad esempio, «guanto chirurgico» per il tipo di protezione per le mani.

Affidabilità nel rilevamento dei DPI

Amazon Rekognition effettua una previsione sulla presenza di DPI, persone e parti del corpo in un'immagine. L'API fornisce un punteggio (50-100) che indica la fiducia di Amazon Rekognition nell'accuratezza di una previsione.

> **Note**
>
> Se prevedi di usare l'operazione `DetectProtectiveEquipment` per prendere una decisione che influisca sui diritti, sulla privacy o sull'accesso ai servizi di una persona, ti consigliamo di trasmettere il risultato a un essere umano per la revisione e la convalida prima di agire.

Riepilogo dei DPI rilevati in un'immagine

Facoltativamente è possibile richiedere un riepilogo degli articoli e delle persone DPI rilevati in un'immagine. È possibile specificare un elenco dei dispositivi di protezione richiesti (protezione per il volto, protezione per le mani o copricapo) e una soglia minima di confidenza (ad esempio, 80%). La risposta include un riepilogo consolidato per immagine identificativo (ID) delle persone con il DPI richiesto, delle persone senza il DPI richiesto e delle persone per le quali non è stato possibile effettuare una determinazione.

Il riepilogo consente di rispondere rapidamente a domande come Quante persone non indossano protezioni per il volto? o Tutti indossano DPI? Ogni persona rilevata nel riepilogo ha un ID univoco. Puoi utilizzare l'ID per scoprire informazioni come la posizione del riquadro di delimitazione di una persona che non indossa dispositivi di protezione individuale.

> **Note**
>
> L'ID viene generato casualmente sulla base di un'analisi per immagine e non è coerente tra le immagini o le analisi multiple della stessa immagine.

Puoi riassumere le protezioni per il viso, i copricapi, le protezioni per le mani o una combinazione a tua scelta. Per specificare i tipi di DPI richiesti, consulta Specificazione dei requisiti di riepilogo. È inoltre possibile specificare un livello di confidenza minimo (50-100) che deve essere soddisfatto affinché i rilevamenti vengano inclusi nel riepilogo.

Per ulteriori informazioni sulla risposta di riepilogo da `DetectProtectiveEquipment`, consulta Comprendere la risposta DetectProtectiveEquipment .

Tutorial: creazione di una AWS Lambda funzione che rileva le immagini con PPE

Puoi creare una AWS Lambda funzione che rileva i dispositivi di protezione individuale (DPI) nelle immagini che si trovano in un bucket Amazon S3. Consulta l' GitHub archivio degli esempi di AWS Documentation SDK per questo tutorial su Java V2.

Comprensione dell'API di rilevamento dei dispositivi di protezione individuale

Le seguenti informazioni descrivono l'API. DetectProtectiveEquipment Per il codice di esempio, consulta Rilevamento dei dispositivi di protezione individuale in un immagine.

Fornire un'immagine

Puoi fornire un'immagine di input (formato JPG o PNG) sia come byte di immagine sia facendo riferimento a un'immagine archiviata in un bucket S3.

Si consiglia di utilizzare immagini in cui il viso della persona è rivolto verso la fotocamera.

Se l'immagine in ingresso non è ruotata di 0 gradi, consigliamo di ruotarla di 0 gradi prima di inviarla a `DetectProtectiveEquipment`. Le immagini in formato JPG potrebbero contenere informazioni di orientamento nei metadati Exif (Exchangeable Image File Format). È possibile utilizzare queste informazioni per scrivere un codice che ruoti l'immagine. Per ulteriori informazioni, consulta Exif versione 2.32. Le immagini in formato PNG non contengono informazioni sull'orientamento dell'immagine.

Per trasferire un'immagine da un bucket Amazon S3, usa un utente con almeno i privilegi AmazonS3. ReadOnlyAccess Usa un utente con privilegi `AmazonRekognitionFullAccess` per chiamare `DetectProtectiveEquipment.`

Nell'esempio seguente input JSON, l'immagine viene passata in un bucket Amazon S3. Per ulteriori informazioni, consulta Lavorare con le immagini. L'esempio richiede un riepilogo di tutti i tipi di DPI (copricapo, protezione per i guanti e protezione per il volto) con una confidenza di rilevamento minima (`MinConfidence`) dell'80%. È necessario specificare un valore `MinConfidence` compreso tra

il 50 e il 100%, in quanto `DetectProtectiveEquipment` restituisce previsioni solo quando la confidenza del rilevamento è compresa tra il 50% e il 100%. Se si specifica un valore inferiore al 50%, i risultati sono gli stessi specificando un valore del 50%. Per ulteriori informazioni, consulta Specificazione dei requisiti di riepilogo.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "worker.jpg"
        }
    },
    "SummarizationAttributes": {
        "MinConfidence": 80,
        "RequiredEquipmentTypes": [
            "FACE_COVER",
            "HAND_COVER",
            "HEAD_COVER"
        ]
    }
}
```

Se disponi di un'ampia raccolta di immagini da elaborare, considera la possibilità di utilizzare Batch AWS per elaborare le chiamate a `DetectProtectiveEquipment` in batch in background.

Specificazione dei requisiti di riepilogo

Facoltativamente, puoi utilizzare il parametro di input `SummarizationAttributes` (ProtectiveEquipmentSummarizationAttributes) per richiedere informazioni di riepilogo sui tipi di DPI rilevati in un'immagine.

Per specificare i tipi di DPI da riepilogare, utilizzate il campo array `RequiredEquipmentTypes`. Nell'array, includete uno o più di `FACE_COVER`, `HAND_COVER` o `HEAD_COVER`.

Utilizzate il campo `MinConfidence` per specificare una confidenza minima di rilevamento (50-100). Il riepilogo non include persone, parti del corpo, parti del corpo coperte e articoli di DPI rilevati con un grado di confidenza inferiore a `MinConfidence`.

Per informazioni sulla risposta di riepilogo da `DetectProtectiveEquipment`, consulta Comprendere la risposta DetectProtectiveEquipment .

Comprendere la risposta DetectProtectiveEquipment

`DetectProtectiveEquipment` restituisce un array di persone rilevate nell'immagine di input. Per ogni persona, vengono restituite informazioni sulle parti del corpo rilevate e sugli articoli di DPI rilevati. Il codice JSON per l'immagine seguente di un lavoratore che indossa un copricapo, una protezione per le mani e una protezione per il volto è il seguente.

Nel JSON, tenere presente quanto segue.

- Persone rilevate: `Persons` è un array di persone rilevate nell'immagine (comprese le persone che non indossano DPI). `DetectProtectiveEquipment` è in grado di rilevare i DPI su un massimo di 15 persone rilevate in un'immagine. Ogni ProtectiveEquipmentPersonoggetto dell'array contiene un ID personale, un riquadro di delimitazione per la persona, le parti del corpo rilevate e gli elementi di DPI rilevati. Il valore di `Confidence` in `ProtectiveEquipmentPerson` indica la percentuale di fiducia di Amazon Rekognition che il riquadro di delimitazione contenga una persona.

- Parti del corpo: `BodyParts` è una serie di parti del corpo (ProtectiveEquipmentBodyPart) rilevate su una persona (comprese le parti del corpo non coperte dai DPI). Ciascuna `ProtectiveEquipmentBodyPart` include il nome (`Name`) della parte del corpo rilevata. `DetectProtectEquipment` è in grado di rilevare parti del corpo del volto, della testa, della mano sinistra e della mano destra. Il campo `Confidence` in `ProtectiveEquipmentBodyPart` indica la percentuale di confidenza di Amazon Rekognition nella precisione di rilevamento della parte del corpo.
- Elementi DPI: l'array `EquipmentDetections` di un oggetto `ProtectiveEquipmentBodyPart` contiene una serie di elementi DPI rilevati. Ogni EquipmentDetectionoggetto contiene i seguenti campi.
  - `Type`— Il tipo di DPI rilevato.
  - `BoundingBox`— un riquadro di delimitazione attorno al DPI rilevato.
  - `Confidence`— La fiducia di Amazon Rekognition che il riquadro di delimitazione contenga i DPI rilevati.
  - `CoversBodyPart`— Indica se il DPI rilevato si trova sulla parte del corpo corrispondente.

  Il CoversBodyPartcampo `Value` è un valore booleano che indica se il DPI rilevato si trova sulla parte del corpo corrispondente. Il campo `Confidence` indica la fiducia nella previsione. È possibile usare `CoversBodyPart` per filtrare i casi in cui il DPI rilevato è presente nell'immagine, ma non effettivamente sulla persona.

  > **Note**
  >
  > `CoversBodyPart` non indica o implica che la persona sia adeguatamente protetta dall'equipaggiamento protettivo o che l'equipaggiamento protettivo stesso sia indossato correttamente.

- Informazioni di riepilogo – `Summary` contiene le informazioni di riepilogo specificate nel parametro `SummarizationAttributes` di input. Per ulteriori informazioni, consulta Specificazione dei requisiti di riepilogo.

  `Summary`è un oggetto di tipo ProtectiveEquipmentSummaryche contiene le seguenti informazioni.
  - `PersonsWithRequiredEquipment`— Una serie IDs di persone in cui ogni persona soddisfa i seguenti criteri.
    - La persona indossa tutti i DPI specificati nel parametro `SummarizationAttributes` di input.

    - La `Confidence` per la persona (`ProtectiveEquipmentPerson`), parte del corpo (`ProtectiveEquipmentBodyPart`), equipaggiamento protettivo (`EquipmentDetection`) è uguale o superiore alla soglia minima di confidenza specificata (`MinConfidence`).
    - Il valore di `CoversBodyPart` per tutti gli articoli di DPI è vero.
- `PersonsWithoutRequiredEquipment`— Una serie IDs di persone che soddisfano uno dei seguenti criteri.
    - Il valore `Confidence` per la persona (`ProtectiveEquipmentPerson`), parte del corpo (`ProtectiveEquipmentBodyPart`) e copertura delle parti del corpo (`CoversBodyPart`) sono superiori alla soglia minima di confidenza specificata (`MinConfidence`), ma alla persona mancano uno o più DPI specificati (`SummarizationAttributes`).
    - Il valore di `CoversBodyPart` è false per ogni DPI specificato (`SummarizationAttributes`) con un valore `Confidence` superiore alla soglia di confidenza minima specificata (`MinConfidence`). La persona dispone inoltre di tutti i DPI specificati (`SummarizationAttributes`), e i valori `Confidence` per persona (`ProtectiveEquipmentPerson`), parte del corpo (`ProtectiveEquipmentBodyPart`) e equipaggiamento protettivo (`EquipmentDetection`) sono superiori o uguali alla soglia minima di confidenza (`MinConfidence`).
- `PersonsIndeterminate`— Una serie IDs di persone rilevate in cui il `Confidence` valore della persona (`ProtectiveEquipmentPerson`), della parte del corpo (), dell'equipaggiamento protettivo (`EquipmentDetection`) o del valore `CoversBodyPart` booleano è inferiore alla soglia di confidenza minima specificata (`MinConfidence`). `ProtectiveEquipmentBodyPart`

Utilizzate la dimensione dell'array per ottenere un conteggio per un particolare riepilogo. Ad esempio, la taglia di `PersonsWithRequiredEquipment` indica il numero di persone rilevate che indossano il tipo di DPI specificato.

Puoi utilizzare l'ID della persona per scoprire ulteriori informazioni su una persona, come la posizione del riquadro di delimitazione della persona. L'ID della persona viene mappato al campo ID di un oggetto restituito (`ProtectiveEquipmentPerson`) in `Persons` (array di`ProtectiveEquipmentPerson`). È quindi possibile ottenere il riquadro di delimitazione e altre informazioni dall'oggetto `ProtectiveEquipmentPerson` corrispondente.

```
{
   "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": []
    }
}
```

Rilevamento dei dispositivi di protezione individuale in un immagine

Per rilevare i dispositivi di protezione individuale (DPI) sulle persone in un'immagine, utilizzate l'operazione API DetectProtectiveEquipmentsenza archiviazione.

Puoi fornire l'immagine di input come matrice di byte dell'immagine (byte dell'immagine codificata in formato Base64) o come un oggetto Amazon S3 utilizzando AWS SDK o AWS Command Line Interface (AWS CLI). Questi esempi utilizzano un'immagine memorizzata in un bucket Amazon S3. Per ulteriori informazioni, consulta Lavorare con le immagini.

Per rilevare i DPI sulle persone in un'immagine

1. Se non lo hai già fatto:

   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

   b. Installa e configura il AWS CLI e il. AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Carica un'immagine (contenente una o più persone che indossano un DPI) nel bucket S3.

   Per le istruzioni, consulta Caricamento di oggetti in Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

3. Utilizzare i seguenti esempi per richiamare l'operazione `DetectProtectiveEquipment`. Per informazioni sulla visualizzazione dei riquadri di delimitazione in un'immagine, consultare Visualizzazione di riquadri di delimitazione.

   Java

   Questo esempio visualizza informazioni sui DPI rilevati sulle persone rilevate in un'immagine.

   Modificare il valore di `amzn-s3-demo-bucket` con il nome del bucket Amazon S3 che contiene l'immagine. Modifica il valore di `photo` nel nome del file immagine.

```
//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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
//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

Questo AWS CLI comando richiede un riepilogo PPE e visualizza l'output JSON per l'operazione CLI`detect-protective-equipment`.

Sostituisci `amzn-s3-demo-bucket` con il nome di un bucket Amazon S3 contenente un'immagine. Sostituisci `input.jpg` con il nome dell'immagine che si desidera utilizzare.

```
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']"
```

Questo AWS CLI comando visualizza l'output JSON per l'operazione `detect-protective-equipment` CLI.

Sostituisci `amzn-s3-demo-bucket` con il nome di un bucket Amazon S3 contenente un'immagine. Sostituisci `input.jpg` con il nome dell'immagine che si desidera utilizzare.

```
aws rekognition detect-protective-equipment \
  --image "S3Object={Bucket=amzn-s3-demo-bucket,Name=input.jpg}"
```

Python

Questo esempio visualizza informazioni sui DPI rilevati sulle persone rilevate in un'immagine.

Modificare il valore di `amzn-s3-demo-bucket` con il nome del bucket Amazon S3 che contiene l'immagine. Modifica il valore di `photo` nel nome del file immagine. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
# 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()
```

Esempio: disegnare riquadri di delimitazione attorno alle protezioni del viso

Gli esempi seguenti mostrano come disegnare dei riquadri di delimitazione attorno alle protezioni del viso rilevate sulle persone. Per un esempio che utilizza AWS Lambda Amazon DynamoDB, consulta l'archivio degli esempi di Documentation AWS SDK. GitHub

Per rilevare le coperture facciali, si utilizza l'operazione API non di storage DetectProtectiveEquipment. L'immagine viene caricata dal file system locale. Fornisci un'immagine di input a `DetectProtectiveEquipment` come matrice di byte dell'immagine (byte dell'immagine codificata in formato base64). Per ulteriori informazioni, consulta Lavorare con le immagini.

L'esempio visualizza un riquadro di delimitazione attorno alle protezioni del viso rilevate. Il riquadro di delimitazione è verde se la protezione del viso copre completamente la parte del corpo. Altrimenti viene visualizzato un riquadro di delimitazione rosso. Come avvertimento, se la confidenza di rilevamento è inferiore al valore di confidenza specificato, viene visualizzato un riquadro di delimitazione giallo all'interno del riquadro di delimitazione facciale. Se non viene rilevata una copertura facciale, attorno alla persona viene disegnato un riquadro di delimitazione rosso.

L’immagine output è simile a quello riportato di seguito.

Per visualizzare i riquadri di delimitazione sulle protezioni del viso rilevate

1. Se non lo hai già fatto:

   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Utilizzare i seguenti esempi per richiamare l'operazione `DetectProtectiveEquipment`. Per informazioni sulla visualizzazione dei riquadri di delimitazione in un'immagine, consultare Visualizzazione di riquadri di delimitazione.

   Java

   Nella funzione `main`, modificate quanto segue:

- Il valore di `photo` del percorso e il nome di un file di immagine locale (PNG o JPEG).
- Il valore del livello `confidence` di confidenza desiderato (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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
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

Nella funzione `main`, modificate quanto segue:

- Il valore di `photo` del percorso e il nome di un file di immagine locale (PNG o JPEG).
- Il valore del livello `confidence` di confidenza desiderato (50-100).
- Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
#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

Nel seguente esempio di CLI, modifica il valore degli argomenti elencati di seguito:

- Il valore di `photo` del percorso e il nome di un file di immagine locale (PNG o JPEG).
- Il valore del livello `confidence` di confidenza desiderato (50-100).
- Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
                                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"]}"
```

Se accedi alla CLI da un dispositivo Windows, usa le virgolette doppie anziché le virgolette singole ed evita le virgolette doppie interne tramite barra rovesciata (ovvero, \) per risolvere eventuali errori del parser che potresti riscontrare. Per un esempio, consulta quanto segue:

```
                                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\"]}"
```

Riconoscimento delle celebrità

Amazon Rekognition consente ai clienti di riconoscere automaticamente decine di migliaia di personalità famose in immagini e video utilizzando il machine learning. I metadati forniti dall'API di riconoscimento di volti celebri riducono in modo significativo lo sforzo manuale ripetitivo necessario per applicare tag ai contenuti e renderli facilmente ricercabili.

La rapida proliferazione di contenuti sotto forma di immagini e video significa che le aziende del settore multimediale spesso faticano a organizzare, cercare e utilizzare i propri cataloghi multimediali su larga scala. I canali di informazione e le emittenti sportive spesso hanno bisogno di trovare rapidamente immagini e video, per rispondere agli eventi attuali e creare una programmazione pertinente. La mancanza di metadati rende difficili queste attività, ma con Amazon Rekognition è possibile applicare tag automaticamente a grandi volumi di contenuti nuovi o di archivio per renderli facilmente ricercabili per un set completo di celebrità internazionali di grande fama come attori, sportivi e creatori di contenuti online.

Il riconoscimento di volti celebri di Amazon Rekognition è progettato per essere utilizzato esclusivamente nei casi in cui si prevede che possa esserci una celebrità nota in un'immagine o in un video. Per informazioni sul riconoscimento di volti che non sono celebrità, consulta Ricerca di volti in una raccolta.

> **Note**
>
> Se sei una celebrità e non vuoi essere incluso in questa funzionalità, contatta l'AWS assistenza o invia un'e-mail a <rekognition-celebrity-opt-out@amazon .com>.

Argomenti

- Differenze fra Riconoscimento di volti celebri e Ricerca di volti
- Riconoscimento delle celebrità in un'immagine
- Riconoscimento di celebrità in un video archiviato
- Come recuperare le informazioni su una celebrità

Differenze fra Riconoscimento di volti celebri e Ricerca di volti

Amazon Rekognition include le funzionalità Riconoscimento di volti celebri e Ricerca di volti. Queste funzionalità hanno alcune differenze importanti per quanto riguarda i casi d'uso e le best practice.

Il Riconoscimento di volti celebri è preimpostato per individuare centinaia di migliaia di persone famose in vari campi, tra cui sport, media, politica ed economia. Questa funzionalità è stata progettata per effettuare ricerche in grandi volumi di immagini o video e identificare quelli che potrebbero contenere una determinata celebrità. Non è stata creata per eseguire confronti tra persone diverse che non sono celebrità. In situazioni in cui è importante che il riconoscimento della celebrità sia accurato, consigliamo di ricorrere anche a operatori umani che prendano in esame i contenuti evidenziati dalla funzionalità per garantire un maggiore livello di precisione e l'applicazione del giudizio umano. Il Riconoscimento di volti celebri non deve essere utilizzato in modi che possano ledere i diritti civili e le libertà individuali.

Il riconoscimento di volti è invece una funzionalità più generica che consente di creare una propria raccolta di volti, con vettori personalizzati, per verificare le identità ed eseguire ricerche di persone che possono anche non essere celebri. Il riconoscimento facciale può essere utilizzato in varie applicazioni, come ad esempio controlli per l'accesso a un edificio, sicurezza pubblica e social media. In questi casi, si consiglia di utilizzare le best practice, soglie di sicurezza appropriate (ad esempio, 99% per gli utilizzi di sicurezza pubblica) e l'intervento umano nelle situazioni in cui la precisione della corrispondenza sia importante.

Per ulteriori informazioni, consulta Ricerca di volti in una raccolta.

Riconoscimento delle celebrità in un'immagine

Per riconoscere volti celebri nelle immagini e ottenere ulteriori informazioni sui volti celebri riconosciuti, utilizzare l'operazione RecognizeCelebrities dell'API non basata su storage. Ad esempio, nel settore dei social media o dell'informazione e dell'intrattenimento, in cui l'elemento tempo può rappresentare un fattore determinante nella raccolta delle informazioni, è possibile utilizzare l'operazione `RecognizeCelebrities` per identificare fino a 64 celebrità in un'immagine e restituire collegamenti alle pagine Web delle celebrità, se disponibili. Amazon Rekognition non ricorda l'immagine in cui ha individuato la celebrità. Queste informazioni devono essere archiviate dall'applicazione in uso.

Se non hai archiviato ulteriori informazioni per un volto celebre restituito da `RecognizeCelebrities` e desideri evitare di analizzare di nuovo un'immagine per ottenerle,

utilizza GetCelebrityInfo. Per effettuare una chiamata a `GetCelebrityInfo`, è necessario l'ID univoco che Amazon Rekognition assegna a ogni celebrità. L'identificatore viene restituito come parte della risposta `RecognizeCelebrities` per ogni volto celebre riconosciuto in un'immagine.

Se disponi di un'ampia raccolta di immagini da elaborare per il riconoscimento di volti celebri, considera la possibilità di utilizzare Batch AWS per elaborare le chiamate a `RecognizeCelebrities` in batch in background. Quando aggiungi una nuova immagine alla tua raccolta, puoi utilizzare una AWS Lambda funzione per riconoscere le celebrità `RecognizeCelebrities` chiamando quando l'immagine viene caricata in un bucket S3.

Chiamata RecognizeCelebrities

Puoi fornire l'immagine di input come array di byte di immagine (byte di immagine con codifica base64) o come oggetto Amazon S3, utilizzando () o l'SDK AWS. AWS Command Line Interface AWS CLI Nella AWS CLI procedura, carichi un'immagine in formato.jpg o .png in un bucket S3. Nelle procedure AWS SDK, si utilizza un'immagine caricata dal file system locale. Per informazioni sui suggerimenti per le immagini di input, consulta Lavorare con le immagini.

Per eseguire questa procedura, è necessario disporre di un file immagine contenente uno o più volti celebri.

Per riconoscere volti celebri in un'immagine

1. Se non lo hai già fatto:

   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Utilizzare i seguenti esempi per richiamare l'operazione `RecognizeCelebrities`.

   Java

   Questo esempio mostra le informazioni sui volti celebri che vengono rilevati in un'immagine.

   Modifica il valore di `photo` con il percorso e il nome di un file immagine archiviato in locale contenente uno o più volti celebri.

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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
//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

Questo AWS CLI comando visualizza l'output JSON per l'operazione `recognize-celebrities` CLI.

Sostituisci `amzn-s3-demo-bucket` con il nome di un bucket Amazon S3 contenente un'immagine. Modifica `input.jpg` con il nome di un file immagine archiviato in locale contenente uno o più volti celebri.

Sostituisci il valore di `profile_name` con il nome del tuo profilo di sviluppatore.

```
aws rekognition recognize-celebrities \
  --image "S3Object={Bucket=amzn-s3-demo-bucket,Name=input.jpg}"
```

Se accedi alla CLI da un dispositivo Windows, usa le virgolette doppie anziché le virgolette singole ed evita le virgolette doppie interne tramite barra rovesciata (ovvero, \) per risolvere eventuali errori del parser che potresti riscontrare. Per un esempio, consulta quanto segue:

```
                                    aws rekognition recognize-celebrities --
image \
                                    "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --profile profile-name
```

Python

Questo esempio mostra le informazioni sui volti celebri che vengono rilevati in un'immagine.

Modifica il valore di `photo` con il percorso e il nome di un file immagine archiviato in locale contenente uno o più volti celebri.

Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

Questo esempio mostra le informazioni sui volti celebri che vengono rilevati in un'immagine.

Modifica il valore di `photo` con il percorso e il nome di un file immagine archiviato in locale contenente uno o più volti celebri. Modifica il valore di `amzn-s3-demo-bucket` con quello del nome del bucket S3 che contiene il file immagine fornito. Modifica il valore di `REGION` con quello del nome della regione associata al tuo utente. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
// 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

Questo esempio mostra le informazioni sui volti celebri che vengono rilevati in un'immagine.

Modifica il valore di `photo` con il percorso e il nome di un file immagine archiviato in locale contenente uno o più volti celebri in formato .jpg o .png.

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

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

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

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        RecognizeCelebritiesRequest recognizeCelebritiesRequest = new
 RecognizeCelebritiesRequest();

        Amazon.Rekognition.Model.Image img = new
 Amazon.Rekognition.Model.Image();
        byte[] data = null;
        try
        {
            using (FileStream fs = new FileStream(photo, FileMode.Open,
 FileAccess.Read))
            {
                data = new byte[fs.Length];
```

```
                fs.Read(data, 0, (int)fs.Length);
            }
        }
        catch(Exception)
        {
            Console.WriteLine("Failed to load file " + photo);
            return;
        }

        img.Bytes = new MemoryStream(data);
        recognizeCelebritiesRequest.Image = img;

        Console.WriteLine("Looking for celebrities in image " + photo + "\n");

        RecognizeCelebritiesResponse recognizeCelebritiesResponse =
 rekognitionClient.RecognizeCelebrities(recognizeCelebritiesRequest);

        Console.WriteLine(recognizeCelebritiesResponse.CelebrityFaces.Count + "
 celebrity(s) were recognized.\n");
        foreach (Celebrity celebrity in
 recognizeCelebritiesResponse.CelebrityFaces)
        {
            Console.WriteLine("Celebrity recognized: " + celebrity.Name);
            Console.WriteLine("Celebrity ID: " + celebrity.Id);
            BoundingBox boundingBox = celebrity.Face.BoundingBox;
            Console.WriteLine("position: " +
               boundingBox.Left + " " + boundingBox.Top);
            Console.WriteLine("Further information (if available):");
            foreach (String url in celebrity.Urls)
                Console.WriteLine(url);
        }
        Console.WriteLine(recognizeCelebritiesResponse.UnrecognizedFaces.Count +
 " face(s) were unrecognized.");
    }
}
```

3. Registra il valore di una delle celebrità IDs visualizzate. Questo valore sarà necessario nella Come recuperare le informazioni su una celebrità.

RecognizeCelebrities richiesta di operazione

L'input per `RecognizeCelebrities` è un'immagine. In questo esempio, l'immagine è passata come byte di immagine. Per ulteriori informazioni, consulta Lavorare con le immagini.

```
{
    "Image": {
        "Bytes": "/AoSiyvFpm....."
    }
}
```

RecognizeCelebrities risposta operativa

Di seguito è riportato un esempio di input e output JSON per `RecognizeCelebrities`.

`RecognizeCelebrities` restituisce un array di volti celebri riconosciuti e un array di volti non riconosciuti. Nell'esempio , tenere presente quanto segue:

- Celebrità riconosciute: `Celebrities` è una matrice di celebrità riconosciute. Ogni oggetto Celebrity nell'array contiene il nome della celebrità e un elenco di URLs riferimenti a contenuti correlati, ad esempio il link IMDB o Wikidata della celebrità. Amazon Rekognition ComparedFacerestituisce un oggetto che l'applicazione può utilizzare per determinare dove si trova il volto della celebrità sull'immagine e un identificatore univoco per la celebrità. Utilizza l'identificatore univoco per recuperare in seguito le informazioni sulla celebrità con l'operazione GetCelebrityInfo dell'API.
- Volti non riconosciuti: `UnrecognizedFaces` è una matrice di volti non corrispondenti a celebrità. Ogni oggetto ComparedFace nell'array contiene un riquadro di delimitazione (nonché altre informazioni) che può essere utilizzato per individuare il volto nell'immagine.

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

Riconoscimento di celebrità in un video archiviato

Il riconoscimento di volti celebri di Video Amazon Rekognition nei video archiviati è un'operazione asincrona. Per riconoscere le celebrità in un video archiviato, utilizzalo per avviare l'analisi video. StartCelebrityRecognition Amazon Rekognition per video pubblica lo stato di completamento dell'analisi video in un argomento Amazon Simple Notification Service. Se l'analisi video ha esito positivo, effettua la chiamata a GetCelebrityRecognition per ottenere i risultati dell'analisi. Per ulteriori informazioni su come avviare analisi video e ottenere i risultati, consultare Chiamata delle operazioni Video Amazon Rekognition.

La procedura si espande nel codice in Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK), che utilizza una coda di Amazon SQS per ottenere lo stato di completamento di una richiesta di analisi video. Per eseguire questa procedura, è necessario disporre di un file video contenente uno o più volti celebri.

Per individuare le celebrità in un video archiviato in un bucket Amazon S3 (SDK)

1. Eseguire Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK).
2. Aggiungere il seguente codice alla classe `VideoDetect` creata nella fase 1.

   Java

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

     //
 Celebrities=========================================================================
     private static void StartCelebrityDetection(String bucket, String video)
 throws Exception{

            NotificationChannel channel= new NotificationChannel()
```

```
                    .withSNSTopicArn(snsTopicArn)
                    .withRoleArn(roleArn);

          StartCelebrityRecognitionRequest req = new
StartCelebrityRecognitionRequest()
                .withVideo(new Video()
                      .withS3Object(new S3Object()
                            .withBucket(bucket)
                            .withName(video)))
                .withNotificationChannel(channel);



          StartCelebrityRecognitionResult startCelebrityRecognitionResult =
rek.startCelebrityRecognition(req);
          startJobId=startCelebrityRecognitionResult.getJobId();

       }

       private static void GetCelebrityDetectionResults() throws Exception{

          int maxResults=10;
          String paginationToken=null;
          GetCelebrityRecognitionResult celebrityRecognitionResult=null;

          do{
             if (celebrityRecognitionResult !=null){
                paginationToken = celebrityRecognitionResult.getNextToken();
             }
             celebrityRecognitionResult = rek.getCelebrityRecognition(new
GetCelebrityRecognitionRequest()
                    .withJobId(startJobId)
                    .withNextToken(paginationToken)
                    .withSortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                    .withMaxResults(maxResults));


             System.out.println("File info for page");
             VideoMetadata
videoMetaData=celebrityRecognitionResult.getVideoMetadata();

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

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

            System.out.println("Job");

            System.out.println("Job status: " +
celebrityRecognitionResult.getJobStatus());


            //Show celebrities
            List<CelebrityRecognition> celebs=
celebrityRecognitionResult.getCelebrities();

            for (CelebrityRecognition celeb: celebs) {
               long seconds=celeb.getTimestamp()/1000;
               System.out.print("Sec: " + Long.toString(seconds) + " ");
               CelebrityDetail details=celeb.getCelebrity();
               System.out.println("Name: " + details.getName());
               System.out.println("Id: " + details.getId());
               System.out.println();
            }
         } while (celebrityRecognitionResult !=null &&
celebrityRecognitionResult.getNextToken() != null);

      }
```

Nella funzione `main`, sostituisci la riga:

```
      StartLabelDetection(amzn-s3-demo-bucket, video);

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

con:

```
      StartCelebrityDetection(amzn-s3-demo-bucket, video);

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

Java V2

Questo codice è tratto dal repository degli esempi GitHub di AWS Documentation SDK. Guarda l'esempio completo qui.

```
//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
```

Nella funzione `main`, sostituisci le righe:

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

con:

```
    analyzer.StartCelebrityDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetCelebrityDetectionResults()
```

Node.JS

Nel seguente esempio di codice Node.Js, sostituisci il valore di `amzn-s3-demo-bucket` con il nome del bucket S3 contenente il tuo video e il valore di `videoName` con il nome del file video. Dovrai inoltre sostituire il valore di `roleArn` con l'ARN associato al tuo ruolo di servizio IAM. Infine, sostituisci il valore di `region` con il nome della regione operativa associata al tuo account. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

Esegui il AWS CLI comando seguente per iniziare a rilevare le celebrità in un video.

```
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
```

Aggiorna i seguenti valori:

- Modifica `amzn-s3-demo-bucket` e `video-name` con il nome del bucket Amazon S3 e il nome del file specificati nella fase 2.
- Cambia `region-name` con la regione AWS che stai utilizzando.
- Sostituisci il valore di `profile-name` con il nome del tuo profilo di sviluppatore.
- Cambia `topic-ARN` con l'ARN dell'argomento Amazon SNS creato nella fase 3 di Configurazione di Video Amazon Rekognition.
- Modifica `role-ARN` con l'ARN del ruolo di servizio IAM creato nella fase 7 di Configurazione di Video Amazon Rekognition.

Se accedi alla CLI da un dispositivo Windows, usa le virgolette doppie anziché le virgolette singole ed evita le virgolette doppie interne tramite barra rovesciata (ovvero, \) per risolvere eventuali errori del parser che potresti riscontrare. Un esempio è fornito di seguito:

```
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
```

Dopo aver eseguito l'esempio di codice precedente, copia il `jobID` restituito e inseriscilo nel comando `GetCelebrityRecognition` di seguito per ottenere i risultati, sostituendo `job-id-number` con il `jobID` ricevuto in precedenza:

```
aws rekognition get-celebrity-recognition --job-id job-id-number --profile
 profile-name
```

> **Note**
>
> Se hai già eseguito un video di esempio diverso da Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK), il codice da sostituire potrebbe essere diverso.

3. Eseguire il codice. Vengono visualizzate le informazioni sui volti celebri riconosciuti nel video.

GetCelebrityRecognition risposta all'operazione

Di seguito è riportata una risposta JSON di esempio. La risposta include quanto segue:

- Celebrità riconosciute: `Celebrities` è una matrice di celebrità e degli orari in cui vengono riconosciute in un video. Esiste un oggetto CelebrityRecognition per ogni ora in cui viene riconosciuto un volto celebre nel video. Ogni `CelebrityRecognition` contiene informazioni su un volto celebre riconosciuto (CelebrityDetail) e l'ora (`Timestamp`) in cui la celebrità è stata riconosciuta nel video. `Timestamp` si misura in millisecondi dall'inizio del video.

- CelebrityDetail— Contiene informazioni su una celebrità riconosciuta. Include il nome della celebrità (`Name`), l'identificatore (`ID`), il sesso noto della celebrità (`KnownGender`) e un elenco di riferimenti a contenuti correlati URLs (). `Urls` Include anche il livello di fiducia di Amazon Rekognition Video nell'accuratezza del riconoscimento e i dettagli sul volto della celebrità. FaceDetail Per ottenere il contenuto correlato in un secondo momento, puoi utilizzare `ID` con getCelebrityInfo.
- VideoMetadata— Informazioni sul video che è stato analizzato.


```
{
    "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
    }
}
```

Come recuperare le informazioni su una celebrità

In queste procedure puoi ottenere informazioni su un volto celebre utilizzando l'operazione getCelebrityInfo dell'API. Il volto celebre viene identificato utilizzando l'ID restituito da una precedente chiamata a RecognizeCelebrities.

Chiamata GetCelebrityInfo

Queste procedure richiedono l'ID celebrità per una celebrità conosciuta da Amazon Rekognition. Utilizza l'ID del volto celebre annotato in Riconoscimento delle celebrità in un'immagine.

Per recuperare le informazioni sul volto celebre (SDK)

1. Se non lo hai già fatto:
   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
   b. Installa e configura AWS CLI e AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Utilizzare i seguenti esempi per richiamare l'operazione `GetCelebrityInfo`.

   Java

   Questo esempio mostra il nome e le informazioni su un volto celebre.

   Sostituisci `id` con una delle celebrità IDs visualizzate inRiconoscimento delle celebrità in un'immagine.

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

Questo codice è tratto dal repository degli esempi GitHub di AWS Documentation SDK. Guarda l'esempio completo qui.

```
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

Questo AWS CLI comando visualizza l'output JSON per l'operazione `get-celebrity-info` CLI. `ID`Sostituiscilo con una delle celebrità visualizzate IDs in. Riconoscimento delle celebrità in un'immagine Sostituisci il valore di `profile-name` con il nome del tuo profilo di sviluppatore.

```
aws rekognition get-celebrity-info --id celebrity-id --profile profile-name
```

Python

Questo esempio mostra il nome e le informazioni su un volto celebre.

Sostituiscilo `id` con una delle celebrità IDs visualizzate in. Riconoscimento delle celebrità in un'immagine Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

Questo esempio mostra il nome e le informazioni su un volto celebre.

Sostituiscilo `id` con una delle celebrità IDs visualizzate in. Riconoscimento delle celebrità in un'immagine

```
//Copyright 2018 Amazon.com, Inc. or its 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 richiesta di operazione

Di seguito è riportato un esempio di input e output JSON per `GetCelebrityInfo`.

L'input per `GetCelebrityInfo` è l'ID del volto celebre richiesto.

```
{
    "Id": "nnnnnnn"
}
```

GetCelebrityInfo risposta operativa

`GetCelebrityInfo` restituisce un array (`Urls`) di collegamenti a informazioni sui volti celebri richiesti.

```
{
    "Name": "Celebrity Name",
    "Urls": [
        "www.imdb.com/name/nmnnnnnnn"
```

```
    ]
}
```

Moderazione dei contenuti

Puoi usare Amazon Rekognition per rilevare contenuti inappropriati, indesiderati o offensivi. Puoi utilizzare la APIs moderazione di Rekognition nei social media, nei media radiotelevisivi, nella pubblicità e nell'e-commerce per creare un'esperienza utente più sicura, fornire garanzie di sicurezza del marchio agli inserzionisti e rispettare le normative locali e globali.

Oggi, molte aziende si affidano esclusivamente a moderatori umani per esaminare i contenuti di terze parti o generati dagli utenti, mentre altre reagiscono semplicemente ai reclami degli utenti rimuovendo immagini, annunci o video offensivi o inappropriati. Tuttavia, i moderatori umani da soli non sono in grado di soddisfare queste esigenze con una qualità o una velocità sufficienti, il che comporta un'esperienza utente scadente, costi elevati per raggiungere la scalabilità o persino una perdita di reputazione del marchio. Utilizzando Rekognition per la moderazione di immagini e video, i moderatori umani possono esaminare un set di contenuti molto più piccolo, in genere dall'1 al 5% del volume totale, già contrassegnato dal machine learning. Ciò consente loro di concentrarsi su attività più importanti e di ottenere comunque una copertura di moderazione completa a una frazione dei costi esistenti. Per configurare forza lavoro umana ed eseguire attività di revisione umana, puoi utilizzare IA aumentata Amazon, che è già integrato con Rekognition.

Puoi migliorare la precisione del modello di deep learning di moderazione con la funzionalità di moderazione personalizzata. Con la moderazione personalizzata, puoi addestrare un adattatore di moderazione personalizzato caricando le tue immagini e annotandole. L'adattatore qualificato può quindi essere fornito all'DetectModerationLabelsoperazione per migliorarne le prestazioni sulle immagini. Per ulteriori informazioni, consulta Migliorare la precisione con la moderazione personalizzata.

Etichette supportate dalle operazioni di moderazione dei contenuti di Rekognition

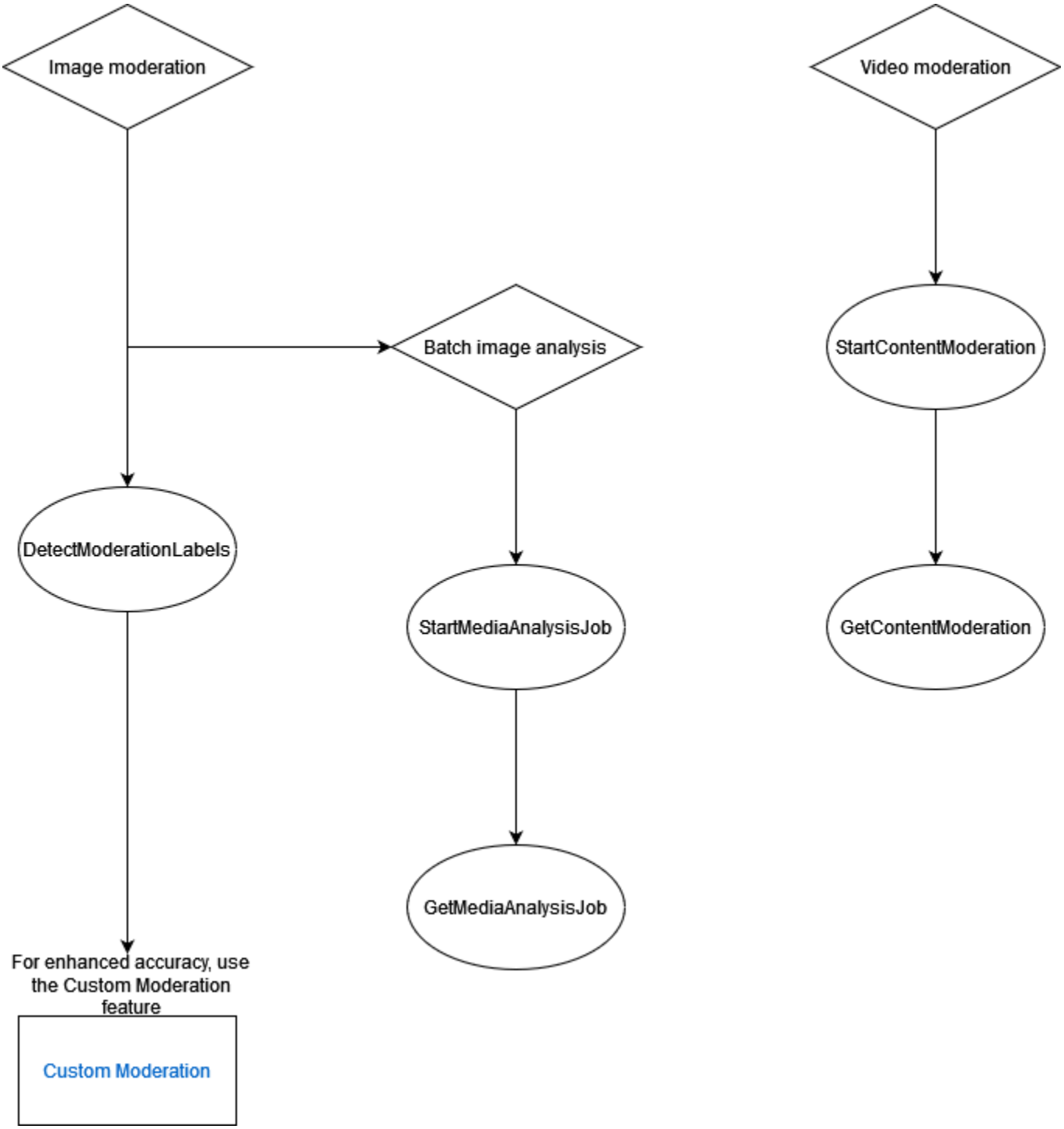
- Per scaricare un elenco delle etichette di moderazione, clicca qui.

Argomenti

- Utilizzo della moderazione di immagini e video APIs
- Test della versione 7 di Content Moderation e trasformazione della risposta dell'API
- Rilevamento di immagini inappropriate
- Rilevamento di video archiviati in modo inappropriato
- Migliorare la precisione con la moderazione personalizzata

- Revisione di contenuti non appropriati con Amazon Augmented AI

Il diagramma seguente mostra l'ordine delle operazioni di chiamata, a seconda degli obiettivi prefissati per l'utilizzo dei componenti immagine o video di Content Moderation:



Utilizzo della moderazione di immagini e video APIs

Nell'API Amazon Rekognition Image, puoi rilevare contenuti inappropriati, indesiderati o offensivi utilizzando e operando in modo sincrono e DetectModerationLabelsasincrono. StartMediaAnalysisJobGetMediaAnalysisJob Puoi utilizzare l'API Amazon Rekognition Video per rilevare tali contenuti in modo asincrono utilizzando le operazioni and. StartContentModerationGetContentModeration

Categorie di etichette

Amazon Rekognition utilizza una tassonomia gerarchica a tre livelli per etichettare le categorie di contenuti inappropriati, indesiderati o offensivi. Ogni etichetta con livello di tassonomia 1 (L1) ha diverse etichette di livello di tassonomia 2 (L2) e alcune etichette di livello di tassonomia 2 possono avere etichette di livello di tassonomia 3 (L3). Ciò consente una classificazione gerarchica del contenuto.

Per ogni etichetta di moderazione rilevata, l'API restituisce anche il`TaxonomyLevel`, che contiene il livello (1, 2 o 3) a cui appartiene l'etichetta. Ad esempio, un'immagine può essere etichettata in base alla seguente categorizzazione:

L1: Nudità non esplicita di parti intime e baci, L2: Nudità non esplicita, L3: nudità implicita.

> **Note**
>
> Ti consigliamo di utilizzare le categorie L1 o L2 per moderare i tuoi contenuti e di utilizzare le categorie L3 solo per rimuovere concetti specifici che non desideri moderare (ad esempio per rilevare contenuti che potresti non voler classificare come contenuti inappropriati, indesiderati o offensivi in base alla tua politica di moderazione).

La tabella seguente mostra le relazioni tra i livelli di categoria e le possibili etichette per ogni livello. Per scaricare un elenco delle etichette di moderazione, fai clic qui.

Categoria di primo livello (L1)	Categoria di secondo livello (L2)	Categoria di terzo livello (L3)	Definizioni
Esplicito	Nudità esplicita	Genitali maschili esposti	Genitali umani maschili, compreso

		il pene (eretto o flaccido), lo scroto e tutti i peli pubici visibili. Questo termine è applicabi le in contesti che coinvolgono l'attività sessuale o qualsiasi contenuto visivo in cui i genitali maschili vengono visualizz ati completamente o parzialmente.
	Genitali femminili esposti	Parti esterne del sistema riprodutt ivo femminile, che comprendono la vulva, la vagina e tutti i peli pubici visibili. Questo termine è applicabile in scenari che coinvolgono l'attività sessuale o qualsiasi contenuto visivo in cui questi aspetti dell'anat omia femminile vengono visualizz ati completamente o parzialmente.

Glutei o ano esposti	Glutei o ano umani, compresi i casi in cui i glutei sono nudi o quando sono distinguibili attravers o abiti trasparenti. La definizione si applica specificamente alle situazioni in cui i glutei o l'ano sono direttame nte e completamente visibili, esclusi i casi in cui qualsiasi forma di biancheria intima o abbigliamento offra una copertura totale o parziale.
Capezzolo femminile esposto	Capezzoli femminili umani, compresi l'aerola (area che circonda i capezzoli) e i capezzoli completamente visibili e parzialmente visibili.

Attività sessuale esplicita	N/D	Rappresentazione di atti sessuali reali o simulati che comprendono rapporti sessuali umani, sesso orale, nonché la stimolazione genitale maschile e la stimolazione genitale femminile da parte di altre parti e oggetti del corpo. Il termine include anche l'eiaculazione o i fluidi vaginali sulle parti del corpo e le pratiche erotiche o i giochi di ruolo che implicano schiavitù, disciplina, dominanza e sottomissione e sadomasochismo.
Giocattoli sessuali	N/D	Oggetti o dispositivi utilizzati per la stimolazione o il piacere sessuale, ad esempio dildo, vibratore, plug anale, battiti, ecc.

Nudità non esplicita di parti intime e baci	Nudità non esplicita	Schiena nuda	Parte posteriore umana dove la maggior parte della pelle è visibile dal collo all'estremità della colonna vertebrale. Questo termine non si applica quando la schiena dell'individuo è parzialmente o completamente occlusa.
		Capezzolo maschile esposto	Capezzoli umani maschili, compresi i capezzoli parzialmente visibili.
		Glutei parzialmente esposti	Glutei umani parzialmente esposti. Questo termine include una regione parzialmente visibile dei glutei o delle guance a causa di abiti corti, o la parte superiore parzialmente visibile della fessura anale. Il termine non si applica ai casi in cui i glutei sono completamente nudi.

		Seno femminile parzialmente esposto	Seno femminile umano parzialmente esposto in cui una parte del seno femminile è visibile o scoperta ma non rivela l'intero seno. Questo termine si applica quando la regione della piega interna del seno è visibile o quando la piega inferiore del seno è visibile con il capezzolo completamente coperto o occluso.
		Nudità implicita	Un individuo nudo, in topless o senza fondo, ma con parti intime come glutei, capezzoli o genitali coperte, occluse o non completamente visibili.
	Parti intime ostruite	Capezzolo femminile ostruito	Rappresentazione visiva di una situazione in cui i capezzoli di una donna sono coperti da indumenti o rivestimenti opachi, ma le loro forme sono chiaramente visibili.

		Genitali maschili ostruiti	Rappresentazione visiva di una situazione in cui i genitali o il pene di un uomo sono coperti da indumenti o rivestimenti opachi, ma la sua forma è chiaramente visibile. Questo termine si applica quando i genitali ostruiti nell'immagine sono ravvicinati.
	Baciarsi sulle labbra	N/D	Rappresentazione delle labbra di una persona che entrano in contatto con le labbra di un'altra persona.
Costumi da bagno o biancheria intima	Costumi da bagno o biancheria intima femminili	N/D	Abbigliamento umano per costumi da bagno femminili (ad es. costumi da bagno interi, bikini, tankini, ecc.) e biancheria intima femminile (ad esempio reggiseni, mutandine, slip, lingerie, perizoma, ecc.)

	Costumi da bagno o biancheria intima maschili	N/D	Abbigliamento umano per costumi da bagno maschili (ad es. costume da bagno, pantaloncini da surf, slip da bagno, ecc.) e biancheria intima maschile (ad es. slip, boxer, ecc.)
Violenza	Armi	N/D	Strumenti o dispositivi utilizzati per causare danni a esseri viventi, strutture o sistemi. Ciò include armi da fuoco (ad esempio pistole, fucili, mitraglia trici, ecc.), armi affilate (ad esempio spade, coltelli, ecc.), esplosivi e munizioni (ad esempio missili, bombe, proiettili, ecc.).
	Violenza grafica	Violenza con armi	L'uso di armi per causare danni, lesioni o morte a se stessi, ad altri individui o proprietà.

Violenza fisica	L'atto di arrecare danno ad altre persone o proprietà (ad esempio, colpire, combattere, tirarsi i capelli, ecc.) o altro atto di violenza che coinvolge la folla o più individui.
Autolesionismo	L'atto di farsi del male, spesso tagliando parti del corpo come braccia o gambe, dove i tagli sono generalmente visibili.
Sangue e sangue	Rappresentazione visiva della violenza su una persona, un gruppo di individui o animali, con ferite aperte, spargimenti di sangue e parti del corpo mutilate.
Esplosioni ed esplosioni	Raffigurazione di un'esplosione violenta e distruttiva di fiamme intense con fumo denso o polvere e fumo che esplode dal terreno.

Immagini forti	Morte ed emaciazione	Corpi emaciati	Corpi umani estremamente magri e denutriti con grave deperimento fisico e esaurimento dei muscoli e dei tessuti adiposi.
		Cadaveri	Cadaveri umani sotto forma di corpi mutilati, cadaveri impiccati o scheletri.
	Si schianta	Incidente aereo	Incidenti con veicoli aerei, come aeroplani , elicotteri o altri veicoli volanti, con conseguenti danni, lesioni o morte. Questo termine si applica quando parti dei veicoli aerei sono visibili.
Droghe e tabacco	Prodotti	Pillole	Tavoli o capsule piccoli, solidi, spesso rotondi o ovali. Questo termine si applica alle pillole presentate singolarm ente, in un flacone o in una bustina trasparente e non si applica alla rappresen tazione visiva di una persona che assume pillole.

	Farmaci e accessori e uso del tabacco	Fumare	L'atto di inalare, espirare e accendere sostanze che bruciano tra cui sigarette, sigari, sigarette elettroniche, narghilè o spinelli.
Alcol	Uso di alcol	Bere	L'atto di bere bevande alcoliche da bottiglie o bicchieri di alcol o liquori.
	Bevande alcoliche	N/D	Primo piano di una o più bottiglie di alcol o liquore, bicchieri o tazze con alcol o liquore e bicchieri o tazze con alcol o liquore detenuti da una persona. Questo termine non si applica a un individuo che beve da bottiglie o bicchieri di alcol o liquori.
Gesti maleducati	Dito medio	N/D	La rappresentazione visiva di un gesto della mano con il dito medio viene estesa verso l'alto mentre le altre dita sono piegate verso il basso.

Gioco d'azzardo	N/D	N/D	L'atto di partecipare a giochi d'azzardo per avere la possibilità di vincere un premio nei casinò, ad esempio carte da gioco, blackjack, roulette, slot machine nei casinò, ecc.
Simboli di odio	Partito nazista	N/D	Rappresentazione visiva di simboli, bandiere o gesti associati al partito nazista.
	Supremazia bianca	N/D	Rappresentazione visiva di simboli o abiti associati al Ku Klux Klan (KKK) e immagini con bandiere confederate.
	Estremista	N/D	Immagini contenenti bandiere di gruppi estremisti e terroristici.

Non tutte le etichette della categoria L2 hanno un'etichetta supportata nella categoria L3. Inoltre, le etichette L3 sotto le etichette «Prodotti» e «Accessori e uso di droghe e tabacco» L2 non sono esaustive. Queste etichette L2 coprono concetti che vanno oltre le suddette etichette L3 e in tali casi, nella risposta API vengono restituite solo le etichette L2.

Puoi determinare l'idoneità dei contenuti in base all'applicazione. Ad esempio, le immagini con contenuti spinti possono essere accettabili, ma le immagini contenenti nudità no. Per filtrare le immagini, utilizza la matrice di etichette ModerationLabel restituita da `DetectModerationLabels` (immagini) e da `GetContentModeration` (video).

Tipo di contenuto

L'API può anche identificare il tipo di contenuto animato o illustrato e il tipo di contenuto viene restituito come parte della risposta:

- I contenuti animati includono videogiochi e animazioni (ad esempio cartoni animati, fumetti, manga, anime).
- I contenuti illustrati includono disegni, dipinti e schizzi.

Confidence

È possibile impostare la soglia di fiducia utilizzata da Amazon Rekognition per rilevare contenuto inappropriato specificando il parametro di input `MinConfidence`. Le etichette non vengono restituite per i contenuti inappropriati rilevati con un livello di affidabilità inferiore a `MinConfidence`.

Specificando un valore inferiore al 50% è probabile `MinConfidence` che si ottengano un numero elevato di risultati falsi positivi (ad esempio un richiamo più elevato, una precisione inferiore). D'altra parte, specificando un valore `MinConfidence` superiore al 50% è probabile che si ottenga un numero inferiore di risultati falsi positivi (ad esempio un richiamo inferiore, una precisione maggiore). Se non si specifica un valore per `MinConfidence`, Amazon Rekognition restituisce etichette per contenuto inappropriato rilevate con almeno un'attendibilità del 50%.

La matrice `ModerationLabel` contiene etichette nelle categorie precedenti e una stima dell'affidabilità sulla precisione dei contenuti riconosciuti. Un'etichetta di primo livello viene restituita insieme alle etichette di secondo livello identificate. Ad esempio, Amazon Rekognition può restituire "Nudità esplicita" con un punteggio di affidabilità elevato, come un'etichetta di primo livello. Questo potrebbe essere sufficiente per le tue esigenze di filtraggio. Tuttavia, se necessario, puoi utilizzare il punteggio di affidabilità di un'etichetta di secondo livello (ad esempio "Graphic Male Nudity" (Nudità maschile grafica)) per ottenere un filtro più granulare. Per vedere un esempio, consulta Rilevamento di immagini inappropriate.

Controllo delle versioni

Immagini Amazon Rekognition e Video Amazon Rekognition restituiscono entrambi la versione del modello di rilevamento della moderazione utilizzato per rilevare contenuti inappropriati (`ModerationModelVersion`).

Ordinamento e aggregazione

Quando recuperi i risultati con GetContentModeration, puoi ordinare e aggregare i risultati.

Ordinamento - La matrice di etichette restituite viene ordinata in base all'ora. Per ordinare in base all'etichetta, specificare `NAME` nel parametro di input `SortBy` per `GetContentModeration`. Se l'etichetta viene visualizzata più volte nel video, saranno presenti più istanze dell'elemento `ModerationLabel`.

Informazioni sull'etichetta: l'elemento dell' ModerationLabelsarray contiene un `ModerationLabel` oggetto, che a sua volta contiene il nome dell'etichetta e la fiducia che Amazon Rekognition ha nell'accuratezza dell'etichetta rilevata. Il timestamp è l'ora in cui è `ModerationLabel` stato rilevato, definito come il numero di millisecondi trascorsi dall'inizio del video. Per i risultati aggregati per video`SEGMENTS`, vengono restituite le strutture `StartTimestampMillis`, `EndTimestampMillis` e `DurationMillis` che definiscono rispettivamente l'ora di inizio, l'ora di fine e la durata di un segmento.

Aggregazione: specifica in che modo i risultati vengono aggregati quando vengono restituiti. L'impostazione predefinita prevede l'aggregazione entro `TIMESTAMPS`. Puoi anche scegliere di aggregare per`SEGMENTS`, che aggrega i risultati in una finestra temporale. Vengono restituite solo le etichette rilevate durante i segmenti.

Stati dell'adattatore di moderazione personalizzato

Gli adattatori di moderazione personalizzati possono avere uno dei seguenti stati: TRAINING_IN_PROGRESS, TRAINING_COMPLETED, TRAINING_FAILED, DELETING, DEPRECATED o EXPIRED. Per una spiegazione completa di questi stati degli adattatori, vedere Gestione degli adattatori.

> **Note**
>
> Amazon Rekognition non è un'autorità su e non dichiara in alcun modo di essere un filtro esaustivo di contenuti inappropriato o offensivi. Inoltre, la moderazione di immagini e video APIs non rileva se un'immagine include contenuti illegali, come CSAM.

Test della versione 7 di Content Moderation e trasformazione della risposta dell'API

Rekognition ha aggiornato il modello di apprendimento automatico per i componenti di immagini e video della funzione di rilevamento delle etichette Content Moderation dalla versione 6.1 alla 7. Questo aggiornamento ha migliorato la precisione complessiva e ha introdotto diverse nuove categorie oltre a modificarne altre.

Se sei già un utente video della versione 6.1, ti consigliamo di intraprendere le seguenti azioni per passare senza problemi alla versione 7:

1. Scarica e usa un SDK privato AWS (vedithe section called “AWS SDK e guida all'uso per la moderazione dei contenuti versione 7 ”) per chiamare l' StartContentModeration API.
2. Consulta l'elenco aggiornato delle etichette e dei punteggi di affidabilità restituiti nella risposta o nella console dell'API. Se necessario, modificate di conseguenza la logica di post-elaborazione dell'applicazione.
3. Il tuo account rimarrà nella versione 6.1 fino al 13 maggio 2024. Se desideri utilizzare la versione 6.1 oltre il 13 maggio 2024, contatta il team di AWS Support entro il 30 aprile 2024 per richiedere un'estensione. Possiamo estendere il tuo account in modo che rimanga sulla versione 6.1 fino al 10 giugno 2024. Se non riceveremo alcuna risposta entro il 30 aprile 2024, il tuo account verrà migrato automaticamente alla versione 7.0 a partire dal 13 maggio 2024.

AWS SDK e guida all'uso per la moderazione dei contenuti versione 7

Scarica l'SDK corrispondente al linguaggio di sviluppo scelto e consulta la guida per l'utente appropriata.

Collegamento all'SDK	Guida all'installazione/guida per l'utente
Java-1.X	Guida - Java 1.pdf
Java-2.X	Guida - Java 2.pdf
JavaScript v2	Guida - JavaScript v2.pdf
JavaScript v3	Guida - JavaScript v3.pdf

Python	Guida - Python e AWS CLI.pdf
Ruby	Guida - RubyV3.pdf
go_v1	Guida - GO V1.pdf
go_v2	Guida - GO V2.pdf
DotNet	Guida - .net.pdf
php	Guida - PHP.pdf

Mappature delle etichette per le versioni da 6.1 a 7

La versione 7 di moderazione dei contenuti ha aggiunto nuove categorie di etichette e modificato i nomi delle etichette esistenti in precedenza. Fai riferimento alla tabella della tassonomia che trovi in the section called “ Categorie di etichette ” quando decidi come mappare le etichette 6.1 a 7 etichette.

Alcuni esempi di mappatura delle etichette sono disponibili nella sezione seguente. Ti consigliamo di esaminare queste mappature e le definizioni delle etichette prima di apportare gli aggiornamenti necessari in base alla logica di post-elaborazione dell'applicazione.

Schema di mappatura L1

Se utilizzi una logica di post-elaborazione che filtra solo in base alla categoria di primo livello (L1) (ad esempio, `Violence` ecc.) `Explicit Nudity``Suggestive`, consulta la tabella seguente per aggiornare il codice.

V6.1 L1	V7 L1
Nudità esplicita	Esplicito
Contenuti allusivi	Nudità non esplicita di parti intime e baci
	Costumi da bagno o biancheria intima
Violenza	Violenza
Immagini forti	Immagini forti

Gesti maleducati	Gesti maleducati
Droghe	Droghe e tabacco
Tabacco	Droghe e tabacco
Alcol	Alcol
Gioco d'azzardo	Gioco d'azzardo
Simboli di odio	Simboli di odio

Schema di mappatura L2

Se utilizzi una logica di post-elaborazione che filtra in base a entrambe le categorie L1 e L2 (ad esempio`Explicit Nudity / Nudity, Suggestive / Female Swimwear Or Underwear`, `Violence / Weapon Violence` ecc.), consulta la tabella seguente per aggiornare il codice.

V6.1 L1	V6.1 L2	V7 L1	V7 L2	V7 L3	V7 ContentTypes
Nudità esplicita	Nudità	Esplicito	Nudità esplicita	Capezzolo femminile esposto	
				Glutei o ano esposti	
	Contenuti sessuali espliciti (maschili)	Esplicito	Nudità esplicita	Genitali maschili esposti	
	Contenuti sessuali espliciti (femminili)	Esplicito	Nudità esplicita	Genitali femminili esposti	

	Atti sessuali	Esplicito	Attività sessuale esplicita		
	Nudità esplicita illustrata	Esplicito	Nudità esplicita		Mappa su «Animato» e «Illustrato»
	Nudità esplicita illustrata	Esplicito	Attività sessuale esplicita		Mappa su «Animato» e «Illustrato»
	Sex toy	Esplicito	Giocattoli sessuali		
Contenuti allusivi	Costumi da bagno o biancheria femminile	Costumi da bagno o biancheria intima	Costumi da bagno o biancheria intima femminili		
	Costumi da bagno o biancheria maschile	Costumi da bagno o biancheria intima	Costumi da bagno o biancheria intima da uomo		
	Nudità parziale	Nudità non esplicita di parti intime e baci	Nudità non esplicita	Nudità implicita	
	Maschio a petto nudo	Nudità non esplicita di parti intime e baci	Nudità non esplicita	Capezzolo maschile esposto	

	Abbigliamento provocante	Nudità non esplicita di parti intime e baci	Nudità non esplicita	
		Nudità non esplicita di parti intime e baci	Parti intime ostruite	
	Situazioni sessuali	Nudità non esplicita di parti intime e baci	Baciarsi sulle labbra	
Violenza	Violenza o sangue grafici	Violenza	Violenza grafica	Sangue e sangue
	Violenza fisica	Violenza	Violenza grafica	Violenza fisica
	Violenza con armi	Violenza	Violenza grafica	Violenza con armi
	Armi	Violenza	Armi	
	Autolesionismo	Violenza	Violenza grafica	Autolesionismo
Immagini forti	Corpi emaciati	Immagini forti	Morte ed emaciazione	Corpi emaciati
	Cadaveri	Immagini forti	Morte ed emaciazione	Cadaveri
	Impiccagioni	Immagini forti	Morte ed emaciazione	Cadaveri

	Incidente aereo	Immagini forti	Si schianta	Incidente aereo
	Esplosioni e deflagrazioni	Violenza	Violenza grafica	Esplosioni ed esplosioni
Gesti maleducati	Dito medio	Gesti maleducati	Dito medio	
Droghe	Prodotti farmaceutici	Droghe e tabacco	Prodotti	
	Uso di droghe	Droghe e tabacco	Farmaci e accessori per il consumo di tabacco	
	Pillole	Droghe e tabacco	Prodotti	Pillole
	Accessori correllati all'uso di droghe	Droghe e tabacco	Farmaci e accessori per il consumo di tabacco	
Tabacco	Prodotti del tabacco	Droghe e tabacco	Prodotti	
	Fumare	Droghe e tabacco	Farmaci e accessori per il consumo di tabacco	Fumare
Alcol	Bere	Alcol	Uso di alcol	Bere
	Bevande alcoliche	Alcol	Bevande alcoliche	

Gioco d'azzardo	Gioco d'azzardo	Gioco d'azzardo	
Simboli di odio	Partito nazista	Simboli di odio	Partito nazista
	Supremazia bianca	Simboli di odio	Supremazia bianca
	Estremista	Simboli di odio	Estremista

Rilevamento di immagini inappropriate

È possibile utilizzare l'DetectModerationLabelsoperazione per determinare se un'immagine contiene contenuti inappropriati o offensivi. Per un elenco delle etichette di moderazione in Amazon Rekognition, consulta Utilizzo della moderazione di immagini e video. APIs

Rilevamento di contenuti inappropriati in un'immagine

L'immagine deve essere un file in formato .png o .jpg. Puoi fornire un'immagine di input come matrice di byte dell'immagine (byte dell'immagine codificata in formato Base64) o specificare un oggetto di Amazon S3. Nelle procedure descritte viene caricata un'immagine (.jpg o .png) nel bucket S3.

Per eseguire queste procedure, devi avere installato l'SDK AWS CLI o l'SDK appropriato. AWS Per ulteriori informazioni, consulta Nozioni di base su Amazon Rekognition. L'account AWS utilizzato deve disporre di autorizzazioni di accesso all'API Amazon Rekognition. Per ulteriori informazioni, consulta la sezione Operazioni definite da Amazon Rekognition.

Per rilevare le etichette di moderazione in un'immagine (SDK)

1. Se non lo hai già fatto:

   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.
   b. Installa e configura il AWS CLI e il AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Carica un'immagine nel bucket S3.

   Per le istruzioni, consulta Caricamento di oggetti in Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

3. Utilizzare i seguenti esempi per richiamare l'operazione `DetectModerationLabels`.

   Java

   In questo esempio vengono forniti i nomi di etichetta di contenuti inappropriati, i livelli di affidabilità e l'etichetta padre per le etichette di moderazione rilevate.

   Sostituisci i valori di `amzn-s3-demo-bucket` e `photo` con il nome del bucket S3 e il nome del file di immagine utilizzato nella fase 2.

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

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.DetectModerationLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectModerationLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.ModerationLabel;
import com.amazonaws.services.rekognition.model.S3Object;

import java.util.List;

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

      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

      DetectModerationLabelsRequest request = new
 DetectModerationLabelsRequest()
```

```
        .withImage(new Image().withS3Object(new
 S3Object().withName(photo).withBucket(bucket)))
        .withMinConfidence(60F);
      try
      {
           DetectModerationLabelsResult result =
 rekognitionClient.detectModerationLabels(request);
           List<ModerationLabel> labels = result.getModerationLabels();
           System.out.println("Detected labels for " + photo);
           for (ModerationLabel label : labels)
           {
              System.out.println("Label: " + label.getName()
               + "\n Confidence: " + label.getConfidence().toString() + "%"
               + "\n Parent:" + label.getParentName());
          }
       }
       catch (AmazonRekognitionException e)
       {
         e.printStackTrace();
       }
    }
}
```

Java V2

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
//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

Questo AWS CLI comando visualizza l'output JSON per l'operazione `detect-moderation-labels` CLI.

Sostituisci `amzn-s3-demo-bucket` e `input.jpg` con il nome del bucket S3 e il nome del file di immagine utilizzato nella fase 2. Sostituisci il valore di `profile_name` con il nome

del tuo profilo di sviluppatore. Per utilizzare un adattatore, fornite l'ARN della versione del progetto al parametro `project-version`.

```
aws rekognition detect-moderation-labels --image "{S3Object:{Bucket:<amzn-s3-
demo-bucket>,Name:<image-name>}}" \
--profile profile-name \
--project-version "ARN"
```

Se accedi alla CLI da un dispositivo Windows, usa le virgolette doppie anziché le virgolette singole ed evita le virgolette doppie interne tramite barra rovesciata (ovvero, \) per risolvere eventuali errori del parser che potresti riscontrare. Per un esempio, consulta quanto segue:

```
aws rekognition detect-moderation-labels --image "{\"S3Object\":{\"Bucket\":
\"amzn-s3-demo-bucket\",\"Name\":\"image-name\"}}" \
--profile profile-name
```

Python

In questo esempio vengono forniti i nomi di etichetta di contenuti inappropriati o offensivi, i livelli di affidabilità e l'etichetta padre per le etichette di contenuto inappropriato rilevate.

Nella funzione `main`, sostituisci i valori di `amzn-s3-demo-bucket` e `photo` con i nomi del bucket S3 e dell'immagine utilizzati nella fase 2. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
#Copyright 2018 Amazon.com, Inc. or its 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 questo esempio vengono forniti i nomi di etichetta di contenuti inappropriati o offensivi, i livelli di affidabilità e l'etichetta padre per le etichette di moderazione rilevate.

Sostituisci i valori di `amzn-s3-demo-bucket` e `photo` con il nome del bucket S3 e il nome del file di immagine utilizzato nella fase 2.

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

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

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

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        DetectModerationLabelsRequest detectModerationLabelsRequest = new
 DetectModerationLabelsRequest()
        {
```

```
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket
                },
            },
            MinConfidence = 60F
        };

        try
        {
            DetectModerationLabelsResponse detectModerationLabelsResponse =
 rekognitionClient.DetectModerationLabels(detectModerationLabelsRequest);
            Console.WriteLine("Detected labels for " + photo);
            foreach (ModerationLabel label in
 detectModerationLabelsResponse.ModerationLabels)
                Console.WriteLine("Label: {0}\n Confidence: {1}\n Parent: {2}",
                    label.Name, label.Confidence, label.ParentName);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```

DetectModerationLabels richiesta di operazione

L'input per `DetectModerationLabels` è un'immagine. In questo input JSON di esempio, l'immagine di origine viene caricata da un bucket Amazon S3. `MinConfidence` è l'affidabilità minima che Immagini Amazon Rekognition deve avere nella precisione dell'etichetta rilevata per ottenerne la restituzione nella risposta.

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

```
    "MinConfidence": 60
}
```

DetectModerationLabels risposta operativa

`DetectModerationLabels` è in grado di recuperare le immagini di input da un bucket S3; in alternativa, è possibile fornirle come byte di immagine. Di seguito è riportato un esempio di risposta generata dalla chiamata all'operazione `DetectModerationLabels`.

Nel seguente esempio di risposta JSON, notare quanto segue:

- Informazioni sul rilevamento di immagini inappropriate: l'esempio mostra un elenco di etichette per i contenuti inappropriati o offensivi presenti nell'immagine. L'elenco include l'etichetta di primo livello e ogni etichetta di secondo livello rilevate nell'immagine.

  Etichetta – Ogni etichetta presenta un nome, una stima del livello di affidabilità di Amazon Rekognition circa la precisione dell'etichetta e il nome della relativa etichetta padre. Il nome dell'elemento padre per un'etichetta di primo livello è `""`.

  Affidabilità etichetta – Ogni etichetta presenta un valore di affidabilità compreso tra 0 e 100 che indica la percentuale di affidabilità di Amazon Rekognition circa la correttezza dell'etichetta. È necessario specificare il livello di affidabilità necessario affinché un'etichetta sia restituita nella risposta nella richiesta di operazione API.

```
{
    "ModerationLabels": [
        {
            "Confidence": 99.44782257080078,
            "Name": "Smoking",
            "ParentName": "Drugs & Tobacco Paraphernalia & Use",
            "TaxonomyLevel": 3
        },
        {
            "Confidence": 99.44782257080078,
            "Name": "Drugs & Tobacco Paraphernalia & Use",
            "ParentName": "Drugs & Tobacco",
            "TaxonomyLevel": 2
        },
        {
            "Confidence": 99.44782257080078,
```

```
            "Name": "Drugs & Tobacco",
            "ParentName": "",
            "TaxonomyLevel": 1
        }
    ],
    "ModerationModelVersion": "7.0",
    "ContentTypes": [
        {
            "Confidence": 99.9999008178711,
            "Name": "Illustrated"
        }
    ]
}
```

Rilevamento di video archiviati in modo inappropriato

Il rilevamento di contenuto inappropriato o offensivo di Video Amazon Rekognition nei video archiviati è un'operazione asincrona. Per iniziare a rilevare contenuti inappropriati o offensivi, chiama. StartContentModeration Amazon Rekognition per video pubblica lo stato di completamento dell'analisi video in un argomento Amazon Simple Notification Service. Se l'analisi video ha esito positivo, chiama GetContentModeration per ottenere i risultati dell'analisi. Per ulteriori informazioni su come avviare analisi video e ottenere i risultati, consultare Chiamata delle operazioni Video Amazon Rekognition. Per un elenco delle etichette di moderazione in Amazon Rekognition, consulta Utilizzo della moderazione di immagini e video. APIs

La procedura si espande nel codice in Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK), che utilizza una coda di Amazon Simple Queue Service per ottenere lo stato di completamento di una richiesta di analisi video.

Per rilevare contenuti non appropriati o offensivi in un video archiviato in un bucket Amazon S3 (SDK)

1. Eseguire Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK).
2. Aggiungere il seguente codice alla classe `VideoDetect` creata nella fase 1.

   Java

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

```
       //Content moderation
==========================================================================
       private static void StartUnsafeContentDetection(String bucket, String
video) throws Exception{

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

           StartContentModerationRequest req = new
StartContentModerationRequest()
                   .withVideo(new Video()
                           .withS3Object(new S3Object()
                               .withBucket(bucket)
                               .withName(video)))
                   .withNotificationChannel(channel);



            StartContentModerationResult startModerationLabelDetectionResult =
rek.startContentModeration(req);
            startJobId=startModerationLabelDetectionResult.getJobId();

        }

        private static void GetUnsafeContentDetectionResults() throws
Exception{

            int maxResults=10;
            String paginationToken=null;
            GetContentModerationResult moderationLabelDetectionResult =null;

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

                moderationLabelDetectionResult = rek.getContentModeration(
                        new GetContentModerationRequest()
                            .withJobId(startJobId)
                            .withNextToken(paginationToken)
                            .withSortBy(ContentModerationSortBy.TIMESTAMP)
                            .withMaxResults(maxResults));
```

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

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


                //Show moderated content labels, confidence and detection
times
                List<ContentModerationDetection> moderationLabelsInFrames=
                        moderationLabelDetectionResult.getModerationLabels();

                for (ContentModerationDetection label:
moderationLabelsInFrames) {
                    long seconds=label.getTimestamp()/1000;
                    System.out.print("Sec: " + Long.toString(seconds));
                    System.out.println(label.getModerationLabel().toString());
                    System.out.println();
                }
            } while (moderationLabelDetectionResult !=null &&
moderationLabelDetectionResult.getNextToken() != null);

        }
```

Nella funzione `main`, sostituisci le righe:

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

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

con:

```
        StartUnsafeContentDetection(amzn-s3-demo-bucket, video);
```

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

Java V2

Questo codice è tratto dall'archivio degli esempi di Documentation SDK AWS . GitHub Guarda l'esempio completo qui.

```
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
```

Nella funzione `main`, sostituisci le righe:

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

con:

```
    analyzer.StartUnsafeContent()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetUnsafeContentResults()
```

> **Note**
>
> Se hai già eseguito un video di esempio diverso da Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK), il codice da sostituire potrebbe essere diverso.

3. Eseguire il codice. Viene visualizzato un elenco di etichette dei contenuti inappropriati presenti nel video.

GetContentModeration risposta operativa

La risposta da `GetContentModeration` è una matrice, `ModerationLabels`, degli oggetti ContentModerationDetection. La matrice contiene un elemento ogni volta che viene rilevata un'etichetta di contenuto inappropriato. All'interno di un `ContentModerationDetectionObject` oggetto, ModerationLabelcontiene informazioni relative a un elemento rilevato con contenuto inappropriato o offensivo. `Timestamp`è l'ora, in millisecondi dall'inizio del video, in cui l'etichetta è stata rilevata. Le etichette sono organizzate gerarchicamente nello stesso modo delle etichette rilevate mediante analisi dell'immagine di contenuti inappropriati. Per ulteriori informazioni, consulta Moderazione dei contenuti.

Di seguito è riportato un esempio di risposta da `GetContentModeration`, ordinata per `NAME` e aggregata per `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"
    }
}
```

Di seguito è riportato un esempio di risposta da `GetContentModeration`, ordinata per `NAME` e aggregata per `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"
    }
}
```


Migliorare la precisione con la moderazione personalizzata

L'DetectModerationLabelsAPI di Amazon Rekognition ti consente di rilevare contenuti inappropriati, indesiderati o offensivi. La funzione Rekognition Custom Moderation consente di migliorare

la precisione utilizzando adattatori. DetectModerationLabels Gli adattatori sono componenti modulari che possono essere aggiunti a un modello di deep learning Rekognition esistente, estendendone le funzionalità per le attività su cui è addestrato. Creando un adattatore e fornendolo all'DetectModerationLabelsoperazione, è possibile ottenere una maggiore precisione per le attività di moderazione dei contenuti relative al caso d'uso specifico.

Quando personalizzi il modello di moderazione dei contenuti di Rekognition per etichette di moderazione specifiche, devi creare un progetto e addestrare un adattatore su un set di immagini che fornisci. È quindi possibile verificare in modo iterativo le prestazioni dell'adattatore e riaddestrarlo al livello di precisione desiderato. I progetti vengono utilizzati per contenere le diverse versioni degli adattatori.

Puoi utilizzare la console Rekognition per creare progetti e adattatori. In alternativa, puoi utilizzare un AWS SDK e i relativi componenti APIs per creare un progetto, addestrare un adattatore e gestire gli adattatori.

Creazione e utilizzo di adattatori

Gli adattatori sono componenti modulari che possono essere aggiunti a un modello di deep learning Rekognition esistente, estendendone le funzionalità per le attività su cui è addestrato. Addestrando un modello di deep learning con adattatori, puoi ottenere una maggiore precisione per le attività di analisi delle immagini relative al tuo caso d'uso specifico.

Per creare e utilizzare un adattatore, è necessario fornire dati di addestramento e test a Rekognition. Puoi farlo in uno di questi due modi:

- Analisi e verifica in blocco: puoi creare un set di dati di formazione analizzando in blocco le immagini che Rekognition analizzerà e alle quali assegnerà le etichette. È quindi possibile rivedere le annotazioni generate per le immagini e verificare o correggere le previsioni. Per ulteriori informazioni su come funziona l'analisi in blocco delle immagini, consulta Analisi in blocco.
- Annotazione manuale: con questo approccio puoi creare i tuoi dati di allenamento caricando e annotando le immagini. Puoi creare i tuoi dati di test caricando e annotando le immagini o suddividendoli automaticamente.

Scegli uno dei seguenti argomenti per avere ulteriori informazioni:

Argomenti

- Analisi e verifica di massa

- Annotazione manuale

Analisi e verifica di massa

Con questo approccio, carichi un gran numero di immagini che desideri utilizzare come dati di allenamento e poi usi Rekognition per ottenere previsioni per queste immagini, che assegna loro automaticamente delle etichette. Puoi utilizzare queste previsioni come punto di partenza per il tuo adattatore. È possibile verificare l'accuratezza delle previsioni e quindi addestrare l'adattatore in base alle previsioni verificate. Questo può essere fatto con la AWS console.

Analisi in blocco e moderazione personalizzata

Carica immagini per l'analisi di massa

Per creare un set di dati di addestramento per il tuo adattatore, carica immagini in blocco per Rekognition per cui prevedere le etichette. Per ottenere i migliori risultati, fornite quante più immagini possibile per l'addestramento, fino al limite di 10000, e assicuratevi che le immagini siano rappresentative di tutti gli aspetti del vostro caso d'uso.

Quando usi la AWS console puoi caricare immagini direttamente dal tuo computer o fornire un bucket Amazon Simple Storage Service per archiviare le tue immagini. Tuttavia, quando si utilizza APIs Rekognition con un SDK, è necessario fornire un file manifest che faccia riferimento alle immagini archiviate in un bucket Amazon Simple Storage Service. Per ulteriori informazioni, consulta Analisi in blocco.

Rivedi le previsioni

Dopo aver caricato le immagini sulla console Rekognition, Rekognition genererà delle etichette per esse. Puoi quindi verificare le previsioni in una delle seguenti categorie: vero positivo, falso positivo, vero negativo, falso negativo. Dopo aver verificato le previsioni, puoi addestrare un adattatore in base al tuo feedback.

Addestramento dell'adattatore

Una volta terminata la verifica delle previsioni restituite dall'analisi di massa, puoi avviare il processo di formazione dell'adattatore.

Ottieni il AdapterId

Una volta che l'adattatore è stato addestrato, puoi ottenere l'ID univoco da utilizzare con l'analisi delle immagini di Rekognition. APIs

Richiama l’operazione API

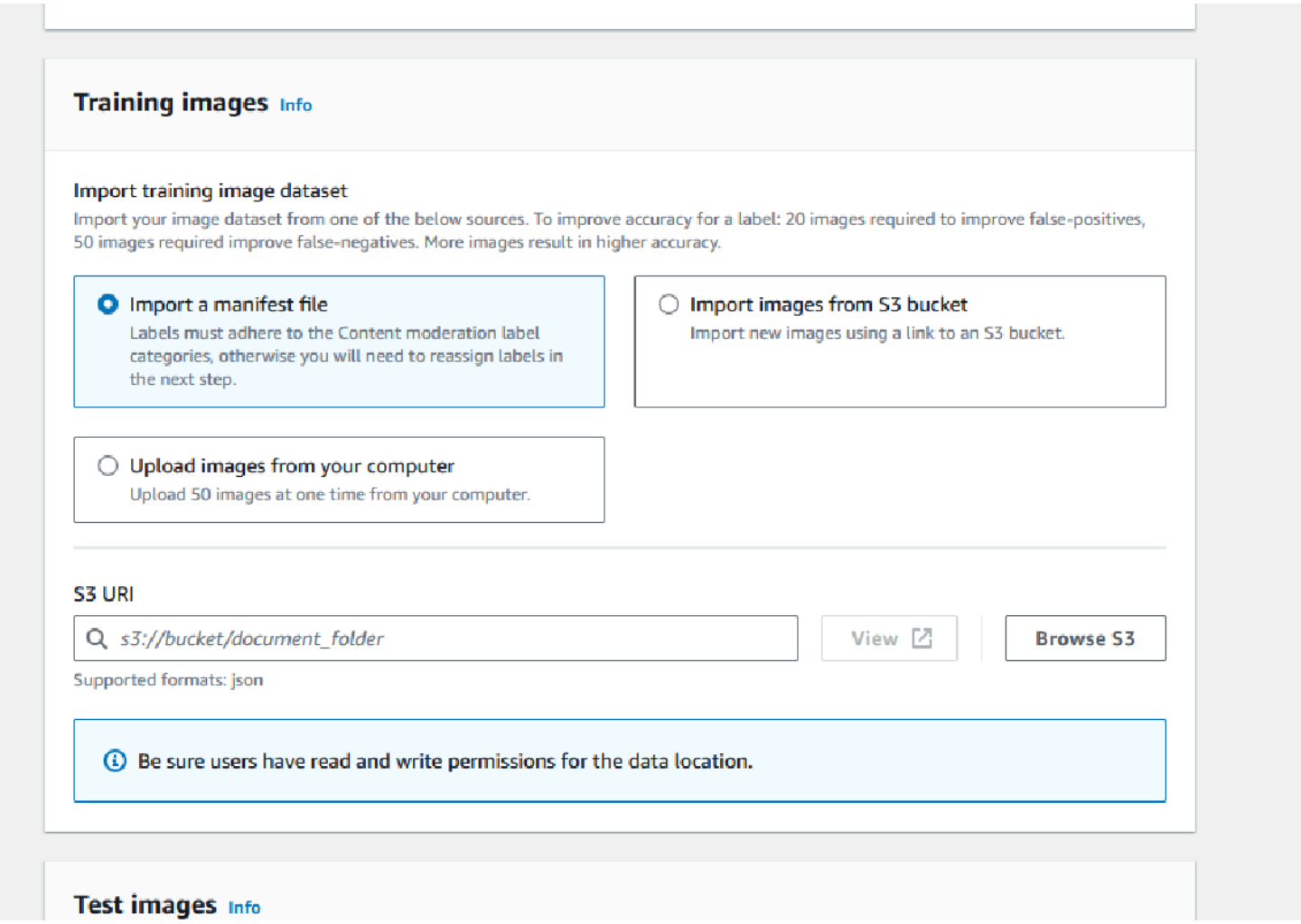
Per applicare il tuo adattatore personalizzato, fornisci il suo ID quando chiami uno dei centri di analisi delle immagini APIs che supporta gli adattatori. Ciò migliora l'accuratezza delle previsioni per le tue immagini.

Annotazione manuale

Con questo approccio puoi creare i tuoi dati di allenamento caricando e annotando le immagini manualmente. Puoi creare i tuoi dati di test caricando e annotando le immagini di test o suddividendoli automaticamente per fare in modo che Rekognition utilizzi automaticamente una parte dei tuoi dati di allenamento come immagini di test.

Caricamento e annotazione di immagini

Per addestrare l'adattatore, dovrai caricare una serie di immagini di esempio rappresentative del tuo caso d'uso. Per ottenere i migliori risultati, fornite quante più immagini possibile per l'addestramento, fino al limite di 10000, e assicuratevi che le immagini siano rappresentative di tutti gli aspetti del vostro caso d'uso.



Quando usi la AWS console puoi caricare immagini direttamente dal tuo computer, fornire un file manifest o fornire un bucket Amazon S3 per archiviare le tue immagini.

Tuttavia, quando si utilizza APIs Rekognition con un SDK, è necessario fornire un file manifest che faccia riferimento alle immagini archiviate in un bucket Amazon S3.

Puoi utilizzare l'interfaccia di annotazione della console Rekognition per annotare le tue immagini. Annota le tue immagini etichettandole con etichette, questo stabilisce una «verità di base» per l'addestramento. È inoltre necessario designare set di formazione e test o utilizzare la funzione di suddivisione automatica, prima di poter addestrare un adattatore. Al termine della designazione dei set di dati e dell'annotazione delle immagini, è possibile creare un adattatore basato sulle immagini annotate del set di test. È quindi possibile valutare le prestazioni dell'adattatore.

Crea un test di prova

Dovrai fornire un set di test annotato o utilizzare la funzione di divisione automatica. Il set di addestramento viene utilizzato per addestrare effettivamente l'adattatore. L'adattatore apprende gli schemi contenuti in queste immagini annotate. Il set di test viene utilizzato per valutare le prestazioni del modello prima di finalizzare l'adattatore.

Addestramento dell'adattatore

Dopo aver completato l'annotazione dei dati di addestramento o aver fornito un file manifesto, è possibile avviare il processo di addestramento dell'adattatore.

Ottieni l'ID adattatore

Una volta che l'adattatore è stato addestrato, puoi ottenere l'ID univoco da utilizzare con l'analisi delle immagini di Rekognition. APIs

Richiama l’operazione API

Per applicare il tuo adattatore personalizzato, fornisci il suo ID quando chiami uno dei centri di analisi delle immagini APIs che supporta gli adattatori. Ciò migliora l'accuratezza delle previsioni per le tue immagini.

Preparazione dei set di dati

La creazione di un adattatore richiede di fornire a Rekognition due set di dati, un set di dati di addestramento e un set di dati di test. Ogni set di dati è composto da due elementi: immagini e annotazioni/etichette. Le sezioni seguenti spiegano a cosa servono etichette e immagini e come si uniscono per creare set di dati.

Immagini

Dovrai addestrare un adattatore su esempi rappresentativi delle tue immagini. Quando selezionate le immagini per l'addestramento, provate a includere almeno alcune immagini che dimostrino la risposta prevista per ciascuna delle etichette che avete scelto come target con l'adattatore.

Per creare un set di dati di addestramento è necessario fornire uno dei due tipi di immagine seguenti:

- Immagini con previsioni false positive. Ad esempio, quando un modello base prevede che un'immagine contenga alcol, ma non è così.

- Immagini con previsioni false negative. Ad esempio, quando un modello base prevede che un'immagine non contenga alcol, ma in realtà ne contiene.

Per creare un set di dati bilanciato, si consiglia di fornire uno dei due tipi di immagine seguenti:

- Immagini con previsioni vere positive. Ad esempio, quando un modello base prevede correttamente che un'immagine contenga alcol. Si consiglia di fornire queste immagini se si forniscono immagini false positive.
- Immagini con previsioni vere negative. Ad esempio, quando un modello base prevede correttamente che un'immagine non contenga alcol. Si consiglia di fornire queste immagini se si forniscono immagini false negative.

Etichette

Un'etichetta si riferisce a uno qualsiasi dei seguenti elementi: oggetti, eventi, concetti o attività. Per moderazione dei contenuti, un'etichetta è un esempio di contenuto inappropriato, indesiderato o offensivo.

Nel contesto della creazione di un adattatore mediante l'addestramento del modello base di Rekognition, quando un'etichetta viene assegnata a un'immagine viene chiamata annotazione. Durante l'addestramento di un adattatore con la Rekognition Console, utilizzerai la Console per aggiungere annotazioni alle tue immagini scegliendo un'etichetta e quindi taggando le immagini che corrispondono all'etichetta. Attraverso questo processo, il modello impara a identificare gli elementi delle immagini in base all'etichetta assegnata. Questo processo di collegamento consente al modello di concentrarsi sui contenuti più pertinenti quando viene creato un adattatore, con conseguente maggiore precisione per l'analisi delle immagini.

In alternativa, è possibile fornire un file manifesto, che contiene informazioni sulle immagini e le relative annotazioni.

Allenare e testare set di dati

Il set di dati di addestramento è la base per la messa a punto del modello e la creazione di un adattatore personalizzato. È necessario fornire un set di dati di addestramento annotato da cui il modello possa imparare. Il modello impara da questo set di dati per migliorare le proprie prestazioni sul tipo di immagini fornite.

Per migliorare la precisione, è necessario creare il set di dati di allenamento annotando/etichettando le immagini. Puoi farlo in due modi:

- Assegnazione manuale delle etichette: puoi utilizzare la Rekognition Console per creare un set di dati di allenamento caricando le immagini che desideri contengano nel set di dati e quindi assegnando manualmente le etichette a queste immagini.
- File manifest: è possibile utilizzare un file manifest per addestrare l'adattatore. Il file manifest contiene informazioni sulle annotazioni di base relative alle immagini di addestramento e test, nonché sulla posizione delle immagini di allenamento. È possibile fornire il file manifest durante l'addestramento di un adattatore utilizzando APIs Rekognition o quando si utilizza la Console. AWS

Il set di dati di test viene utilizzato per valutare le prestazioni dell'adattatore dopo l'allenamento. Per garantire una valutazione affidabile, il set di dati di test viene creato utilizzando una parte del set di dati di addestramento originale che il modello non ha mai visto prima. Questo processo garantisce che le prestazioni dell'adattatore vengano valutate con nuovi dati, creando misurazioni e metriche accurate. Per un miglioramento ottimale della precisione, consultare Best practice per l'addestramento degli adattatori.

Gestione degli adattatori con la AWS CLI e SDKs

Rekognition ti consente di utilizzare molteplici funzionalità che sfruttano modelli di visione artificiale preaddestrati. Con questi modelli puoi svolgere attività come il rilevamento delle etichette e la moderazione dei contenuti. È inoltre possibile personalizzare questi determinati modelli utilizzando un adattatore.

Puoi utilizzare la creazione e la gestione dei progetti di Rekognition APIs (come and CreateProjectVersion) per creare CreateProjecte addestrare adattatori. Le pagine seguenti descrivono come utilizzare le operazioni API per creare, addestrare e gestire gli adattatori, utilizzando la AWS console, l' AWS SDK scelto o la AWS CLI.

Dopo aver addestrato un adattatore, è possibile utilizzarlo per eseguire l'inferenza con le funzionalità supportate. Attualmente, gli adattatori sono supportati quando si utilizza la funzione di moderazione dei contenuti.

Quando addestrate un adattatore utilizzando un AWS SDK, dovete fornire le vostre etichette di base (annotazioni sulle immagini) sotto forma di file manifesto. In alternativa, puoi utilizzare la Rekognition Console per creare e addestrare un adattatore.

> **Note**
>
> Gli adattatori non possono essere copiati. È possibile copiare solo le versioni del progetto Rekognition Custom Labels.

Argomenti

- Stati dell'adattatore
- Creare un progetto
- Descrizione di progetti
- Eliminazione di un progetto
- Creazione di una versione del progetto
- Descrivere una versione del progetto
- Eliminazione di una versione di progetto

Stati dell'adattatore

L'adattatore di moderazione personalizzato (versioni del progetto) può avere uno dei seguenti stati:

- TRAINING_IN_PROGRESS: l'adattatore è in corso di formazione sui file che hai fornito come documenti di formazione.
- TRAINING_COMPLETED - L'adattatore ha completato con successo l'addestramento ed è pronto per consentirvi di esaminarne le prestazioni.
- TRAINING_FAILED - L'adattatore non è riuscito a completare l'addestramento per qualche motivo. Consultate il file manifesto di output e il riepilogo del manifesto di output per informazioni sulla causa dell'errore.
- ELIMINAZIONE - L'adattatore è in fase di eliminazione.
- OBSOLETO: l'adattatore è stato addestrato su una versione precedente del modello base di Content Moderation. È in un periodo di prova e scadrà entro 60-90 giorni dal rilascio della nuova versione del modello base. Durante il periodo di prova, è comunque possibile utilizzare l'adattatore per l'inferenza con DetectModerationLabelsle nostre operazioni StartMediaAnalysisJobAPI. Fai riferimento alla Custom Moderation Console per la data di scadenza degli adattatori.
- SCADUTO: l'adattatore è stato addestrato su una versione precedente del modello base di moderazione dei contenuti e non può più essere utilizzato per ottenere risultati personalizzati con le operazioni o l'API. DetectModerationLabels StartMediaAnalysisJob Se in una richiesta di inferenza

viene specificato un adattatore scaduto, verrà ignorato e la risposta verrà invece restituita dalla versione più recente del modello base di moderazione personalizzata.

Creare un progetto

Con l'CreateProjectoperazione è possibile creare un progetto che conterrà un adattatore per le operazioni di rilevamento delle etichette di Rekognition. Un progetto è un gruppo di risorse e, nel caso di operazioni di rilevamento di etichette come DetectModerationLabels, un progetto consente di memorizzare adattatori che è possibile utilizzare per personalizzare il modello Rekognition di base. Quando si richiama CreateProject, si fornisce il nome del progetto che si desidera creare all'argomento. ProjectName

Per creare un progetto con la AWS console:

- Accedi alla Rekognition Console
- Fai clic su Moderazione personalizzata
- Scegliere Crea progetto
- Seleziona Crea un nuovo progetto o Aggiungi a un progetto esistente
- Aggiungi un nome di progetto
- Aggiungi un nome per l'adattatore
- Se lo desideri, aggiungi una descrizione
- Scegli come importare le immagini di allenamento: file Manifest, dal bucket S3 o dal tuo computer
- Scegli se vuoi dividere automaticamente i dati di allenamento o importare un file manifest
- Seleziona se desideri o meno che il progetto si aggiorni automaticamente
- Fare clic su Crea progetto

Per creare un progetto con AWS CLI e SDK:

1. Se non l'hai già fatto, installa e configura la AWS CLI e il. AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Usa il seguente codice per creare un progetto:

CLI

```
# Request
# Creating Content Moderation Project
aws rekognition create-project \
    --project-name "project-name" \
    --feature CONTENT_MODERATION \
    --auto-update ENABLED
    --profile profile-name
```

Descrizione di progetti

Puoi utilizzare l'DescribeProjectsAPI per ottenere informazioni sui tuoi progetti, incluse informazioni su tutti gli adattatori associati a un progetto.

Per descrivere i progetti con AWS CLI e SDK:

1. Se non l'hai già fatto, installa e configura la AWS CLI e il. AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Utilizza il seguente codice per descrivere un progetto:

CLI

```
# Request
# Getting CONTENT_MODERATION project details
aws rekognition describe-projects \
    --features CONTENT_MODERATION
    --profile profile-name
```

Eliminazione di un progetto

Puoi eliminare un progetto utilizzando la console Rekognition o chiamando l'API. DeleteProject Per eliminare un progetto, devi prima eliminare ogni adattatore associato. Un progetto o un modello eliminato non può essere ripristinato.

Per eliminare un progetto con la console: AWS

- Accedi alla Rekognition Console.

- Fai clic su Moderazione personalizzata.
- Prima di eliminare il progetto stesso, devi eliminare ogni adattatore associato al progetto. Eliminate tutti gli adattatori associati al progetto selezionando l'adattatore e quindi selezionando Elimina.
- Selezionate il progetto, quindi fate clic sul pulsante Elimina.

Per eliminare un progetto con AWS CLI e SDK:

1. Se non l'hai già fatto, installa e configura la AWS CLI e il. AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Utilizza il seguente codice per eliminare un progetto:

CLI

```
aws rekognition delete-project
  --project-arn project_arn \
  --profile profile-name
```

Creazione di una versione del progetto

È possibile addestrare un adattatore per la distribuzione utilizzando l'CreateProjectVersionoperazione. CreateProjectVersion crea prima una nuova versione di un adattatore associato a un progetto e poi inizia ad addestrare l'adattatore. La risposta CreateProjectVersion è un Amazon Resource Name (ARN) per la versione del modello. Il completamento dell'addestramento richiede tempo. Puoi conoscere lo stato attuale DescribeProjectVersions chiamando. Quando si addestra un modello, Rekognition utilizza addestramento e i set di dati di test associati al progetto. È possibile creare un set di dati utilizzando la console. Per ulteriori informazioni, consulta la sezione sui set di dati.

Per creare una versione di progetto con la console Rekognition:

- Accedi alla Rekognition Console AWS
- Fai clic su Moderazione personalizzata
- Scelta di un progetto.
- Nella pagina dei dettagli del progetto, scegli Crea adattatore

- Nella pagina «Crea un progetto», inserisci i dettagli richiesti per Dettagli del progetto, Immagini di formazione e Immagini di test, quindi seleziona Crea progetto.
- Nella pagina «Assegna etichette alle immagini», aggiungi delle etichette alle immagini e, al termine, seleziona Inizia la formazione

Per creare una versione del progetto con AWS CLI e SDK:

1. Se non l'hai già fatto, installa e configura la AWS CLI e il. AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Usa il codice seguente per creare una versione del progetto:

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
```

Descrivere una versione del progetto

È possibile elencare e descrivere gli adattatori associati a un progetto utilizzando l'DescribeProjectVersionsoperazione. È possibile specificare fino a 10 versioni del modello in ProjectVersionArns. Se non specifichi un valore, vengono restituite descrizioni per tutte le versioni del modello nel progetto.

Per descrivere una versione del progetto con AWS CLI e SDK:

1. Se non l'hai già fatto, installa e configura la AWS CLI e il. AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Usa il codice seguente per descrivere una versione del progetto:

CLI

```
aws rekognition describe-project-versions
  --project-arn project_arn \
  --version-names [versions]
```

Eliminazione di una versione di progetto

È possibile eliminare un adattatore Rekognition associato a un progetto utilizzando l'operazione. DeleteProjectVersion Non puoi eliminare un adattatore se è in esecuzione o in fase di addestramento. Per verificare lo stato di un adattatore, richiamate l' DescribeProjectVersions operazione e controllate il campo Status da essa restituito. Per interrompere una chiamata all'adattatore in esecuzione StopProjectVersion. Se un modello è in fase di addestramento, attendi che finisca prima di eliminarlo. Prima di eliminare il progetto stesso, devi eliminare ogni adattatore associato al progetto.

Per eliminare una versione del progetto con la console Rekognition:

- Accedi alla Rekognition Console
- Fai clic su Moderazione personalizzata
- Dalla scheda Progetti puoi vedere tutti i tuoi progetti e gli adattatori associati. Seleziona un adattatore, quindi seleziona Elimina.

Per eliminare una versione del progetto con AWS CLI e SDK:

1. Se non l'hai già fatto, installa e configura la AWS CLI e il. AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.
2. Usa il codice seguente per eliminare una versione del progetto:

CLI

```
# Request
aws rekognition delete-project-version
  --project-version-arn model_arn \
```

```
    --profile profile-name
```

Tutorial sull'adattatore di moderazione personalizzato

Questo tutorial mostra come creare, addestrare, valutare, utilizzare e gestire gli adattatori utilizzando la Rekognition Console. Per creare, utilizzare e gestire gli adattatori con l' AWS SDK, consulta. Gestione degli adattatori con la AWS CLI e SDKs

Gli adattatori consentono di migliorare la precisione delle operazioni API di Rekognition, personalizzando il comportamento del modello in base alle proprie esigenze e ai propri casi d'uso. Dopo aver creato un adattatore con questo tutorial, sarete in grado di usarlo per analizzare le vostre immagini con operazioni come DetectModerationLabels, oltre a riaddestrare l'adattatore per ulteriori miglioramenti futuri.

In questo tutorial, apprenderai come:

- Creare un progetto utilizzando Rekognition Console
- Annotare dei dati di addestramento
- Addestrare il tuo adattatore sulla base del tuo set di dati di allenamento
- Verificare le prestazioni del tuo adattatore
- Usare l'adattatore per l'analisi delle immagini

Prerequisiti

Prima di completare questo tutorial si consiglia di leggere Creazione e utilizzo di adattatori.

Per creare un adattatore, puoi utilizzare lo strumento Rekognition Console per creare un progetto, caricare e annotare le tue immagini e quindi addestrare un adattatore su queste immagini. Per iniziare, consulta Creazione di un progetto e formazione di un adattatore.

In alternativa, puoi utilizzare la console o l'API di Rekognition per recuperare le previsioni per le immagini e poi verificarle prima di addestrare un adattatore su queste previsioni. Per iniziare, consulta Analisi in blocco, verifica delle previsioni e formazione di un adattatore.

Annotazione dell'immagine

Puoi annotare tu stesso le immagini etichettandole con la console Rekognition oppure utilizzare l'analisi Rekognition Bulk per annotare le immagini che puoi poi verificare che siano state etichettate correttamente. Scegli uno degli argomenti seguenti per iniziare.

Argomenti

- Creazione di un progetto e formazione di un adattatore
- Analisi in blocco, verifica delle previsioni e formazione di un adattatore

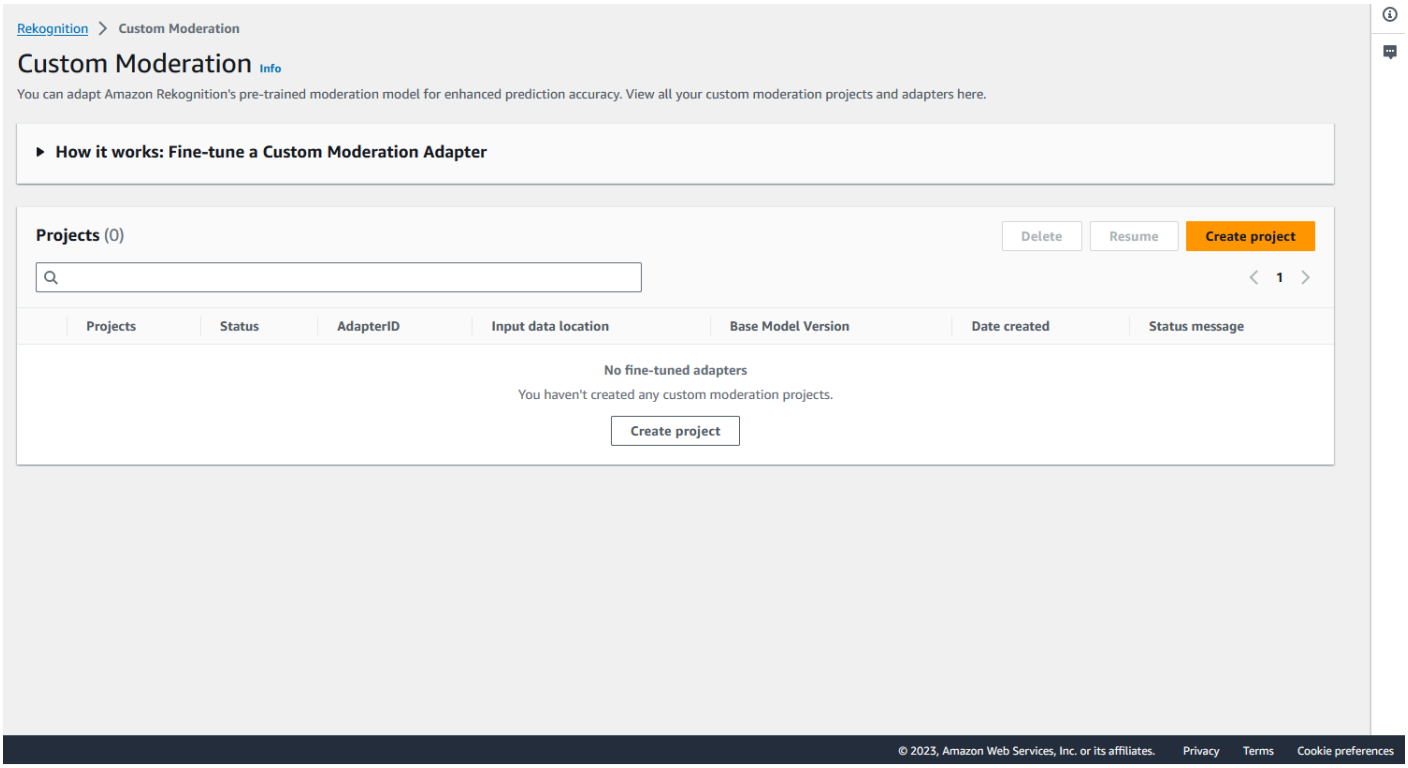
Creazione di un progetto e formazione di un adattatore

Completa le operazioni seguenti per addestrare l'adattatore annotando le immagini utilizzando la console Rekognition.

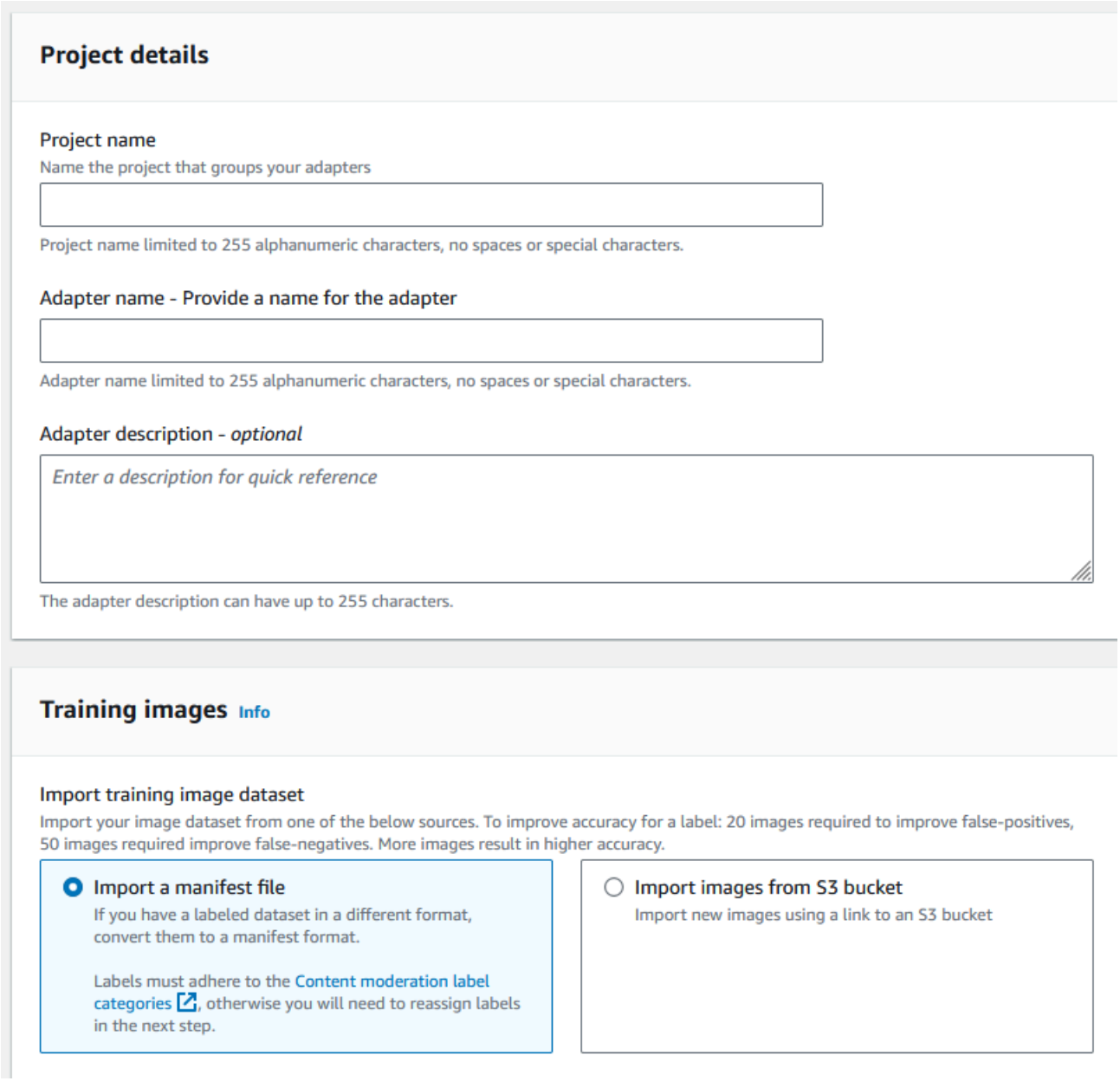
Creazione di un progetto

Prima di poter addestrare o utilizzare un adattatore, devi creare il progetto che lo conterrà. È inoltre necessario fornire le immagini utilizzate per addestrare l'adattatore. Per creare un progetto, un adattatore e i tuoi set di dati di immagini:

1. Accedi alla console di AWS gestione e apri la console Rekognition all'indirizzo. https://console.aws.amazon.com/rekognition/
2. Nel riquadro a sinistra, scegliere Moderazione personalizzata. Viene visualizzata la pagina iniziale di Rekognition Custom Moderation.



3. La pagina di destinazione Custom Moderation mostra un elenco di tutti i tuoi progetti e adattatori, e c'è anche un pulsante per creare un adattatore. Scegli Crea progetto per creare un nuovo progetto e un adattatore.
4. Se è la prima volta che crei un adattatore, ti verrà richiesto di creare un bucket Amazon S3 per archiviare i file relativi al tuo progetto e al tuo adattatore. Scegli Crea bucket Amazon S3.
5. Nella pagina seguente, inserisci il nome dell'adattatore e il nome del progetto. Se lo desideri, fornisci una descrizione dell'adattatore.



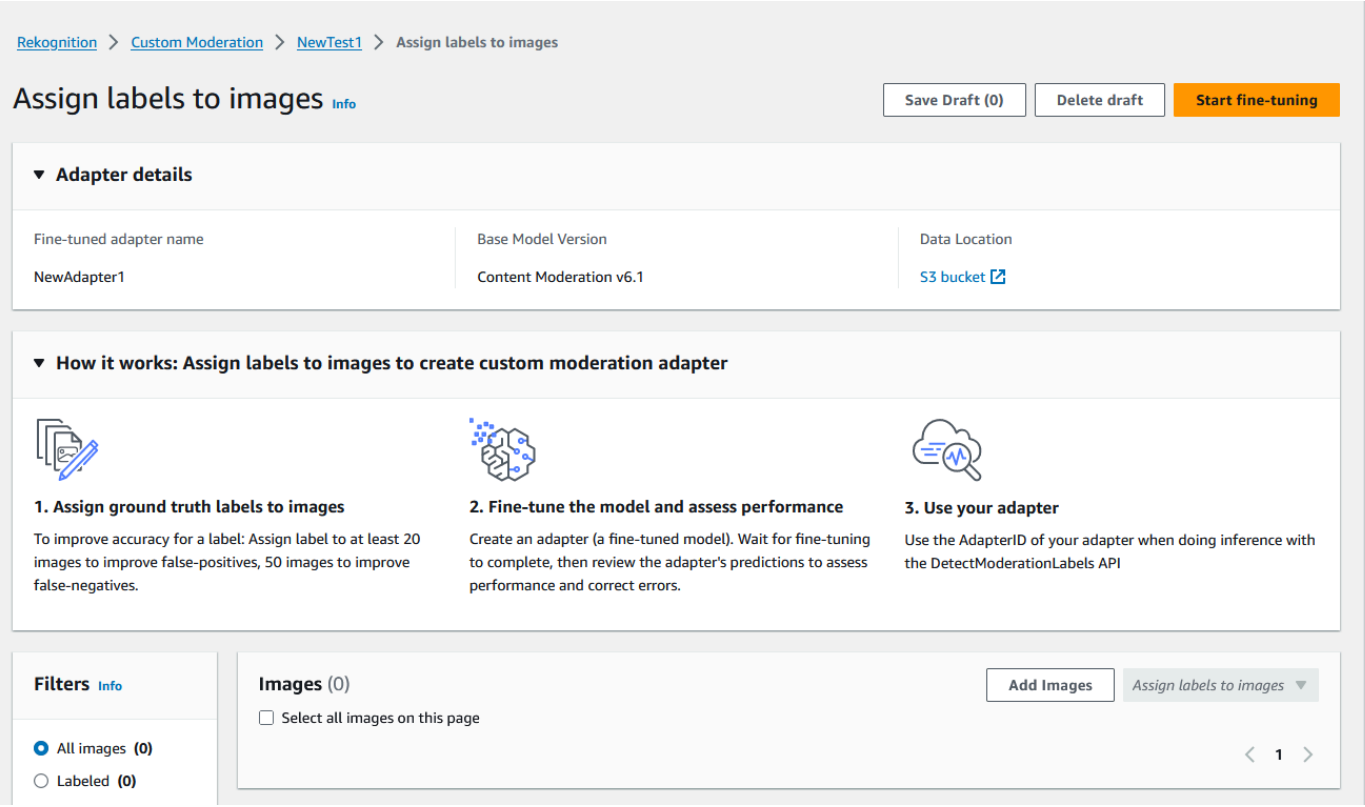
6. In questo passaggio, fornirai anche le immagini per l'adattatore. Puoi selezionare: Importa immagini dal tuo computer, Importa file manifest o Importa immagini dal bucket Amazon S3. Se scegli di importare le tue immagini da un bucket Amazon S3, fornisci il percorso del bucket e della cartella che contiene le immagini di allenamento. Se carichi le immagini direttamente dal computer, tieni presente che puoi caricare solo fino a 30 immagini alla volta. Se utilizzi un file manifesto che contiene annotazioni, puoi saltare i passaggi elencati di seguito relativi all'annotazione delle immagini e passare alla sezione relativa a Visualizzazione delle prestazioni dell'adattatore.
7. Nella sezione Dettagli del set di dati di test, scegli Autosplit per fare in modo che Rekognition selezioni automaticamente la percentuale appropriata delle immagini come dati di test, oppure puoi scegliere Importa manualmente il file manifest.

8. Dopo aver inserito queste informazioni, seleziona Crea progetto.

Addestramento dell'adattatore

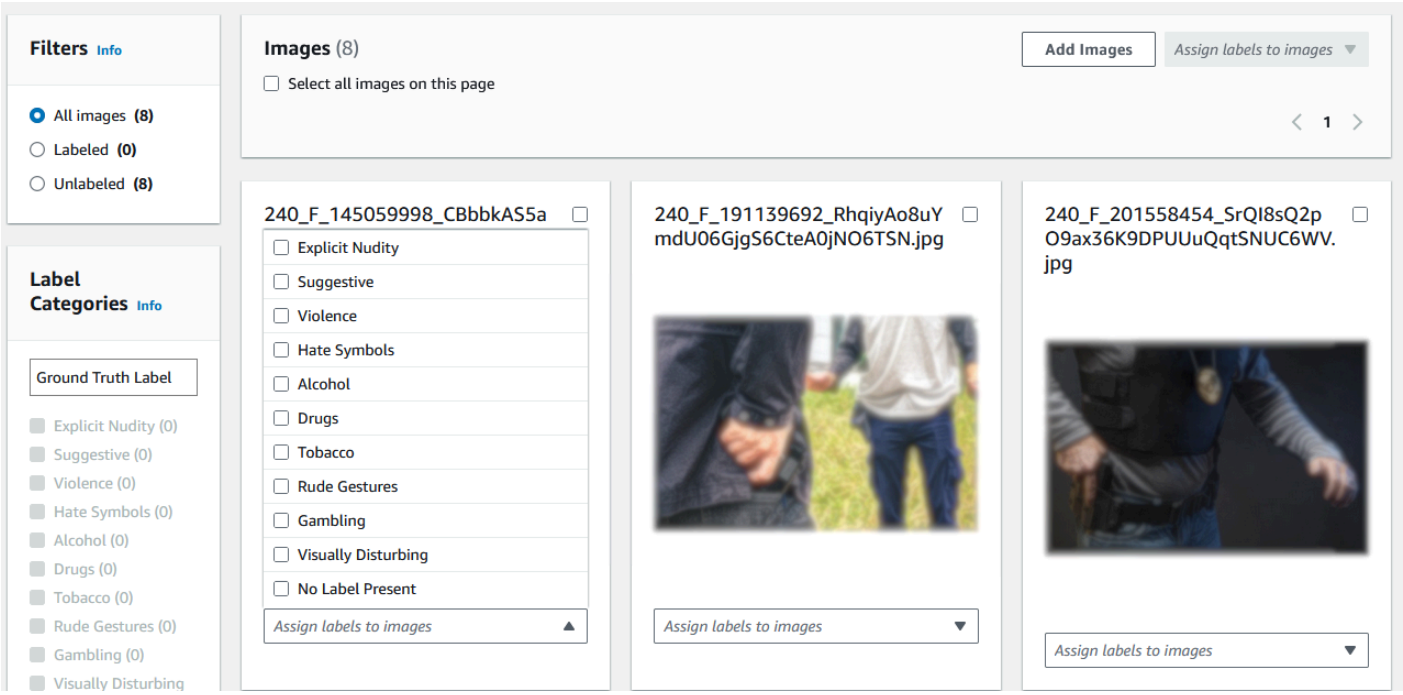
Per addestrare un adattatore sulle tue immagini non annotate:

1. Seleziona il progetto che contiene l'adattatore, quindi scegli l'opzione Assegna etichetta alle immagini.
2. Nella pagina Assegna etichetta alle immagini, puoi vedere tutte le immagini che sono state caricate come immagini di formazione. Puoi filtrare queste immagini in base allo stato etichettato/senza etichetta e per categoria di etichetta utilizzando i due pannelli di selezione degli attributi sulla sinistra. È possibile aggiungere altre immagini al set di dati di allenamento selezionando il pulsante Aggiungi immagini.



3. Dopo aver aggiunto immagini al set di dati di allenamento, è necessario annotare le immagini con delle etichette. Dopo aver caricato le immagini, la pagina «Assegna etichette alle immagini» si aggiornerà per mostrare le immagini che hai caricato. Ti verrà richiesto di selezionare l'etichetta appropriata per le tue immagini da un elenco a discesa di etichette supportate da Rekognition Moderation. È possibile selezionare più di un'etichetta.

4. Continua questo processo finché non avrai aggiunto etichette a ciascuna delle immagini nei tuoi dati di allenamento.
5. Dopo aver etichettato tutti i dati, seleziona Inizia addestramento per iniziare ad addestrare il modello, che crea l'adattatore.



6. Prima di iniziare il processo di formazione, puoi aggiungere qualsiasi tag all'adattatore che desideri. È inoltre possibile fornire all'adattatore una chiave di crittografia personalizzata o utilizzare una chiave KMS. AWS Una volta che hai finito di aggiungere i tag che desideri e di personalizzare la crittografia a tuo piacimento, seleziona Addestra adattatore per avviare il processo di formazione del tuo adattatore.
7. Attendi che l'adattatore finisca l'addestramento. Una volta completato l'addestramento, riceverai una notifica che indica che l'adattatore è terminato.

Una volta che lo stato dell'adattatore è «Addestramento completato», puoi rivedere le metriche dell'adattatore

Analisi in blocco, verifica delle previsioni e formazione di un adattatore

Completa i seguenti passaggi per addestrare il tuo adattatore verificando le previsioni delle analisi di massa basate sul modello di moderazione dei contenuti di Rekognition.

Per addestrare un adattatore verificando le previsioni del modello di moderazione dei contenuti di Rekognition, devi:

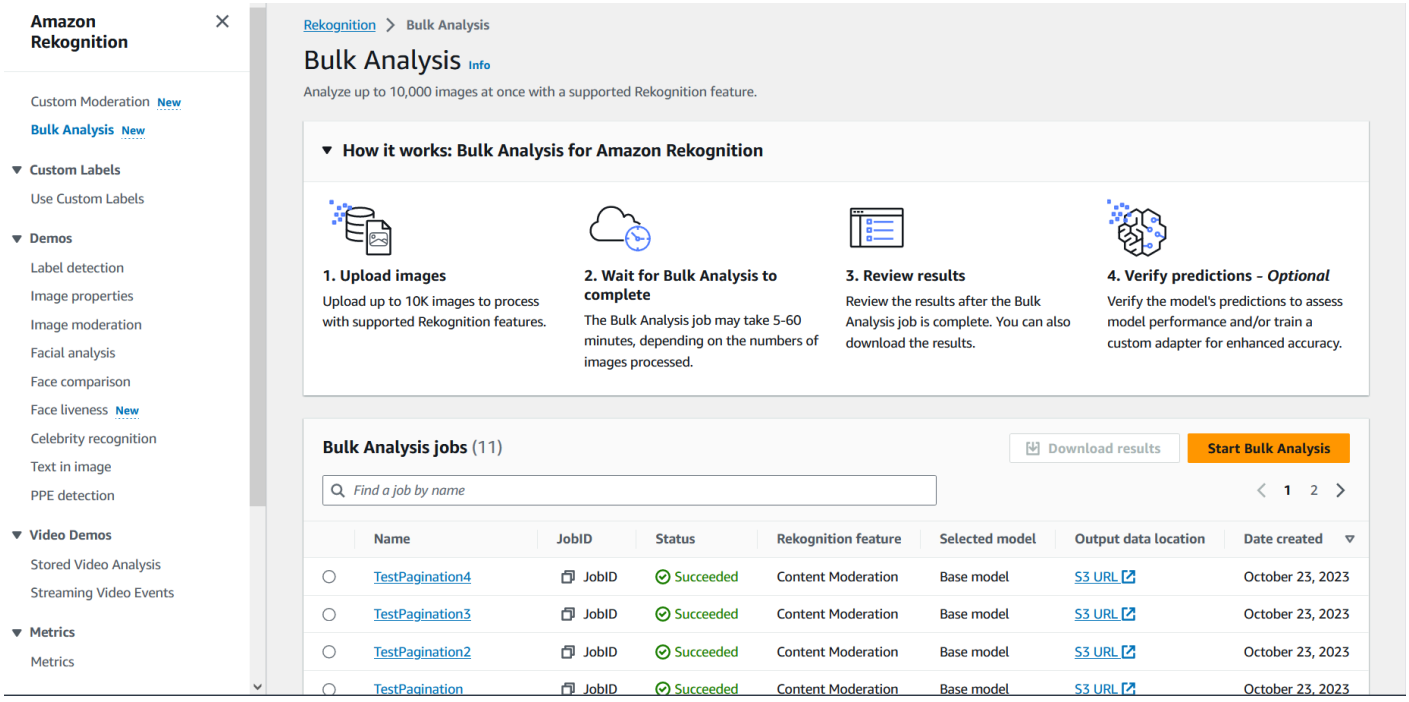
1. Eseguire un'analisi collettiva delle tue immagini
2. Verificare le previsioni restituite per le tue immagini

Puoi ottenere previsioni per le immagini eseguendo analisi in blocco con il modello base di Rekognition o con un adattatore che hai già creato.

Eseguire un'analisi collettiva delle tue immagini

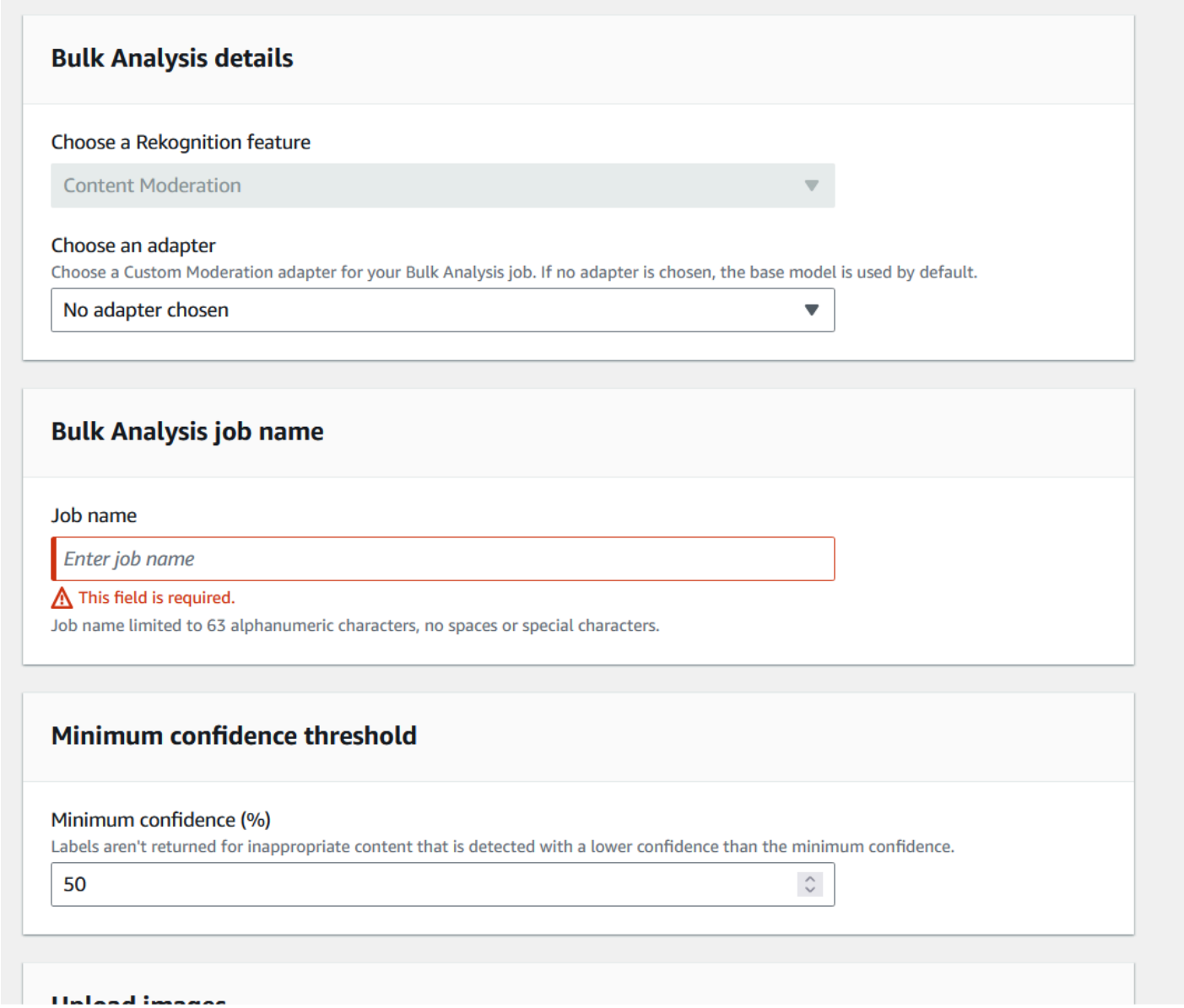
Per addestrare un adattatore sulle previsioni che hai verificato, devi prima avviare un processo di analisi in blocco per analizzare un batch di immagini utilizzando il modello base di Rekognition o un adattatore di tua scelta. Per eseguire un processo di analisi in blocco:

1. Accedi AWS Management Console e apri la console Amazon Rekognition all'indirizzo. https://console.aws.amazon.com/rekognition/
2. Nel riquadro a sinistra, scegliere Analisi in blocco. Viene visualizzata la pagina iniziale dedicata all'analisi di massa. Scegli Avvia analisi in blocco. La panoramica della funzionalità Bulk Analysis mostra i passaggi per caricare le immagini, attendere l'analisi, rivedere i risultati e, facoltativamente, verificare le previsioni dei modelli. Elenca i lavori recenti di Bulk Analysis per la moderazione dei contenuti utilizzando il modello base.



3. Se è la prima volta che crei un adattatore, ti verrà richiesto di creare un bucket Amazon Simple Storage Service per archiviare i file relativi al tuo progetto e al tuo adattatore. Scegli Crea bucket Amazon S3.

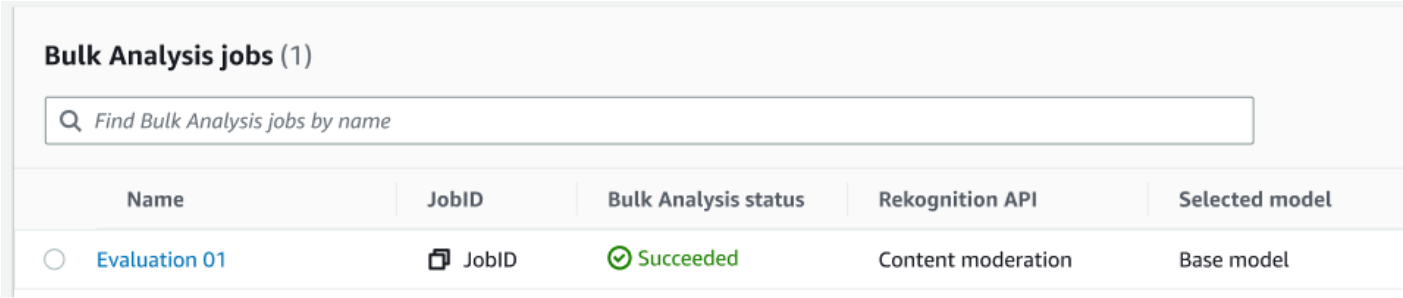
4. Seleziona l'adattatore che desideri utilizzare per l'analisi di massa utilizzando il menu a tendina Scegli un adattatore. Se non viene scelto alcun adattatore, per impostazione predefinita verrà utilizzato il modello base. Ai fini del presente tutorial non selezionare un adattatore.



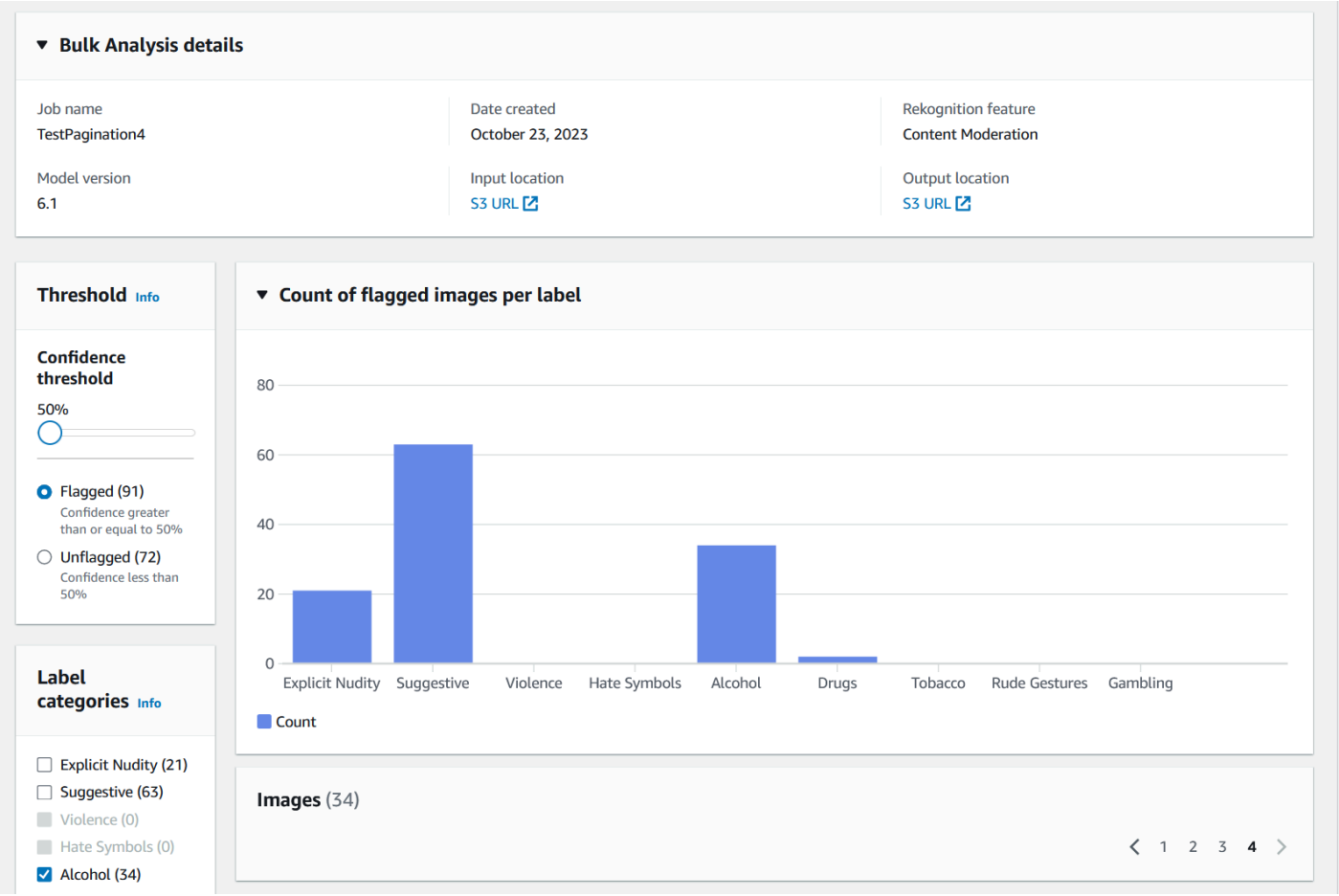
5. Nel campo Nome del processo di analisi in blocco, inserisci il nome del processo di analisi in blocco.
6. Scegliete un valore per la soglia di confidenza minima. Le previsioni delle etichette con un valore inferiore alla soglia di confidenza prescelta non verranno restituite. Tieni presente che quando valuterai le prestazioni del modello in un secondo momento, non sarai in grado di regolare la soglia di confidenza al di sotto della soglia di confidenza minima scelta.
7. In questo passaggio, fornirai anche le immagini che desideri analizzare con l'analisi in blocco. Queste immagini possono essere utilizzate anche per addestrare l'adattatore. Puoi scegliere

Carica immagini dal tuo computer o Importa immagini dal bucket Amazon S3. Se scegli di importare i tuoi documenti da un bucket Amazon S3, fornisci il percorso del bucket e della cartella che contiene le immagini di allenamento. Se carichi i documenti direttamente dal computer, tieni presente che puoi caricare solo fino a 50 immagini alla volta.

8. Dopo aver inserito queste informazioni, scegli Avvia analisi. Questo avvierà il processo di analisi utilizzando il modello base di Rekognition.
9. Puoi controllare lo stato del tuo processo di analisi in blocco controllando lo stato dell'analisi di massa del lavoro nella pagina principale di analisi in blocco. Quando lo stato di analisi in blocco diventa «Riuscito», i risultati dell'analisi sono pronti per essere esaminati.



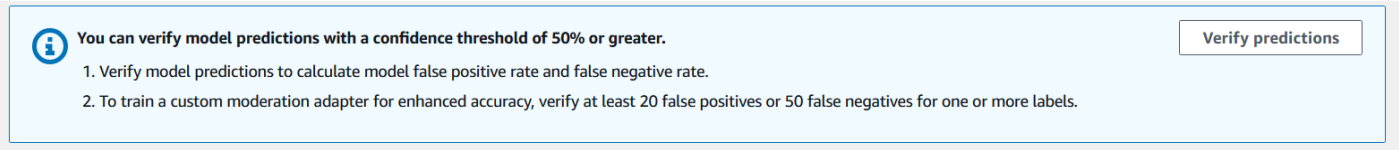
10. Scegli l'analisi che hai creato dall'elenco dei lavori di Analisi in blocco.
11. Nella pagina dei dettagli di analisi in blocco puoi vedere le previsioni che il modello base di Rekognition ha fatto per le immagini che hai caricato.
12. Esamina le prestazioni del modello base. È possibile modificare la soglia di confidenza che l'adattatore deve avere per assegnare un'etichetta a un'immagine utilizzando il cursore della soglia di confidenza. Il numero di istanze contrassegnate e non contrassegnate cambierà man mano che regolate la soglia di confidenza. Il pannello Categorie di etichette mostra le categorie di primo livello riconosciute da Rekognition ed è possibile selezionare una categoria in questo elenco per visualizzare tutte le immagini a cui è stata assegnata quell'etichetta.



Verifica le previsioni

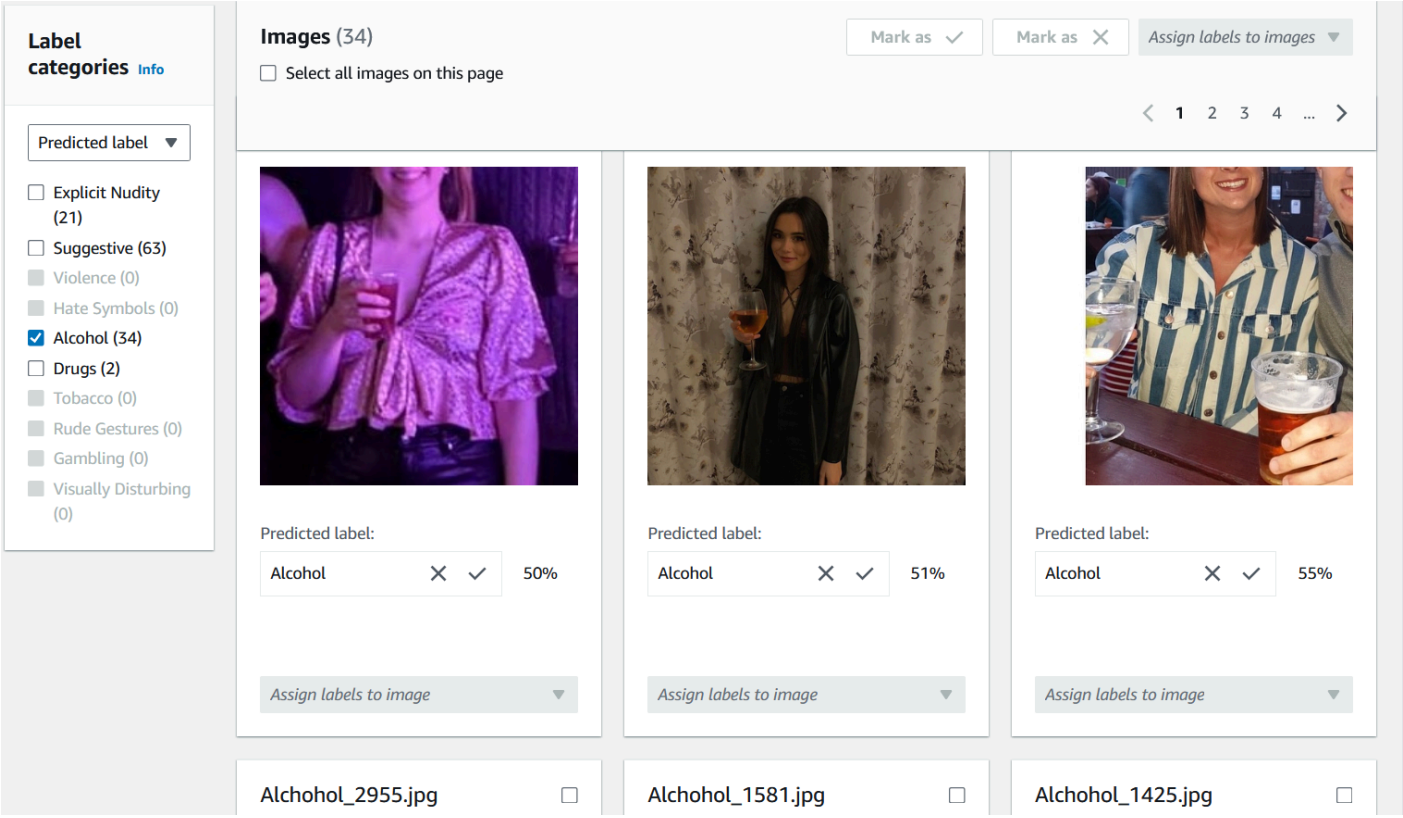
Se hai verificato l'accuratezza del modello base di Rekognition o di un adattatore scelto e desideri migliorarla, puoi utilizzare il flusso di lavoro di verifica:

1. Dopo aver esaminato le prestazioni del modello base, dovrai verificare le previsioni. La correzione delle previsioni consentirà di addestrare un adattatore. Scegli Verifica le previsioni nella parte superiore della pagina di analisi in blocco.

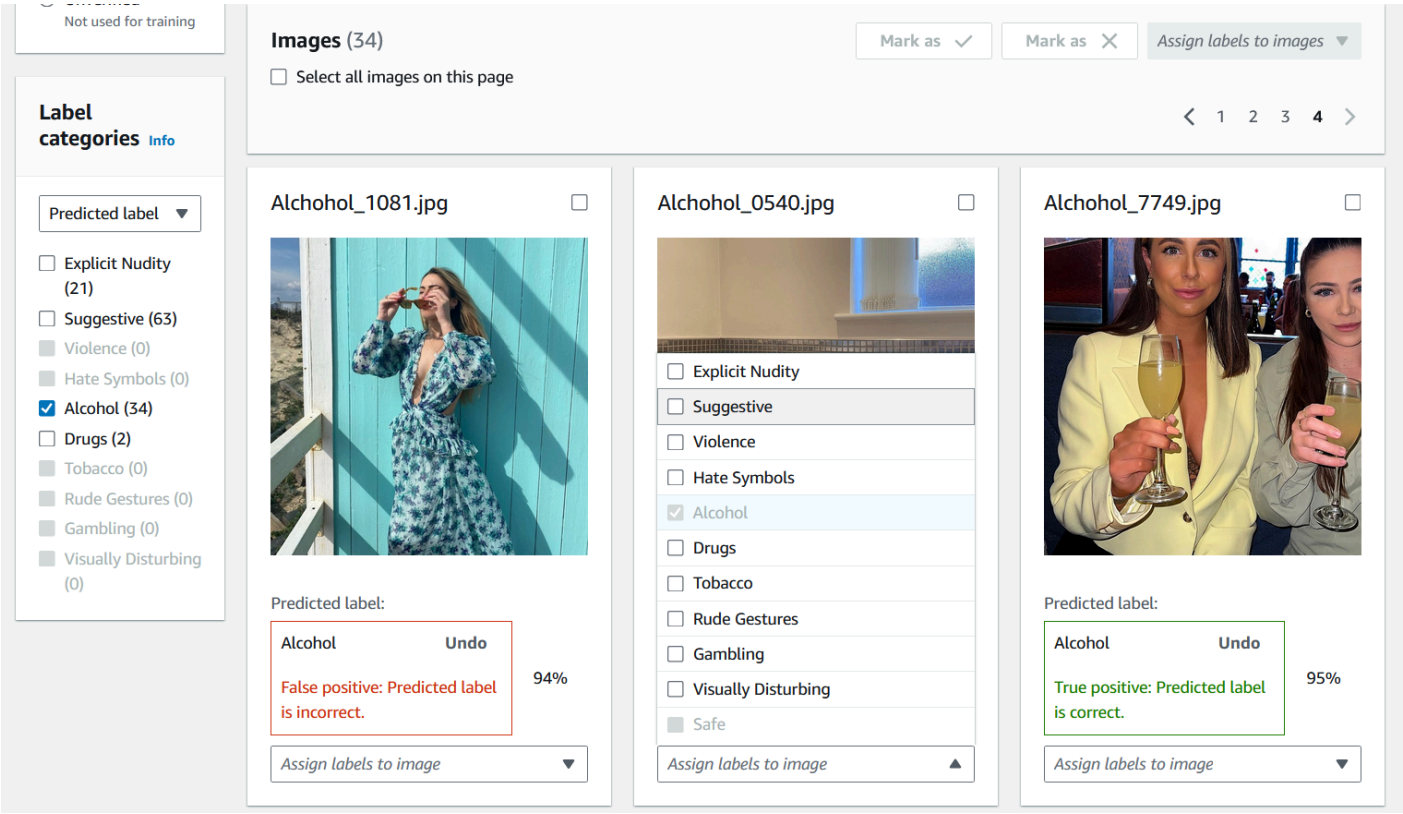


2. Nella pagina Verifica le previsioni, puoi vedere tutte le immagini che hai fornito al modello base di Rekognition o a un adattatore scelto, insieme all'etichetta prevista per ogni immagine. È necessario verificare che ogni previsione sia corretta o errata utilizzando i pulsanti sotto l'immagine. Usa il pulsante «X» per contrassegnare una previsione come errata e il pulsante

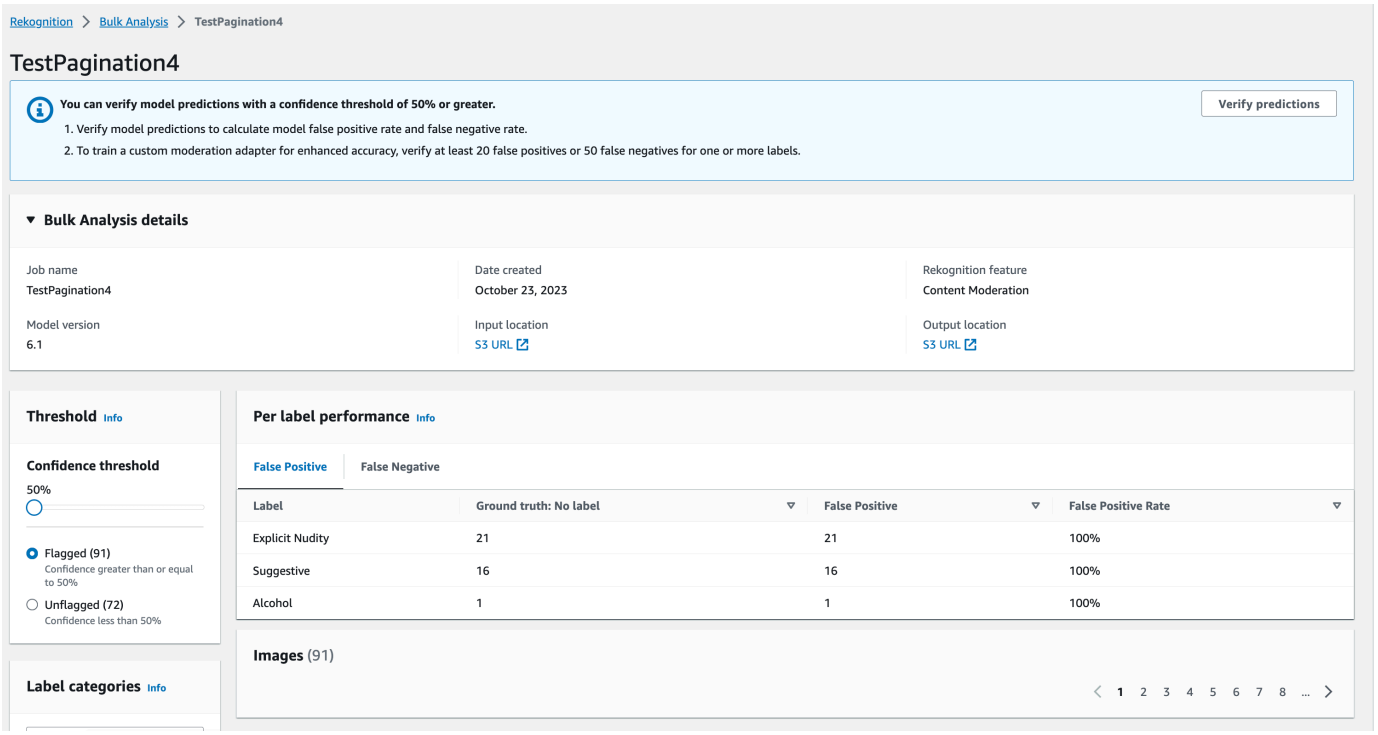
con il segno di spunta per contrassegnare una previsione come corretta. Per addestrare un adattatore dovrai verificare almeno 20 previsioni false positive e 50 previsioni false negative per una determinata etichetta. Maggiore è il numero di previsioni verificate, migliori saranno le prestazioni dell'adattatore.



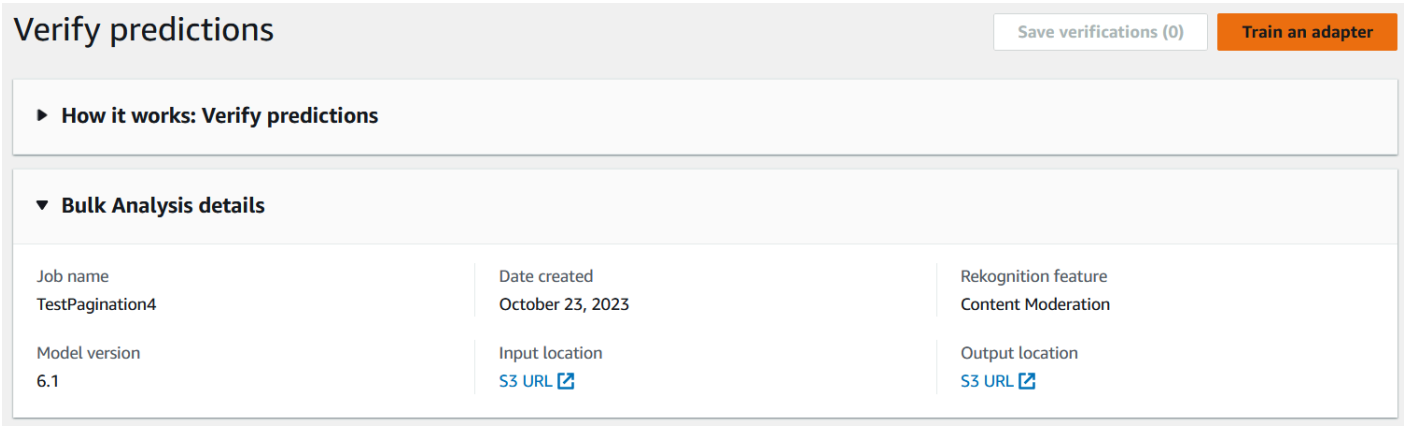
Dopo aver verificato una previsione, il testo sotto l'immagine cambierà per mostrarti il tipo di previsione che hai verificato. Dopo aver verificato un'immagine, puoi anche aggiungere altre etichette all'immagine utilizzando il menu Assegna etichette all'immagine. Puoi vedere quali immagini sono contrassegnate o meno dal modello in base alla soglia di confidenza prescelta o filtrare le immagini per categoria.



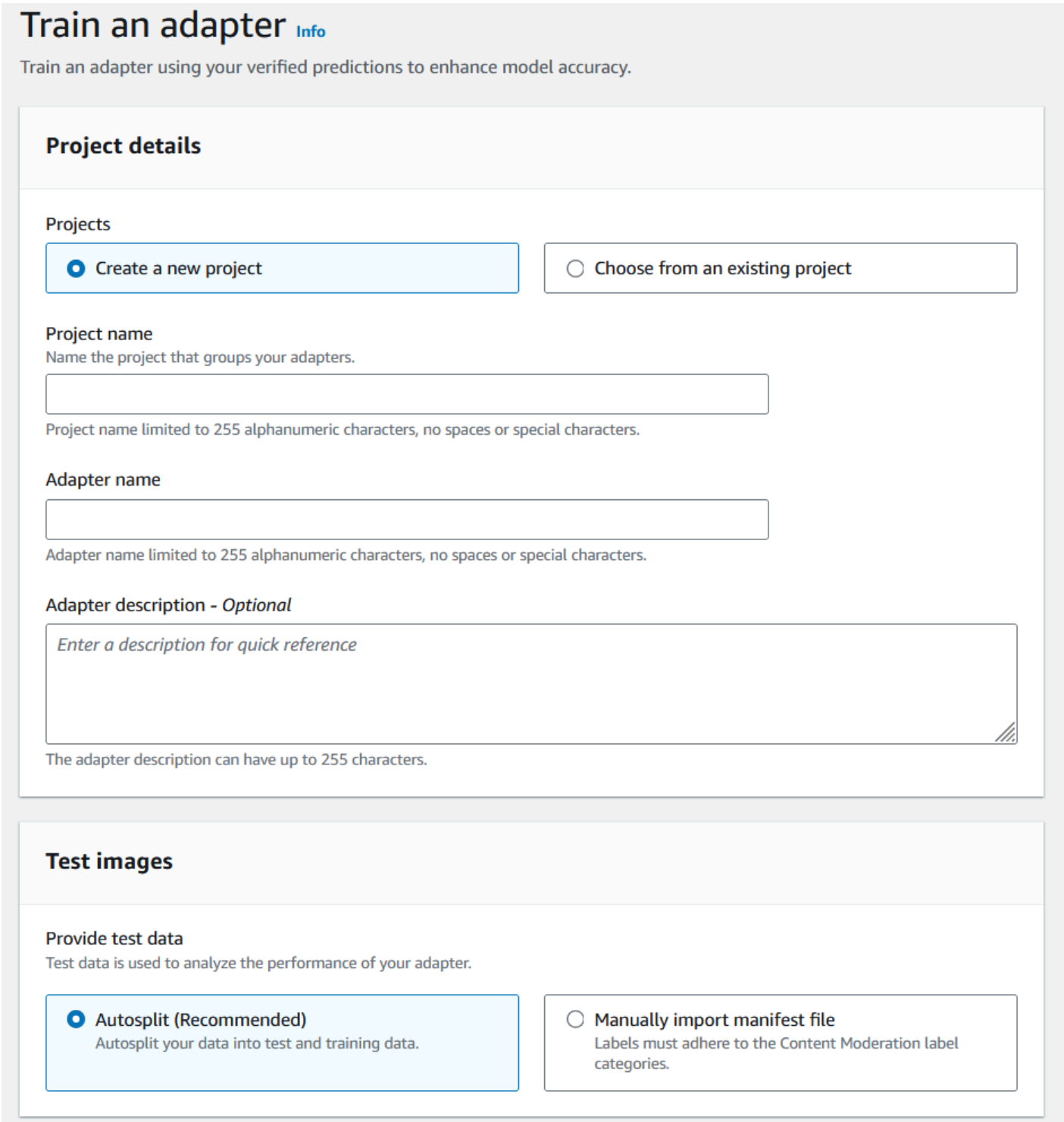
3. Una volta terminata la verifica di tutte le previsioni che desideri verificare, puoi visualizzare le statistiche relative alle previsioni verificate nella sezione Prestazioni per etichetta della pagina Verifica. Puoi anche tornare alla pagina dei dettagli dell'analisi in blocco per visualizzare queste statistiche.



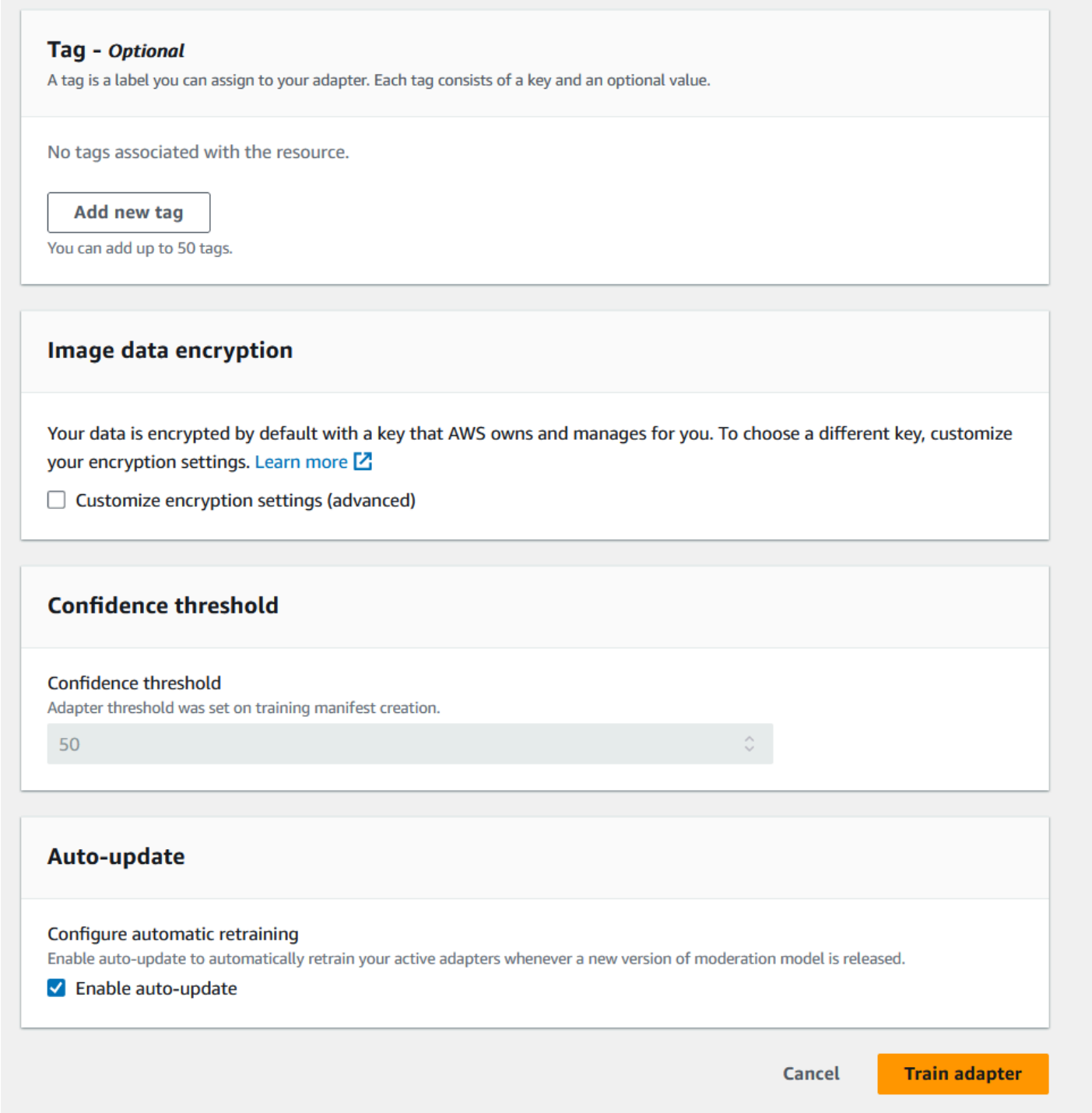
4. Quando sei soddisfatto delle statistiche relative alle prestazioni per etichetta, vai nuovamente alla pagina Verifica le previsioni, quindi seleziona il pulsante Addestra un adattatore per iniziare ad addestrare l'adattatore.



5. Nella pagina Addestra un adattatore ti verrà richiesto di creare un progetto o scegliere un progetto esistente. Assegna un nome al progetto e all'adattatore che saranno contenuti nel progetto. È inoltre necessario specificare la fonte delle immagini di test. Quando si specificano le immagini, è possibile scegliere Autosplit per fare in modo che Rekognition utilizzi automaticamente una parte dei dati di allenamento come immagini di test, oppure è possibile specificare manualmente un file manifest. Si consiglia di scegliere Autosplit.



6. Specificate i tag che desiderate, oltre a una AWS KMS chiave se non desiderate utilizzare la chiave predefinita AWS . Si consiglia di lasciare abilitato l'aggiornamento automatico.
7. Seleziona Addestra adattatore.

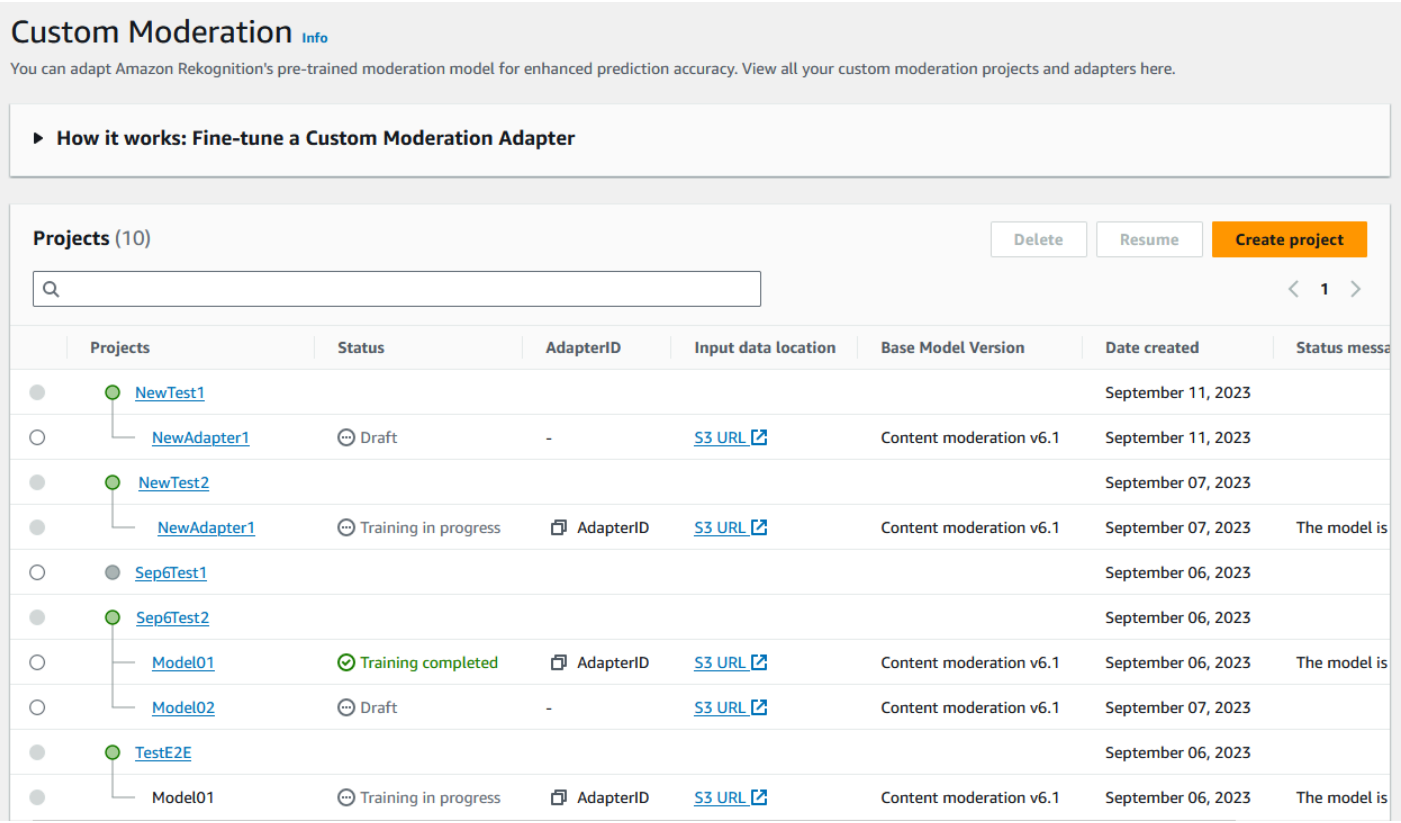


8. Una volta che lo stato del tuo adattatore nella pagina iniziale di Custom Moderation è diventato «Addestramento completata», puoi verificarne le prestazioni. Per ulteriori informazioni, consulta Visualizzazione delle prestazioni dell'adattatore.

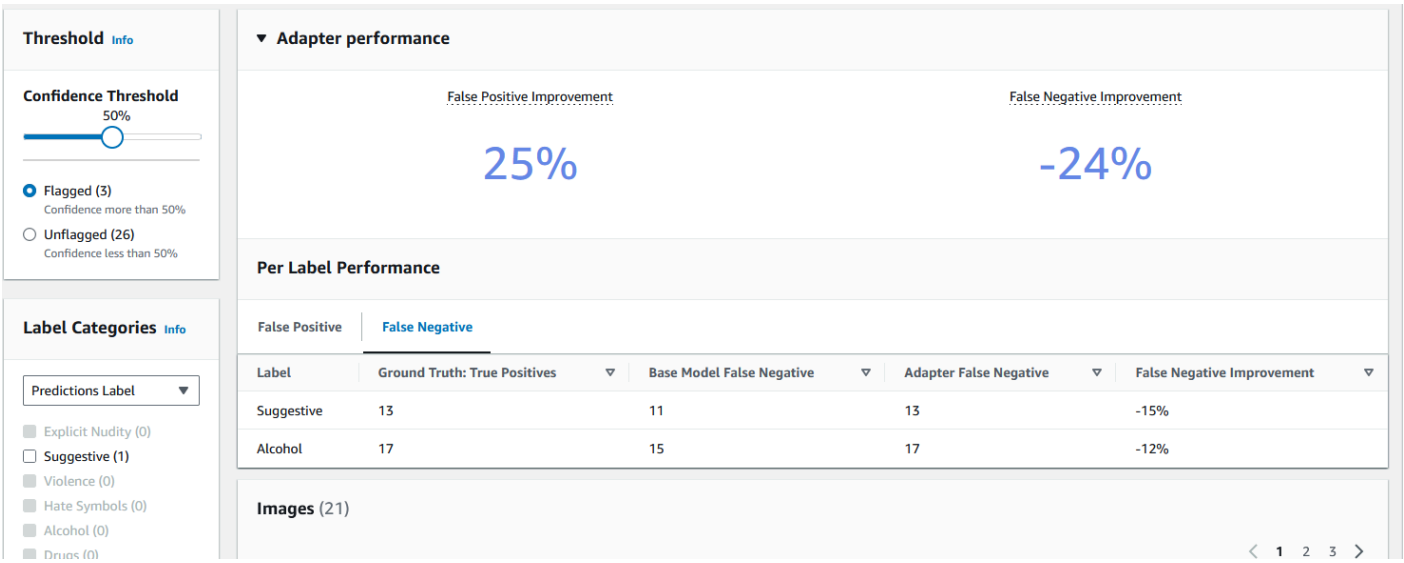
Visualizzazione delle prestazioni dell'adattatore

Per verificare le prestazioni del tuo adattatore:

1. Quando usi la console, potrai vedere lo stato di tutti gli adattatori associati a un progetto nella scheda Progetti della pagina iniziale sulla moderazione personalizzata. Vai alla pagina iniziale dedicata alla moderazione personalizzata.



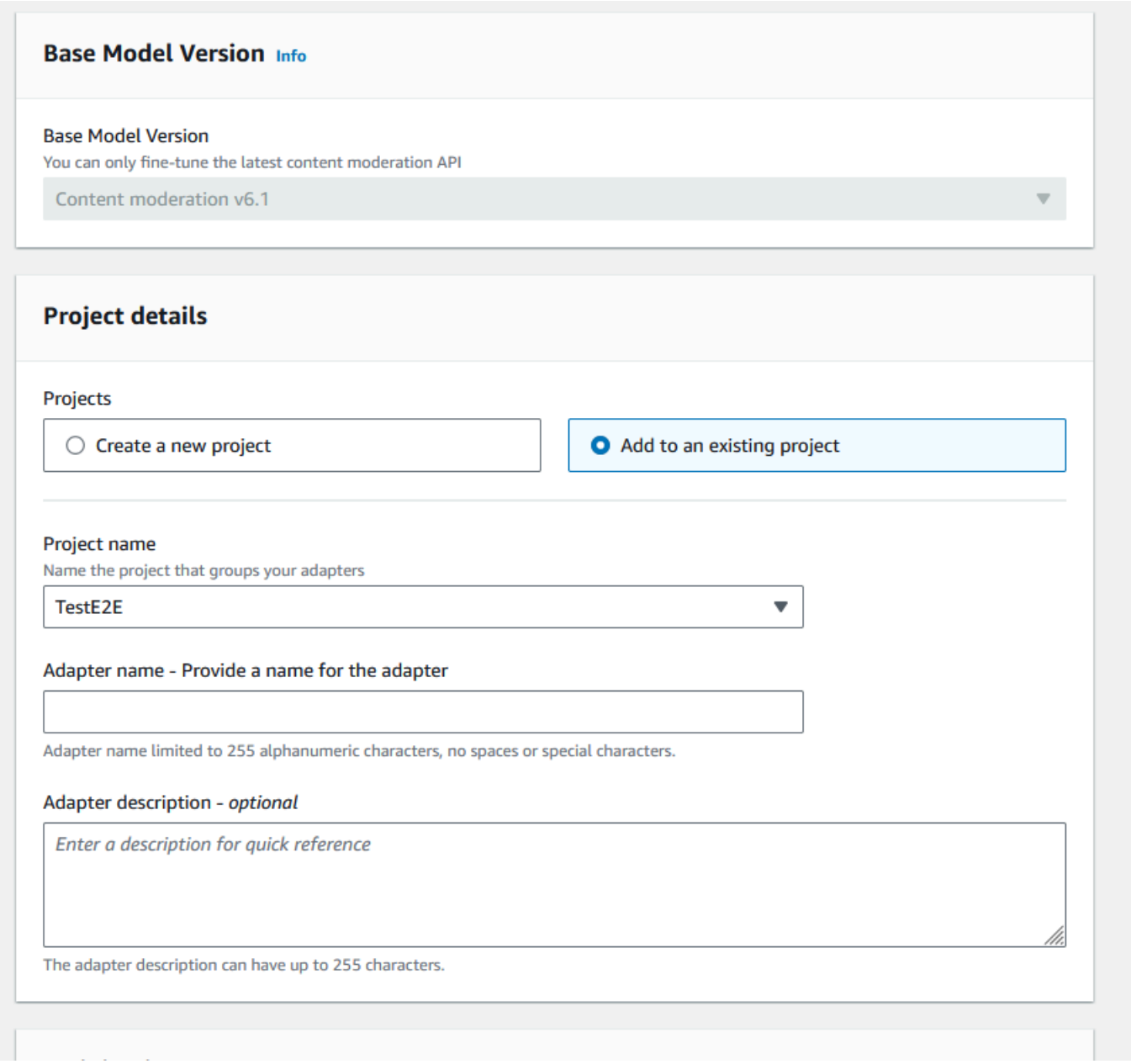
2. Seleziona l'adattatore che desideri recensire da questo elenco. Nella seguente pagina dei dettagli dell'adattatore, puoi visualizzare una serie di metriche relative all'adattatore.



3. Tramite il pannello Soglia è possibile modificare la soglia di confidenza minima che l'adattatore deve avere per assegnare un'etichetta a un'immagine. Il numero di istanze contrassegnate e non contrassegnate cambierà man mano che regolate la soglia di confidenza. Puoi anche filtrare per categoria di etichette per visualizzare le metriche relative alle categorie che hai selezionato. Imposta la soglia prescelta.
4. Puoi valutare le prestazioni dell'adattatore sulla base dei dati di test esaminando le metriche nel pannello Adapter Performance. Queste metriche vengono calcolate confrontando le estrazioni dell'adattatore con le annotazioni «verità di base» sul set di test.

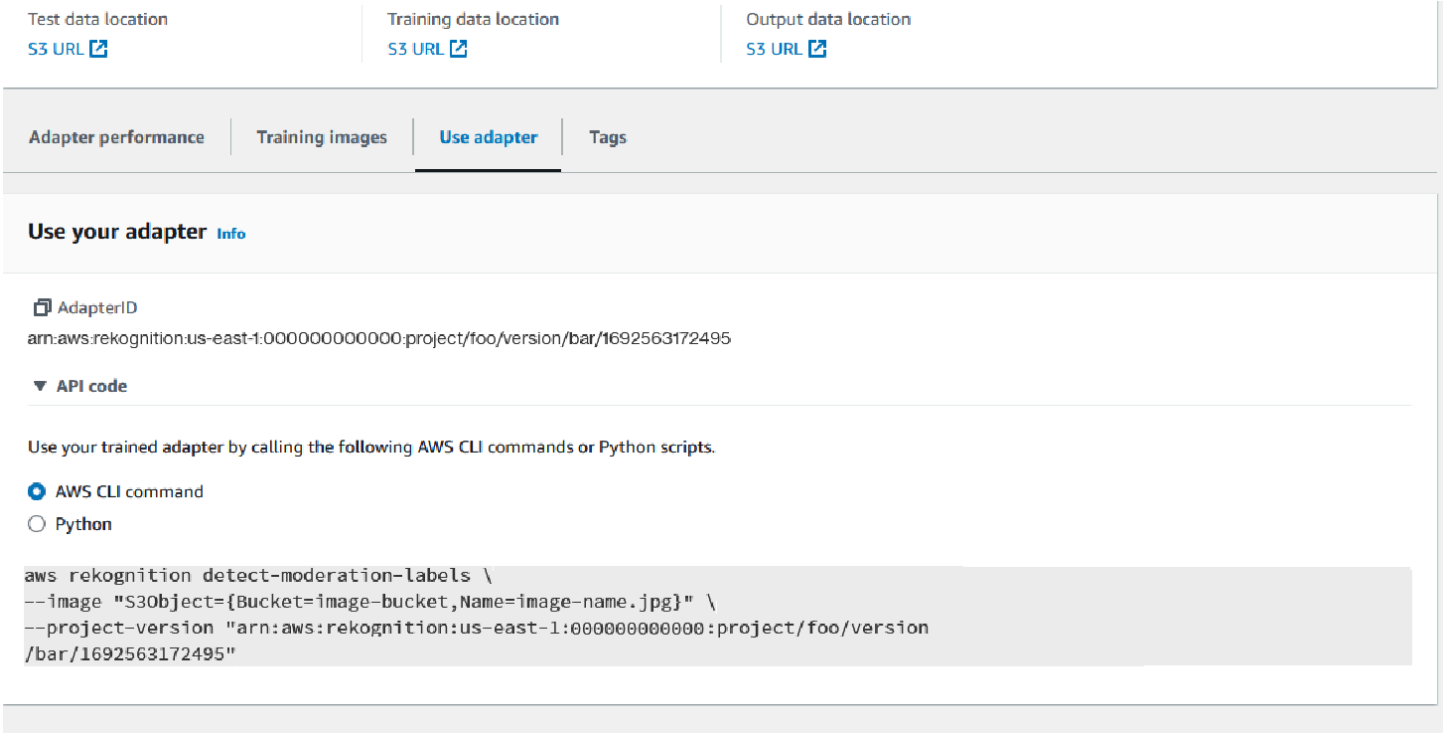
Il pannello delle prestazioni dell'adattatore mostra i tassi di miglioramento dei falsi positivi e dei falsi negativi relativi all'adattatore che avete creato. La scheda performance per etichetta può essere utilizzata per confrontare le prestazioni dell'adattatore e del modello base per ciascuna categoria di etichette. Mostra il conteggio delle previsioni false positive e false negative sia per il modello di base che per l'adattatore, stratificate per categoria di etichette. Esaminando queste metriche è possibile determinare dove è necessario migliorare l'adattatore. Per ulteriori informazioni su tali parametri, consulta Valutazione e miglioramento dell'adattatore.

Per migliorare le prestazioni, puoi raccogliere più immagini di allenamento e quindi creare un nuovo adattatore basato sull'interno del progetto. Basta tornare alla pagina di destinazione moderazione personalizzata e creare un nuovo adattatore all'interno del progetto, fornendo altre immagini di formazione su cui addestrare l'adattatore. Questa volta scegli l'opzione Aggiungi a un progetto esistente anziché Crea un nuovo progetto e seleziona il progetto in cui desideri creare il nuovo adattatore dal menu a tendina Nome progetto. Come prima, annota le tue immagini o fornisci un file manifesto con le annotazioni.



Utilizzo dell'adattatore

Dopo aver creato l'adattatore, puoi fornirlo a un'operazione Rekognition supportata come. DetectModerationLabels Per vedere esempi di codice che puoi usare per eseguire inferenze con il tuo adattatore, seleziona la scheda «Usa adattatore», dove puoi vedere esempi di codice sia per la AWS CLI che per Python. Puoi anche visitare la rispettiva sezione della documentazione relativa all'operazione per cui hai creato un adattatore per vedere altri esempi di codice, istruzioni di configurazione e un esempio di JSON.



Eliminazione dell'adattatore e del progetto

Puoi eliminare singoli adattatori o eliminare il tuo progetto. Prima di eliminare il progetto stesso, devi eliminare ogni adattatore associato al progetto.

1. Per eliminare un adattatore associato al progetto, scegliete l'adattatore, quindi scegliete Elimina.
2. Per eliminare un progetto, scegli il progetto che desideri eliminare, quindi seleziona Elimina.

Valutazione e miglioramento dell'adattatore

Dopo ogni ciclo di formazione sugli adattatori, ti consigliamo di esaminare le metriche delle prestazioni nello strumento Rekognition Console per determinare quanto l'adattatore si avvicina al livello di prestazioni desiderato. Puoi quindi migliorare ulteriormente la precisione dell'adattatore per le tue immagini caricando un nuovo batch di immagini di addestramento e addestrando un nuovo adattatore all'interno del tuo progetto. Dopo aver creato una versione migliorata dell'adattatore, puoi utilizzare la console per eliminare tutte le versioni precedenti dell'adattatore che non ti servono più.

Puoi anche recuperare le metriche utilizzando l'operazione API. DescribeProjectVersions

Parametri prestazionali

Una volta terminato il processo di formazione e creato l'adattatore, è importante valutare in che modo l'adattatore estrae le informazioni dalle immagini.

Nella Rekognition Console sono disponibili due metriche per aiutarti ad analizzare le prestazioni dell'adattatore: miglioramento falso positivo e miglioramento falso negativo.

Puoi visualizzare queste metriche per qualsiasi adattatore selezionando la scheda «Prestazioni dell'adattatore» nella parte dedicata agli adattatori della console. Il pannello delle prestazioni dell'adattatore mostra i tassi di miglioramento dei falsi positivi e dei falsi negativi relativi all'adattatore che avete creato.

Il miglioramento dei falsi positivi misura il miglioramento del riconoscimento dei falsi positivi da parte dell'adattatore rispetto al modello base. Se il valore di miglioramento dei falsi positivi è del 25%, significa che l'adattatore ha migliorato il riconoscimento dei falsi positivi del 25% nel set di dati del test.

Il miglioramento dei falsi negativi misura il miglioramento del riconoscimento dei falsi negativi da parte dell'adattatore rispetto al modello base. Se il valore di miglioramento dei falsi negativi è del 25%, significa che l'adattatore ha migliorato il riconoscimento dei falsi negativi del 25% nel set di dati del test.

La scheda performance per etichetta può essere utilizzata per confrontare le prestazioni dell'adattatore e del modello base per ciascuna categoria di etichette. Mostra il conteggio delle previsioni false positive e false negative sia per il modello di base che per l'adattatore, stratificate per categoria di etichette. Esaminando queste metriche è possibile determinare dove è necessario migliorare l'adattatore.

Ad esempio, se il tasso di falsi negativi del modello di base per la categoria etichetta alcol è 15 mentre il tasso di falsi negativi dell'adattatore è 15 o superiore, sapete che dovreste concentrarvi sull'aggiunta di altre immagini contenenti l'etichetta alcol quando create un nuovo adattatore.

Quando si utilizzano le operazioni dell'API Rekognition, la metrica F1-Score viene restituita quando si chiama l'operazione. DescribeProjectVersions

Migliora il tuo modello

L'implementazione degli adattatori è un processo iterativo, poiché probabilmente dovrai addestrare un adattatore più volte per raggiungere il livello di precisione desiderato. Dopo aver creato e addestrato l'adattatore, ti consigliamo di testarlo e valutarne le prestazioni su vari tipi di etichette.

Se la precisione dell'adattatore è carente in qualsiasi area, aggiungi nuovi esempi di quelle immagini per aumentare le prestazioni dell'adattatore per quelle etichette. Provate a fornire all'adattatore esempi diversi e aggiuntivi che riflettano i casi in cui non funziona. Fornire all'adattatore immagini rappresentative e varie consente di gestire diversi esempi del mondo reale.

Dopo aver aggiunto nuove immagini al set di allenamento, riabilita l'adattatore, quindi rivalutalo sul set di test e sulle etichette. Ripeti questa procedura finché l'adattatore non raggiunge il livello di prestazioni desiderato. Se fornite immagini e annotazioni più rappresentative, i punteggi falsi positivi e falsi negativi miglioreranno gradualmente nel corso delle successive iterazioni di allenamento.

Formati di file manifest

Le seguenti sezioni mostrano esempi dei formati di file manifest per i file di input, output e valutazione.

Manifest di input

Un file manifest è un file delimitato da righe json, in cui ogni riga contiene un codice JSON che contiene informazioni su una singola immagine.

Ogni voce del manifest di input deve contenere il campo `source-ref` con il percorso dell'immagine nel bucket Amazon S3 e, per la moderazione personalizzata, il campo `content-moderation-groundtruth` con le annotazioni di base. Tutte le immagini in un set di dati dovrebbero trovarsi nello stesso bucket. La struttura è comune sia ai file manifest di addestramento che a quelli di test.

L'operazione `CreateProjectVersion` di moderazione personalizzata utilizza le informazioni fornite nel manifest di input per addestrare un adattatore.

L'esempio seguente è una riga di un file manifest per una singola immagine che contiene una sola classe non sicura:

```
{
   "source-ref": "s3://foo/bar/1.jpg",
   "content-moderation-groundtruth": {
        "ModerationLabels": [
            {
                "Name": "Rude Gesture"
            }
        ]
   }
}
```

L'esempio seguente è una riga di un file manifest per una singola immagine non sicura che contiene più classi non sicure, in particolare nudità e gesti maleducati.

```
{
   "source-ref": "s3://foo/bar/1.jpg",
   "content-moderation-groundtruth": {
        "ModerationLabels": [
            {
                "Name": "Rude Gesture"
            },
            {
                "Name": "Nudity"
            }
        ]
   }
}
```

L'esempio seguente è una riga di un file manifest per una singola immagine che contiene una sola classe non sicura:

```
{
   "source-ref": "s3://foo/bar/1.jpg",
   "content-moderation-groundtruth": {
        "ModerationLabels": []
   }
}
```

Per l'elenco completo delle etichette supportate, consulta Moderazione dei contenuti.

Manifest di output

Al termine di un processo di formazione, viene restituito un file manifest di output. Un file manifest è un file delimitato da righe JSON, in cui ogni riga contiene un codice JSON che contiene informazioni su una singola immagine. Amazon S3 Path to the OutputManifest può essere ottenuto dalla `DescribeProjectVersion` risposta:

- `TrainingDataResult.Output.Assets[0].GroundTruthManifest.S3Object` per il set di dati di addestramento

- `TestingDataResult.Output.Assets[0].GroundTruthManifest.S3Object` per testare il set di dati

Le seguenti informazioni vengono restituite per ogni voce del manifest di output:

Nome chiave	Descrizione
`source-ref`	Riferimento a un'immagine in s3 fornita nel manifesto di input
`content-moderation-groundtruth`	Annotazioni di base relative alla verità fornite nel manifesto di input
`detect-moderation-labels`	Previsioni dell'adattatore, solo parte del set di dati di test
`detect-moderation-labels-base-model`	Previsioni del modello di base, solo parte del set di dati di test

Le previsioni dell'adattatore e del modello di base vengono restituite alla ConfidenceTrehsold versione 5.0 nel formato simile alla risposta. DetectModerationLabels

L'esempio seguente mostra la struttura delle previsioni dei modelli Adapter e Base:

```
{
   "ModerationLabels": [
      {
         "Confidence": number,
         "Name": "string",
         "ParentName": "string"
      }
   ],
   "ModerationModelVersion": "string",
   "ProjectVersion": "string"
}
```

Per l'elenco completo delle etichette restituite, consulta Moderazione dei contenuti.

Risultati della valutazione manifest

Al termine di un processo di addestramento, viene restituito un file manifest. Il manifest dei risultati della valutazione è un file JSON generato dal processo di addestramento e contiene informazioni sulle prestazioni dell'adattatore sui dati del test.

Il percorso di Amazon S3 verso il manifesto dei risultati della valutazione può essere ottenuto dal `EvaluationResult.Summary.S3Object` campo nella DescribeProejctVersion risposta.

Il seguente esempio illustra la struttura dei risultati della valutazione manifest:

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

Il file della valutazione manifest contiene:

- Risultati aggregati come definiti da `F1Score`
- Dettagli sul lavoro di valutazione ProjectVersionArn, tra cui il numero di immagini di formazione, il numero di immagini di test e le etichette su cui è stato addestrato l'adattatore.
- FalseNegative Risultati aggregati TruePositive, TrueNegative FalsePositive, e relativi alle prestazioni sia del modello di base che dell'adattatore.
- Per etichetta TruePositive, TrueNegative FalsePositive, e FalseNegative risultati relativi alle prestazioni del modello base e dell'adattatore, calcolati in base alla soglia di confidenza di input.
- Risultati aggregati e per etichetta TruePositive, TrueNegative FalsePositive, e FalseNegative risultati relativi alle prestazioni del modello di base e dell'adattatore a diverse soglie di confidenza. La soglia di confidenza varia da 5 a 100 a intervalli di 5.

Best practice per l'addestramento degli adattatori

Si consiglia di attenersi alle seguenti best practice durante la creazione, l'addestramento e l'utilizzo degli adattatori:

1. I dati dell'immagine di esempio devono catturare gli errori rappresentativi che i clienti intendono eliminare. Se il modello commette errori ripetuti su immagini visivamente simili, assicurati di portare molte di quelle immagini per l'addestramento.

2. Invece di inserire solo immagini in cui la modella commette errori su una particolare etichetta di moderazione, assicurati anche di inserire immagini che dimostrino che la modella non commette errori sull'etichetta Moderazione.
3. Fornire un minimo di 50 campioni di falsi negativi OPPURE 20 campioni di falsi positivi per l'addestramento e un minimo di 20 campioni per i test. Tuttavia, fornite quante più immagini annotate possibile per migliorare le prestazioni dell'adattatore.
4. Annotazione di tutte le etichette che ritieni importanti per tutte le immagini: se decidi di dover annotare l'occorrenza di un'etichetta su un'immagine, assicurati di annotare l'occorrenza di questa etichetta su tutte le altre immagini.
5. I dati dell'immagine di esempio devono contenere quante più varianti possibili sull'etichetta, concentrandosi su istanze rappresentative delle immagini che verranno analizzate in un ambiente di produzione.

Impostazione delle AutoUpdate autorizzazioni

Rekognition supporta AutoUpdate la funzionalità per adattatori personalizzati. Ciò significa che la riqualificazione automatizzata viene data la massima priorità quando AutoUpdate Flag è ABILITATO su un progetto. Questi aggiornamenti automatici richiedono l'autorizzazione ad accedere ai set di dati di formazione/test e alla AWS KMS chiave con cui addestrare l'adattatore per il cliente. Puoi fornire queste autorizzazioni seguendo la procedura seguente.

Autorizzazioni Bucket Amazon S3

Per impostazione predefinita, tutti gli oggetti e i bucket Amazon S3 sono privati. Solo il proprietario della risorsa, l' AWS account che ha creato il bucket, può accedere al bucket e a tutti gli oggetti in esso contenuti. Tuttavia, il proprietario della risorsa può concedere le autorizzazioni di accesso ad altre risorse e ad altri utenti mediante una policy di bucket.

Se desideri creare o modificare un bucket Amazon S3 da utilizzare come fonte di set di dati di input e destinazione dei risultati di addestramento in un corso di addestramento dell'adattatore personalizzato, devi modificare ulteriormente la policy del bucket. Per leggere o scrivere in un bucket Amazon S3, Rekognition deve disporre delle seguenti autorizzazioni.

Rekognition ha richiesto la policy Amazon S3

Rekognition richiede una politica di autorizzazione con i seguenti attributi:

- Statement ID SID

- Nome del bucket
- Nome principale del servizio per Rekognition.
- Le risorse necessarie per Rekognition, il bucket e tutto il suo contenuto
- Le azioni necessarie che Rekognition deve intraprendere.

La seguente policy consente a Rekognition di accedere a un bucket Amazon S3 durante la riqualificazione automatica.

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

Puoi seguire questa guida per aggiungere la policy sui bucket di cui sopra al tuo bucket S3.

Puoi trovare ulteriori informazioni sulle policy relative ai bucket qui.

AWS KMS Autorizzazioni chiave

Rekognition ti consente di fornire un adattatore personalizzato KmsKeyId opzionale durante l'addestramento. Se fornita, Rekognition utilizza questa chiave per crittografare le immagini di addestramento e test copiate nel servizio per l'addestramento dei modelli. La chiave viene anche

utilizzata per crittografare i risultati della formazione e i file manifest scritti nel OutputConfig bucket di output Amazon S3 ().

Se scegli di fornire una chiave KMS come input per il corso di addestramento personalizzato per gli adattatori (ad esempio `Rekognition:CreateProjectVersion`), devi modificare ulteriormente la policy della chiave KMS per consentire al principale del servizio Rekognition di utilizzare questa chiave per la riqualificazione automatica in futuro. Rekognition deve disporre delle seguenti autorizzazioni.

Politica chiave di Rekognition Required AWS KMS

Amazon Rekognition richiede una politica di autorizzazione con i seguenti attributi:

- Statement ID SID
- Nome principale del servizio per Amazon Rekognition.
- Le azioni necessarie che Amazon Rekognition deve intraprendere.

La seguente policy della chiave consente ad Amazon Rekognition di accedere a una chiave Amazon KMS durante la riqualificazione automatica:

```
{
  "Version": "2023-10-06",
  "Statement": [
    {
      "Sid": "KeyPermissions",
      "Effect": "Allow",
      "Principal": {
            "Service": "rekognition.amazonaws.com"
        },
      "Action": [
        "kms:DescribeKey",
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ]
      "Resource": "*"
    }
  ]
}
```

Puoi seguire questa guida per aggiungere la AWS KMS politica di cui sopra alla tua chiave. AWS KMS

Puoi trovare ulteriori informazioni sulle AWS KMS politiche qui.

AWS Notifica Health Dashboard per Rekognition

Your AWS Health Dashboard fornisce supporto per le notifiche provenienti da Rekognition. Queste notifiche forniscono informazioni e indicazioni sulla correzione delle modifiche programmate nei modelli Rekognition che potrebbero influire sulle tue applicazioni. Al momento sono disponibili solo gli eventi specifici della funzionalità di moderazione dei contenuti di Rekognition.

L' AWS Health Dashboard fa parte del servizio AWS sanitario. Questo servizio non richiede l'installazione e può essere visualizzato da qualsiasi utente autenticato nell'account. Per ulteriori informazioni, consulta Nozioni di base su Dashboard AWS Health.

Se visualizzi una notifica simile a uno dei seguenti, questo deve essere gestito come un allarme su cui intervenire.

Esempio di notifica: è disponibile una nuova versione del modello per Rekognition Content Moderation.

Rekognition pubblica `AWS_MODERATION_MODEL_VERSION_UPDATE_NOTIFICATION` l'evento su Health Dashboard per AWS indicare che è stata rilasciata una nuova versione del modello di moderazione. Questo evento è importante se si utilizza l' DetectModerationLabels API e gli adattatori con questa API. I nuovi modelli possono influire sulla qualità a seconda del caso d'uso e alla fine sostituiranno le versioni precedenti del modello. Si consiglia di convalidare la qualità del modello e di tenere presente le tempistiche di aggiornamento del modello quando si riceve questo avviso.

Se ricevi una notifica di aggiornamento della versione del modello, devi trattarla come un allarme per intervenire. Se non utilizzi adattatori, dovresti valutare la qualità del modello aggiornato in base al tuo caso d'uso esistente. Se si utilizzano adattatori, è necessario addestrare nuovi adattatori con il modello aggiornato e valutarne la qualità. Se disponi di un set di addestramento automatico, i nuovi adattatori verranno addestrati automaticamente e quindi potrai valutarne la qualità.

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

Consulta Monitoraggio degli eventi di AWS Health con Amazon EventBridge per rilevare e reagire agli eventi AWS Health utilizzando EventBridge.

Revisione di contenuti non appropriati con Amazon Augmented AI

IA aumentata Amazon (Amazon A2I) consente di creare i flussi di lavoro necessari per la revisione umana delle previsioni di machine learning.

Amazon Rekognition è direttamente integrato con Amazon A2I in modo che sia possibile implementare facilmente la revisione umana per il caso d'uso del rilevamento di immagini non sicure. Amazon A2I fornisce un flusso di lavoro di revisione umana per la moderazione di immagini. Ciò consente di riesaminare facilmente le previsioni di Amazon Rekognition. È possibile definire soglie di affidabilità per il caso d'uso e regolarle nel tempo. Con Amazon A2I, puoi utilizzare un pool di revisori all'interno della tua organizzazione o Amazon Mechanical Turk. Puoi avvalerti anche di fornitori di forza lavoro che sono stati pre-selezionati da AWS per la qualità e la conformità alle procedure di sicurezza.

Le fasi seguenti illustrano come configurare Amazon A2I con Amazon Rekognition. Innanzitutto, creare una definizione di flusso con Amazon A2I che presenti le condizioni che attivano la revisione umana. Quindi, si passa la definizione di flusso del nome della risorsa Amazon (ARN) all'operazione `DetectModerationLabel` di Amazon Rekognition. Nella risposta `DetectModerationLabel`, puoi vedere se è necessaria la revisione umana. I risultati della revisione umana sono disponibili in un bucket Amazon S3 impostato dalla definizione del flusso.

Per visualizzare una end-to-end dimostrazione di come usare Amazon A2I con Amazon Rekognition, consulta uno dei seguenti tutorial nella Amazon AI Developer Guide. SageMaker

- Demo: introduzione alla console Amazon A2I
- Demo: introduzione all'API Amazon A2I

  Per iniziare a utilizzare l'API, puoi anche eseguire un notebook Jupyter di esempio. Vedi Utilizzare un'istanza SageMaker Notebook con Amazon A2I Jupyter Notebook per utilizzare l'integrazione del notebook Amazon Augmented AI (Amazon A2I) con Amazon Rekognition [Esempio] in un'istanza di notebook AI. SageMaker

Funzionamento DetectModerationLabels con Amazon A2I

> **Note**
>
> Crea tutte le risorse Amazon A2I e Amazon Rekognition nella stessa regione AWS.

1. Completa i prerequisiti elencati in Getting Started with Amazon Augmented AI nella SageMaker documentazione AI.

   Inoltre, ricorda di configurare le autorizzazioni IAM come indicato nella pagina Autorizzazioni e sicurezza in Amazon Augmented AI nella SageMaker documentazione AI.
2. Segui le istruzioni per la creazione di un flusso di lavoro di revisione umana nella documentazione sull'SageMaker intelligenza artificiale.

   Un flusso di lavoro di revisione umana gestisce l'elaborazione di un'immagine. Contiene le condizioni che attivano una revisione umana, il team di lavoro a cui viene inviata l'immagine, il modello di interfaccia utente utilizzato dal team di lavoro e il bucket Amazon S3 a cui vengono inviati i risultati del team di lavoro.

All'interno della `CreateFlowDefinition` chiamata, è necessario impostare il valore `HumanLoopRequestSource` su "AWS/Rekognition/DetectModerationLabels/Image/V3". Dopodiché, devi decidere come impostare le tue condizioni che attivano la revisione umana.

Con Amazon Rekognition hai due opzioni per `ConditionType`: `ModerationLabelConfidenceCheck` e `Sampling`.

`ModerationLabelConfidenceCheck` crea un loop umano quando l'attendibilità di un'etichetta di moderazione è all'interno di un intervallo. Infine, `Sampling` invia una percentuale casuale dei documenti elaborati per la revisione umana. Ogni `ConditionType` utilizza un set diverso di `ConditionParameters` per impostare i risultati attesi dalla revisione umana.

`ModerationLabelConfidenceCheck` ha il `ConditionParameters` `ModerationLableName` che imposta la chiave che deve sottoposta a revisione umana. Inoltre, ha fiducia, che imposta l'intervallo percentuale per l'invio a una recensione umana con LessThan GreaterThan, ed Equals. `Samplingha` `RandomSamplingPercentage` che stabilisce una percentuale di documenti che verranno inviati alla revisione umana.

L'esempio di codice riportato di seguito è una chiamata parziale di `CreateFlowDefinition`. Invia un'immagine per la revisione umana se è valutata meno del 98% sull'etichetta "Provocante" e più del 95% sull'etichetta "Costumi da bagno femminili o biancheria intima". Ciò significa che se l'immagine non è considerata provocante ma contiene una donna in biancheria intima o costume da bagno, è possibile ricontrollare l'immagine utilizzando la revisione umana.

```
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` restituisce un `FlowDefinitionArn`, che si utilizza nella fase successiva quando si chiama `DetectModerationLabels`.

Per ulteriori informazioni, CreateFlowDefinitionconsulta l' SageMaker AI API Reference.

3. Impostare il parametro `HumanLoopConfig` quando si chiama `DetectModerationLabels`, come in Rilevamento di immagini inappropriate. Consulta il passaggio 4 per esempi di chiamata `DetectModerationLabels` con `HumanLoopConfig` impostato.

   a. All'interno del parametro `HumanLoopConfig`, `FlowDefinitionArn` impostare l'ARN della definizione di flusso creata nella fase 2.

   b. Impostare `HumanLoopName`. Dovrebbe essere univoco all'interno di una regione e deve essere in caratteri minuscoli.

   c. (Opzionale) È possibile utilizzare `DataAttributes` per impostare se l'immagine passata ad Amazon Rekognition contiene o meno informazioni di identificazione personale. È necessario impostare questo parametro per inviare informazioni ad Amazon Mechanical Turk.

4. Esegui `DetectModerationLabels`.

   Gli esempi seguenti mostrano come utilizzare AWS CLI e AWS SDK per Python (Boto3) eseguire `DetectModerationLabels` `HumanLoopConfig` set.

AWS SDK per Python (Boto3)

Gli esempi seguenti utilizzano il kit SDK per Python (Boto3). Per ulteriori informazioni, consulta detect_moderation_labels nella Documentazione di riferimento delle API AWS SDK per 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

Il seguente esempio di richiesta utilizza la AWS CLI. Per ulteriori informazioni, consulta la sezione detect-moderation-labels nella Documentazione di riferimento della AWS CLI.

```
$ aws rekognition detect-moderation-labels \
    --image "S3Object={Bucket='bucket_name',Name='image_name'}" \
    --human-loop-config
 HumanLoopName="human_loop_name",FlowDefinitionArn="arn:aws:sagemaker:aws-
region:aws_account_number:flow-
definition/
flow_def_name",DataAttributes='{ContentClassifiers=["FreeOfPersonallyIdentifiableInforma
 "FreeOfAdultContent"]}'
```

```
$ aws rekognition detect-moderation-labels \
    --image "S3Object={Bucket='bucket_name',Name='image_name'}" \
    --human-loop-config \
        '{"HumanLoopName": "human_loop_name", "FlowDefinitionArn":
 "arn:aws:sagemaker:aws-region:aws_account_number:flow-
```

```
definition/flow_def_name", "DataAttributes": {"ContentClassifiers":
 ["FreeOfPersonallyIdentifiableInformation", "FreeOfAdultContent"]}}'
```

Quando esegui `DetectModerationLabels` con `HumanLoopConfig` enabled, Amazon Rekognition SageMaker chiama l'operazione API AI. `StartHumanLoop` Questo comando prende la risposta da `DetectModerationLabels` e la controlla rispetto alle condizioni della definizione di flusso nell'esempio. Se soddisfa le condizioni per la revisione, restituisce un `HumanLoopArn`. Ciò significa che i membri del team di lavoro impostati nella definizione del flusso ora possono rivedere l'immagine. La chiamata all'operazione di runtime di Amazon Augmented AI `DescribeHumanLoop` fornisce informazioni sull'esito del ciclo. Per ulteriori informazioni, consulta la DescribeHumanLoopdocumentazione di riferimento dell'API Amazon Augmented AI.

Dopo aver esaminato l'immagine, è possibile visualizzare i risultati nel bucket specificato nel percorso di output della definizione di flusso. Amazon A2I ti invierà inoltre una notifica tramite Amazon CloudWatch Events una volta completata la revisione. Per vedere quali eventi cercare, consulta CloudWatch Events in the SageMaker AI Documentation.

Per ulteriori informazioni, consulta Getting Started with Amazon Augmented AI nella documentazione SageMaker sull'IA.

Rilevamento del testo

Amazon Rekognition è in grado di rilevare testo in immagini e video. Può quindi convertire il testo rilevato in testo leggibile dalla macchina. È possibile utilizzare il rilevamento del testo leggibile dalla macchina nelle immagini per implementare soluzioni quali:

- Ricerche visive. Ad esempio, per recuperare e visualizzare immagini contenenti lo stesso testo.
- Informazioni sui contenuti. Ad esempio, per fornire informazioni su temi ricorrenti nel testo riconosciuto in fotogrammi video estratti. L'applicazione è in grado di eseguire ricerche di testo riconosciuto per contenuti rilevanti, ad esempio notizie, risultati sportivi, numeri di maglia degli atleti e didascalie.
- Navigazione. Ad esempio, per sviluppare un'app abilitata per la sintesi vocale per persone ipovedenti in grado di riconoscere i nomi di ristoranti e negozi o i cartelli stradali.
- Supporto per il trasporto e la sicurezza pubblica. Ad esempio, per rilevare i numeri di targa dalle immagini delle videocamere stradali.
- Filtraggio. Ad esempio, per filtrare Informazioni di identificazione personale (PII) dalle immagini.

Per il rilevamento del testo nei video, è possibile implementare soluzioni quali:

- Ricerca di video per clip con parole chiave di testo specifiche, ad esempio il nome dell'ospite su un grafico in un notiziario.
- Moderazione dei contenuti per garantire la conformità agli standard organizzativi mediante il rilevamento di testo accidentale, parolacce o spam.
- Trovare tutte le sovrapposizioni di testo sulla timeline video per ulteriori elaborazioni, ad esempio la sostituzione con testo in un'altra lingua per l'internazionalizzazione dei contenuti.
- Trovare posizioni di testo, in modo che altri elementi grafici possano essere allineati di conseguenza.

Per rilevare il testo nelle immagini in formato JPEG o PNG, utilizzare l'DetectTextoperazione.
Per rilevare in modo asincrono il testo nel video, utilizzate le operazioni and.
StartTextDetectionGetTextDetection Le operazioni di rilevamento sia di immagini che di video supportano la maggior parte dei tipi di carattere, inclusi quelli altamente stilizzati. Dopo aver rilevato il testo, Amazon Rekognition crea una rappresentazione di parole e linee di testo rilevate, mostra la relazione tra di loro e indica dove si trova il testo su un'immagine o una cornice di video.

Le operazioni `DetectText` e `GetTextDetection` rilevano parole e righe. Una parola è costituita da uno o più caratteri dello script che non sono separati da spazi. `DetectText`è in grado di rilevare fino a 100 parole in un'immagine. `GetTextDetection`è inoltre in grado di rilevare fino a 100 parole per frame di video.

Una parola corrisponde a uno o più caratteri non separati da spazi. Amazon Rekognition è progettato per rilevare parole in inglese, arabo, russo, tedesco, francese, italiano, portoghese e spagnolo.

Una riga è una stringa di parole equidistanti Una riga non è necessariamente una frase completa (i punti non indicano la fine di una riga). Ad esempio, Amazon Rekognition rileva un numero di patente come una riga. Una riga termina quando non c'è testo allineato dopo di essa o quando c'è un ampio spazio tra le parole, rispetto alla lunghezza delle parole. In pratica, a seconda dello spazio tra le parole, Amazon Rekognition potrebbe rilevare più righe di testo allineate nella stessa direzione. Se una frase si espande su più righe, l'operazione restituisce più righe.

Considera l'immagine seguente:

Le caselle blu rappresentano informazioni sul testo rilevato e la posizione del testo restituito dall'operazione `DetectText`. In questo esempio, Amazon Rekognition rileva «È», «LUNEDÌ», «ma», «continua a» e «sorridere» come parole. Amazon Rekognition rileva «È», «LUNEDÌ», «ma»,

«continua a» e «sorridere» come righe. Per essere rilevato, l'orientamento del testo deve essere +/- 90° sull'asse orizzontale.

Per vedere un esempio, consulta Rilevamento di testo in un'immagine.

Argomenti

- Rilevamento di testo in un'immagine
- Rilevamento del testo in un video archiviato

Rilevamento di testo in un'immagine

Puoi fornire un'immagine di input come matrice di byte dell'immagine (byte dell'immagine codificata in formato Base64) o come oggetto di Amazon S3. In questa procedura, puoi caricare un'immagine in formato JPEG o PNG nel bucket S3 e specificare il nome file.

Per rilevare il testo in un'immagine (API)

1. Qualora non fossero già stati soddisfatti, completa i seguenti prerequisiti:

   a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

   b. Installa e configura il e il. AWS Command Line Interface AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Carica l'immagine contenente testo nel bucket S3.

   Per le istruzioni, consulta Caricamento di oggetti in Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

3. Utilizzare i seguenti esempi per richiamare l'operazione `DetectText`.

   Java

   Il codice di esempio seguente mostra le righe e le parole rilevate in un'immagine.

   Sostituisci i valori `amzn-s3-demo-bucket` e `photo` con i nomi del bucket S3 e dell'immagine utilizzati nella fase 2.

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

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

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.DetectTextRequest;
import com.amazonaws.services.rekognition.model.DetectTextResult;
import com.amazonaws.services.rekognition.model.TextDetection;
import java.util.List;



public class DetectText {

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


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

      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();



      DetectTextRequest request = new DetectTextRequest()
              .withImage(new Image()
              .withS3Object(new S3Object()
              .withName(photo)
              .withBucket(bucket)));


      try {
         DetectTextResult result = rekognitionClient.detectText(request);
         List<TextDetection> textDetections = result.getTextDetections();

         System.out.println("Detected lines and words for " + photo);
         for (TextDetection text: textDetections) {

                 System.out.println("Detected: " + text.getDetectedText());
```

```
                System.out.println("Confidence: " +
 text.getConfidence().toString());
                System.out.println("Id : " + text.getId());
                System.out.println("Parent Id: " + text.getParentId());
                System.out.println("Type: " + text.getType());
                System.out.println();
            }
         } catch(AmazonRekognitionException e) {
            e.printStackTrace();
         }
      }
}
```

Java V2

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
/**
*  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

Questo AWS CLI comando visualizza l'output JSON per l'operazione `detect-text` CLI.

Sostituisci i valori `amzn-s3-demo-bucket` e `Name` con i nomi del bucket S3 e dell'immagine utilizzati nella fase 2.

Sostituisci il valore di `profile_name` con il nome del tuo profilo di sviluppatore.

```
aws rekognition detect-text  --image "{"S3Object":{"Bucket":"amzn-s3-demo-
bucket","Name":"image-name"}}" --profile default
```

Se accedi alla CLI da un dispositivo Windows, usa le virgolette doppie anziché le virgolette singole ed evita le virgolette doppie interne tramite barra rovesciata (ovvero, \) per risolvere eventuali errori del parser che potresti riscontrare. Per un esempio, consulta quanto segue:

```
aws rekognition detect-text  --image "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --profile default
```

Python

Il codice di esempio seguente mostra le righe e le parole rilevate in un'immagine.

Sostituisci i valori `amzn-s3-demo-bucket` e `photo` con i nomi del bucket S3 e dell'immagine utilizzati nella fase 2. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

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

Il codice di esempio seguente mostra le righe e le parole rilevate in un'immagine.

Sostituisci i valori `amzn-s3-demo-bucket` e `photo` con i nomi del bucket S3 e dell'immagine utilizzati nella fase 2.

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

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

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

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        DetectTextRequest detectTextRequest = new DetectTextRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
```

```
                {
                    Name = photo,
                    Bucket = bucket
                }
            }
        };

        try
        {
            DetectTextResponse detectTextResponse =
 rekognitionClient.DetectText(detectTextRequest);
            Console.WriteLine("Detected lines and words for " + photo);
            foreach (TextDetection text in detectTextResponse.TextDetections)
            {
                Console.WriteLine("Detected: " + text.DetectedText);
                Console.WriteLine("Confidence: " + text.Confidence);
                Console.WriteLine("Id : " + text.Id);
                Console.WriteLine("Parent Id: " + text.ParentId);
                Console.WriteLine("Type: " + text.Type);
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```

Node.JS

Il codice di esempio seguente mostra le righe e le parole rilevate in un'immagine.

Sostituisci i valori `amzn-s3-demo-bucket` e `photo` con i nomi del bucket S3 e dell'immagine utilizzati nella fase 2. Sostituisci il valore di `region` con la regione trovata nelle tue credenziali .aws. Sostituisci il valore di `profile_name` nella riga che crea la sessione di Rekognition con il nome del tuo profilo di sviluppatore.

```
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 richiesta di operazione

Nell'operazione `DetectText` puoi specificare un'immagine di input come matrice di byte con codifica base64 o come immagine archiviata in un bucket Amazon S3. La seguente richiesta JSON di esempio mostra l'immagine caricata da un bucket Amazon S3.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
```

```
            "Name": "inputtext.jpg"
        }
    }
}
```

Filtri

La possibilità di filtrare per regione del testo, dimensione e punteggio di affidabilità offre la flessibilità necessaria a controllare l'output del rilevamento di testo. Utilizzando le regioni di interesse, è possibile limitare facilmente il rilevamento del testo alle regioni pertinenti, ad esempio in alto a destra della foto del profilo o di una posizione fissa in relazione a un punto di riferimento quando si leggono i numeri delle parti da un'immagine di una macchina. Il filtro relativo alla dimensione del riquadro di delimitazione del testo può essere utilizzato per evitare testo troppo piccolo sullo sfondo che può risultare un disturbo o irrilevante. Il filtro di affidabilità delle parole ti permette di rimuovere i risultati inaffidabili perché sfocati o sbavati.

Per informazioni sui valori dei filtri, consulta `DetectTextFilters`.

Puoi utilizzare i filtri seguenti:

- MinConfidence—Imposta il livello di confidenza del rilevamento delle parole. Le parole con sicurezza di rilevamento al di sotto di questo livello sono escluse dal risultato. I valori dovrebbero essere compresi tra 0 e 100.
- MinBoundingBoxWidth— Imposta la larghezza minima del riquadro di delimitazione delle parole. Le parole con caselle di delimitazione inferiori a questo valore sono escluse dal risultato. Il valore è relativo alla larghezza del fotogramma dell'immagine.
- MinBoundingBoxHeight— Imposta l'altezza minima del riquadro di delimitazione delle parole. Le parole con altezze dei box di delimitazione inferiori a questo valore sono escluse dal risultato. Il valore è relativo all'altezza del fotogramma dell'immagine.
- RegionsOfInterest— Limita il rilevamento a una regione specifica della cornice dell'immagine. I valori sono relativi alle dimensioni del fotogramma. Per il testo solo parzialmente all'interno di un'area, la risposta non è definita.

DetectText risposta operativa

L'`DetectText`operazione analizza l'immagine e restituisce un array TextDetections, in cui ogni elemento (`TextDetection`) rappresenta una riga o una parola rilevata nell'immagine. Per ogni elemento, `DetectText` restituisce le informazioni riportate di seguito.

- Il testo rilevato (`DetectedText`)
- Le relazioni tra parole e righe (`Id` e `ParentId`)
- La posizione del testo nell'immagine (`Geometry`)
- L'affidabilità di Amazon Rekognition riguardo alla precisione del testo rilevato e del riquadro di delimitazione (`Confidence`)
- Il tipo di testo rilevato (`Type`)

Testo rilevato

Ogni elemento `TextDetection` contiene testo riconosciuto (parole o righe) nel campo `DetectedText`. Una parola corrisponde a uno o più caratteri non separati da spazi. `DetectText` è in grado di rilevare fino a 100 parole in un'immagine. Il testo restituito potrebbe includere caratteri che rendono irriconoscibile una parola. Ad esempio, C@t in luogo di Cat. Per determinare se un elemento `TextDetection` rappresenta una riga di testo o una parola, usa il campo `Type`.

Ogni elemento `TextDetection` include un valore percentuale che rappresenta il grado di affidabilità di Amazon Rekognition riguardo alla precisione del testo rilevato e del riquadro di delimitazione attorno al testo.

Relazioni tra parole e righe

Ogni elemento `TextDetection` ha un campo di identificazione, `Id`. L'`Id` mostra la posizione della parola in una riga. Se l'elemento è una parola, il campo di identificazione padre `ParentId` identifica la riga in cui è stata rilevata la parola. Il `ParentId` per una riga è nullo. Ad esempio, la riga "but keep" nell'immagine di esempio contiene i seguenti valori `Id` e `ParentId`

Testo	ID	ID padre
but keep	3	
but	8	3
keep	9	3

Posizione del testo in un'immagine

Per determinare dove si trova il testo riconosciuto in un'immagine, utilizzare le informazioni sul riquadro di delimitazione (Geometria) restituite da `DetectText`. L'oggetto `Geometry` contiene due tipi di informazioni sul riquadro di delimitazione per le righe e le parole rilevate:

- Un contorno rettangolare grossolano allineato all'asse in un oggetto BoundingBox
- Un poligono più dettagliato costituito da più coordinate X e Y in una matrice Point

Il riquadro di delimitazione e le coordinate del poligono mostrano dove è posizionato il testo nell'immagine di origine. I valori delle coordinate sono espressi in percentuale rispetto alla dimensione generale dell'immagine. Per ulteriori informazioni, consulta BoundingBox.

La seguente risposta JSON dell'operazione `DetectText` mostra le parole e le righe rilevate nell'immagine seguente.

```
{
 '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': 0.3957496834617615},
                                 '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'}
```

Rilevamento del testo in un video archiviato

Il rilevamento del testo di Video Amazon Rekognition nei video archiviati è un'operazione asincrona. Per iniziare a rilevare il testo, chiama. StartTextDetection Amazon Rekognition per video pubblica lo stato di completamento dell'analisi video in un argomento Amazon SNS. Se l'analisi video ha esito positivo, chiama GetTextDetection per ottenere i risultati dell'analisi. Per ulteriori informazioni su come avviare analisi video e ottenere i risultati, consultare Chiamata delle operazioni Video Amazon Rekognition.

Questa procedura espande il codice in Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK). Utilizza una coda Amazon SQS per ottenere lo stato di completamento di una richiesta di analisi video.

Per rilevare testo in un video archiviato in un bucket Amazon S3 (SDK)

1. Segui le fasi in Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK).
2. Aggiungere il codice seguente alla classe `VideoDetect` nel passaggio 1.

   Java

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


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

    NotificationChannel channel= new NotificationChannel()
```

```
            .withSNSTopicArn(snsTopicArn)
            .withRoleArn(roleArn);

    StartTextDetectionRequest req = new StartTextDetectionRequest()
            .withVideo(new Video()
                    .withS3Object(new S3Object()
                        .withBucket(bucket)
                        .withName(video)))
            .withNotificationChannel(channel);


    StartTextDetectionResult startTextDetectionResult =
 rek.startTextDetection(req);
    startJobId=startTextDetectionResult.getJobId();

}

private static void GetTextDetectionResults() throws Exception{

    int maxResults=10;
    String paginationToken=null;
    GetTextDetectionResult textDetectionResult=null;

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

        }


        textDetectionResult = rek.getTextDetection(new
 GetTextDetectionRequest()
             .withJobId(startJobId)
             .withNextToken(paginationToken)
             .withMaxResults(maxResults));

        VideoMetadata videoMetaData=textDetectionResult.getVideoMetadata();

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

```
        //Show text, confidence values
        List<TextDetectionResult> textDetections =
 textDetectionResult.getTextDetections();


        for (TextDetectionResult text: textDetections) {
            long seconds=text.getTimestamp()/1000;
            System.out.println("Sec: " + Long.toString(seconds) + " ");
            TextDetection detectedText=text.getTextDetection();

            System.out.println("Text Detected: " +
 detectedText.getDetectedText());
                System.out.println("Confidence: " +
 detectedText.getConfidence().toString());
                System.out.println("Id : " + detectedText.getId());
                System.out.println("Parent Id: " + detectedText.getParentId());
                System.out.println("Bounding Box" +
 detectedText.getGeometry().getBoundingBox().toString());
                System.out.println("Type: " + detectedText.getType());
                System.out.println();
        }
    } while (textDetectionResult !=null && textDetectionResult.getNextToken() !=
 null);


}
```

Nella funzione `main`, sostituisci le righe:

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

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

con:

```
        StartTextDetection(amzn-s3-demo-bucket, video);

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

Java V2

Questo codice è tratto dal repository degli esempi GitHub di AWS Documentation SDK. Guarda l'esempio completo qui.

```
//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
```

Nella funzione `main`, sostituisci le righe:

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

con:

```
    analyzer.StartTextDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetTextDetectionResults()
```

CLI

Esegui il AWS CLI comando seguente per iniziare a rilevare il testo in un video.

```
 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
```

Aggiorna i seguenti valori:

- Modifica `amzn-s3-demo-bucket` e `video-name` con il nome del bucket Amazon S3 e il nome del file specificati nella fase 2.
- Cambia `region-name` con la regione AWS che stai utilizzando.
- Sostituisci il valore di `profile-name` con il nome del tuo profilo di sviluppatore.
- Cambia `topic-ARN` con l'ARN dell'argomento Amazon SNS creato nella fase 3 di Configurazione di Video Amazon Rekognition.
- Modifica `role-ARN` con l'ARN del ruolo di servizio IAM creato nella fase 7 di Configurazione di Video Amazon Rekognition.

Se accedi alla CLI da un dispositivo Windows, usa le virgolette doppie anziché le virgolette singole ed evita le virgolette doppie interne tramite barra rovesciata (ovvero, \) per risolvere eventuali errori del parser che potresti riscontrare. Un esempio è fornito di seguito:

```
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
```

Dopo aver eseguito l'esempio di codice precedente, copia il `jobID` restituito e inseriscilo nel comando `GetTextDetection` di seguito per ottenere i risultati, sostituendo `job-id-number` con il `jobID` ricevuto in precedenza:

```
aws rekognition get-text-detection --job-id job-id-number --profile profile-name
```

> **Note**
>
> Se hai già eseguito un video di esempio diverso da Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK), il codice da sostituire potrebbe essere diverso.

3. Eseguire il codice. Il testo rilevato nel video viene visualizzato in un elenco.

Filtri

I filtri sono parametri di richiesta facoltativi che possono essere utilizzati quando si chiama `StartTextDetection`. La possibilità di filtrare per regione del testo, dimensione e punteggio di affidabilità offre la flessibilità necessaria a controllare l'output del rilevamento di testo. Usando le regioni di interesse è possibile limitare facilmente il rilevamento del testo alle regioni desiderate, per esempio, la terza regione dal basso per le grafiche o la regione nell'angolo in alto a sinistra per leggere i tabelloni dei punteggi in una partita di calcio. Il filtro relativo alla dimensione del riquadro di delimitazione del testo può essere utilizzato per evitare testo troppo piccolo sullo sfondo che può risultare un disturbo o irrilevante. Infine, il filtro di affidabilità delle parole ti permette di rimuovere i risultati inaffidabili perché sfocati o sbavati.

Per informazioni sui valori dei filtri, consulta `DetectTextFilters`.

Puoi utilizzare i filtri seguenti:

- MinConfidence—Imposta il livello di confidenza del rilevamento delle parole. Le parole con sicurezza di rilevamento al di sotto di questo livello sono escluse dal risultato. I valori dovrebbero essere compresi tra 0 e 100.
- MinBoundingBoxWidth— Imposta la larghezza minima del riquadro di delimitazione delle parole. Le parole con caselle di delimitazione inferiori a questo valore sono escluse dal risultato. Il valore è relativo alla larghezza del fotogramma video.
- MinBoundingBoxHeight— Imposta l'altezza minima del riquadro di delimitazione delle parole. Le parole con altezze dei box di delimitazione inferiori a questo valore sono escluse dal risultato. Il valore è relativo all'altezza del fotogramma video.
- RegionsOfInterest— Limita il rilevamento a una regione specifica del riquadro. I valori sono relativi alle dimensioni del fotogramma. Per gli oggetti solo parzialmente all'interno delle regioni, la risposta non è definita.

GetTextDetection risposta

`GetTextDetection` restituisce una matrice (`TextDetectionResults`) che contiene informazioni sul testo rilevato nel video. Esiste un elemento della matrice, TextDetection, per ogni rilevamento di parola o riga nel video. Gli elementi della matrice sono ordinati in base al tempo, espresso in millisecondi, dall'inizio del video.

Di seguito è riportato un esempio di risposta JSON parziale di `GetTextDetection`. Nella risposta, tenere presente quanto segue:

- Informazioni di testo: l'elemento dell'`TextDetectionResult`array contiene informazioni sul testo rilevato (TextDetection) e sull'ora in cui il testo è stato rilevato nel video (`Timestamp`).
- Informazioni di paginazione: L'esempio illustra una pagina di informazioni di rilevamento del testo. Puoi specificare il numero di elementi della persona da restituire nel parametro di input `MaxResults` per `GetTextDetection`. Se esistono più risultati rispetto a `MaxResults` o ci sono più risultati rispetto al massimo predefinito, `GetTextDetection` restituisce un token (`NextToken`) utilizzato per ottenere la pagina successiva dei risultati. Per ulteriori informazioni, consulta Ottenere i risultati dell'analisi di Video Amazon Rekognition.
- Informazioni video - La risposta include informazioni sul formato video (`VideoMetadata`) in ogni pagina di informazioni restituita da `GetTextDetection`.


```
{
    "JobStatus": "SUCCEEDED",
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 174441,
        "Format": "QuickTime / MOV",
        "FrameRate": 29.970029830932617,
        "FrameHeight": 480,
        "FrameWidth": 854
    },
    "TextDetections": [
        {
            "Timestamp": 967,
            "TextDetection": {
                "DetectedText": "Twinkle Twinkle Little Star",
                "Type": "LINE",
                "Id": 0,
                "Confidence": 99.91780090332031,
                "Geometry": {
                    "BoundingBox": {
                        "Width": 0.8337579369544983,
                        "Height": 0.08365312218666077,
                        "Left": 0.08313830941915512,
                        "Top": 0.4663468301296234
                    },
                    "Polygon": [
                        {
                            "X": 0.08313830941915512,
```

```
                        "Y": 0.4663468301296234
                    },
                    {
                        "X": 0.9168962240219116,
                        "Y": 0.4674469828605652
                    },
                    {
                        "X": 0.916861355304718,
                        "Y": 0.5511001348495483
                    },
                    {
                        "X": 0.08310343325138092,
                        "Y": 0.5499999523162842
                    }
                ]
            }
        }
    },
    {
        "Timestamp": 967,
        "TextDetection": {
            "DetectedText": "Twinkle",
            "Type": "WORD",
            "Id": 1,
            "ParentId": 0,
            "Confidence": 99.98338317871094,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.2423887550830841,
                    "Height": 0.0833333358168602,
                    "Left": 0.08313817530870438,
                    "Top": 0.46666666865348816
                },
                "Polygon": [
                    {
                        "X": 0.08313817530870438,
                        "Y": 0.46666666865348816
                    },
                    {
                        "X": 0.3255269229412079,
                        "Y": 0.46666666865348816
                    },
                    {
                        "X": 0.3255269229412079,
```

```
                        "Y": 0.550000011920929
                    },
                    {
                        "X": 0.08313817530870438,
                        "Y": 0.550000011920929
                    }
                ]
            }
        }
    },
    {
        "Timestamp": 967,
        "TextDetection": {
            "DetectedText": "Twinkle",
            "Type": "WORD",
            "Id": 2,
            "ParentId": 0,
            "Confidence": 99.982666015625,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.2423887550830841,
                    "Height": 0.08124999701976776,
                    "Left": 0.3454332649707794,
                    "Top": 0.46875
                },
                "Polygon": [
                    {
                        "X": 0.3454332649707794,
                        "Y": 0.46875
                    },
                    {
                        "X": 0.5878220200538635,
                        "Y": 0.46875
                    },
                    {
                        "X": 0.5878220200538635,
                        "Y": 0.550000011920929
                    },
                    {
                        "X": 0.3454332649707794,
                        "Y": 0.550000011920929
                    }
                ]
            }
```

```
            }
        },
        {
            "Timestamp": 967,
            "TextDetection": {
                "DetectedText": "Little",
                "Type": "WORD",
                "Id": 3,
                "ParentId": 0,
                "Confidence": 99.8787612915039,
                "Geometry": {
                    "BoundingBox": {
                        "Width": 0.16627635061740875,
                        "Height": 0.08124999701976776,
                        "Left": 0.6053864359855652,
                        "Top": 0.46875
                    },
                    "Polygon": [
                        {
                            "X": 0.6053864359855652,
                            "Y": 0.46875
                        },
                        {
                            "X": 0.7716627717018127,
                            "Y": 0.46875
                        },
                        {
                            "X": 0.7716627717018127,
                            "Y": 0.550000011920929
                        },
                        {
                            "X": 0.6053864359855652,
                            "Y": 0.550000011920929
                        }
                    ]
                }
            }
        },
        {
            "Timestamp": 967,
            "TextDetection": {
                "DetectedText": "Star",
                "Type": "WORD",
                "Id": 4,
```

```
                "ParentId": 0,
                "Confidence": 99.82640075683594,
                "Geometry": {
                    "BoundingBox": {
                        "Width": 0.12997658550739288,
                        "Height": 0.08124999701976776,
                        "Left": 0.7868852615356445,
                        "Top": 0.46875
                    },
                    "Polygon": [
                        {
                            "X": 0.7868852615356445,
                            "Y": 0.46875
                        },
                        {
                            "X": 0.9168618321418762,
                            "Y": 0.46875
                        },
                        {
                            "X": 0.9168618321418762,
                            "Y": 0.550000011920929
                        },
                        {
                            "X": 0.7868852615356445,
                            "Y": 0.550000011920929
                        }
                    ]
                }
            }
        }
    ],
    "NextToken": "NiHpGbZFnkM/S8kLcukMni15wb05iKtquu/Mwc+Qg1LVlMjjKNOD0Z0GusSPg7TONLe
+OZ3P",
    "TextModelVersion": "3.0"
}
```

Rilevamento di segmenti video nei video archiviati

Video Amazon Rekognition fornisce un'API che identifica segmenti di video utili, come cornici nere e titoli di coda.

Gli utenti visualizzano una quantità di contenuti senza precedenti. In particolare, le piattaforme Over-The-Top (OTT) e Video-On-Demand (VOD) offrono una ricca selezione di contenuti sempre, ovunque e su qualsiasi schermo. Con la proliferazione dei volumi di contenuti, le aziende del settore multimediale devono affrontare sfide nella preparazione e nella gestione dei contenuti. Per questo è fondamentale offrire un'esperienza di visualizzazione di alta qualità e una migliore monetizzazione dei contenuti. Oggi, le aziende utilizzano grandi team di forza lavoro umana qualificata per svolgere attività come le seguenti.

- Trovare dove si trovano i titoli di coda in una parte di contenuto.
- Scelta dei punti giusti in cui inserire annunci pubblicitari, ad esempio nelle sequenze silenziose di cornici nere
- Suddividere i video in clip più piccole per una migliore indicizzazione

Questi processi manuali sono costosi, lenti e non possono essere dimensionati per tenere il passo con il volume di contenuti prodotti, concessi in licenza e recuperati quotidianamente dagli archivi.

Puoi utilizzare Amazon Rekognition Video per automatizzare le attività operative di analisi dei media utilizzando il rilevamento di segmenti video completamente gestito e progettato appositamente, basato sull'apprendimento automatico (ML). APIs Utilizzando il APIs segmento Amazon Rekognition Video, puoi analizzare facilmente grandi volumi di video e rilevare marcatori come cornici nere o modifiche alle riprese. Ricevi i timecode, i timestamp e i numeri di frame SMPTE (Society of Motion Picture and Television Engineers) per ogni rilevamento. Non è richiesta alcuna esperienza di machine learning.

Video Amazon Rekognition analizza i video archiviati in un bucket Amazon Simple Storage Service (Amazon S3). I codici temporali SMPTE restituiti sono accurati in frame: Video Amazon Rekognition fornisce il numero esatto di frame di un segmento di video rilevato e gestisce automaticamente vari formati di frame rate video. È possibile utilizzare i metadati precisi a livello di fotogramma da Video Amazon Rekognition per automatizzare completamente determinate attività o ridurre significativamente il carico di lavoro di revisione degli operatori umani qualificati, in modo che possano concentrarsi su un lavoro più creativo. Puoi eseguire attività quali la preparazione di

contenuti, l'inserimento di annunci pubblicitari e l'aggiunta di "binge-marker" ai contenuti su scala nel cloud.

Per informazioni sui prezzi, consulta Prezzi di Amazon Rekognition.

Il rilevamento dei segmenti di Video Amazon Rekognition supporta due tipi di attività di segmentazione: rilevamento Segnali d'azione tecnici e Rilevamento delle riprese.

Argomenti

- Segnali d'azione tecnici
- Rilevamento delle riprese
- Informazioni sull'API di rilevamento dei segmenti di Video Amazon Rekognition
- Utilizzo dell'API Amazon Rekognition Segment
- Esempio: rilevamento di segmenti in un video archiviato

Segnali d'azione tecnici

Un segnale d’azione identifica fotogrammi neri, barre colore, titoli di testa, titoli di coda, loghi degli studi e il contenuto principale del programma in un video.

Fotogrammi neri

I video spesso contengono fotogrammi neri vuoti privi di audio di breve durata che vengono utilizzati come segnali di azione tecnici per inserire pubblicità o per contrassegnare la fine di un segmento di programma, come una scena o i titoli di testa. Con Video Amazon Rekognition, puoi rilevare sequenze di fotogrammi neri per automatizzare l'inserimento di annunci pubblicitari, i contenuti di pacchetti per VOD e delimitare vari segmenti di programma o scene. I fotogrammi neri con audio (dissolvenze o voci fuori campo) sono considerati come contenuti e non vengono restituiti.

Crediti

Video Amazon Rekognition consente di identificare automaticamente i fotogrammi esatti in cui iniziano e terminano i titoli di coda e i titoli di testa per un film o una serie TV. Con queste informazioni, potete generare «binge marker» o istruzioni interattive per i visualizzatori, come «Next Episode» o «Skip Intro», nelle applicazioni di video on demand (VOD). È inoltre possibile rilevare il primo e l'ultimo fotogramma del contenuto del programma in un video. Video Amazon Rekognition è addestrato a gestire un'ampia varietà di stili di crediti iniziali e finali, dai semplici crediti ricorrenti ai crediti più impegnativi insieme ai contenuti.

Barre colore

Video Amazon Rekognition consente di rilevare sezioni di video con barre colore SMPTE, che sono un insieme di colori visualizzati in modelli specifici per garantire che il colore sia calibrato correttamente su monitor, programmi e telecamere. Per ulteriori informazioni sulle barre colore SMPTE, consulta Barre colore SMPTE. Questi metadati sono utili per preparare contenuti di applicazioni VOD rimuovendo i segmenti di barre colore o per rilevare problemi come la perdita dei segnali di trasmissione quando le barre colore compaiono continuamente come segnale predefinito al posto dei contenuti.

Liste

Le liste sono sezioni del video, in genere vicine all'inizio, che contengono metadati di testo relativi all'episodio, allo studio, al formato video, ai canali audio e altro ancora. Video Amazon Rekognition è in grado di identificare l'inizio e la fine delle liste, semplificando l'utilizzo dei metadati di testo o la rimozione della lavagna durante la preparazione dei contenuti per la visualizzazione finale.

Loghi Studio

I loghi degli studi sono sequenze che mostrano i loghi o gli emblemi dello studio di produzione coinvolto nella realizzazione dello spettacolo. Video Amazon Rekognition è in grado di rilevare queste sequenze in modo che gli utenti possano esaminarle per identificare gli studi.

Contenuti

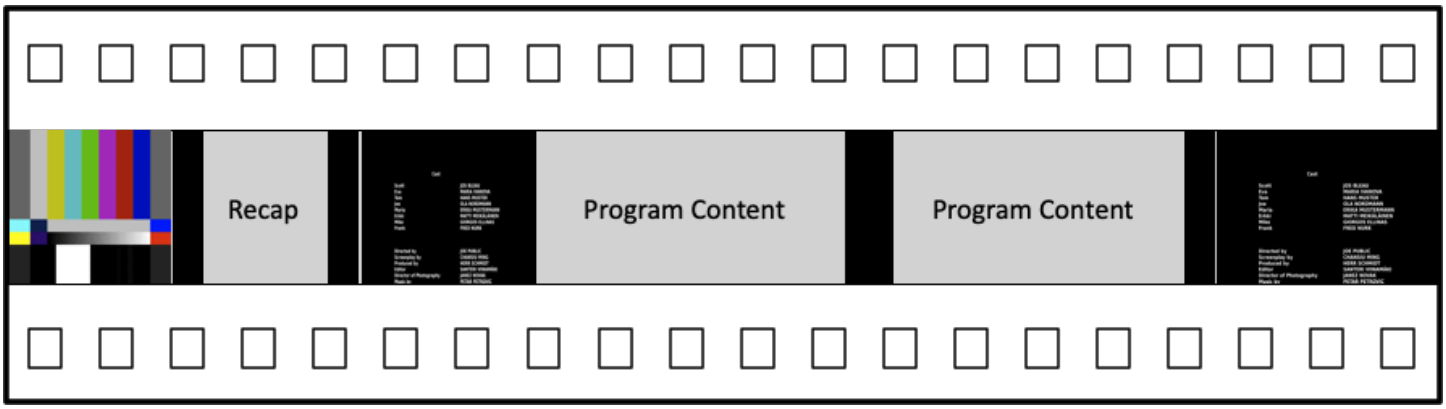
I contenuti sono le parti della serie TV o del film che contengono il programma o elementi correlati. I fotogrammi neri, i crediti, le barre di colore, le lavagne e i loghi di studio non sono considerati contenuti. Video Amazon Rekognition è in grado di rilevare l'inizio e la fine di ogni segmento di contenuto del video, in modo da poter trovare la durata del programma o segmenti specifici.

I segmenti di contenuto includono, tra l'altro, quanto segue:

- Scene di programma tra due interruzioni pubblicitarie
- Un breve riassunto dell'episodio precedente all'inizio del video
- Contenuti bonus dopo i crediti
- Contenuti «senza testo», ad esempio un insieme di tutte le scene del programma che originariamente contenevano testo sovrapposto, ma in cui il testo è stato rimosso per supportare la traduzione in altre lingue.

Dopo che Video Amazon Rekognition avrà terminato il rilevamento di tutti i segmenti di contenuto, puoi applicare le informazioni sul dominio o inviarli alla revisione umana per classificare ulteriormente ogni segmento. Ad esempio, se utilizzi video che iniziano sempre con un riepilogo, puoi classificare il primo segmento di contenuto come riepilogo.

Il diagramma seguente mostra i segmenti dei segnali d'azione tecnici della sequenza temporale di una serie o di un film. Nota le barre colorate e i titoli di testa, i segmenti di contenuto come il riepilogo e il programma principale, i riquadri neri in tutto il video e i titoli di coda.



Rilevamento delle riprese

Una ripresa è costituita da una serie di immagini consecutive collegate tra loro, riprese in modo contiguo da un'unica telecamera e che rappresentano un'azione continua nel tempo e nello spazio. Con Video Amazon Rekognition, è possibile rilevare l'inizio, la fine e la durata di ogni ripresa, nonché un conteggio per tutte le riprese in una parte di contenuto. È possibile utilizzare i metadati di ripresa per attività quali le seguenti.

- Creazione di video promozionali utilizzando riprese selezionate.
- Inserimento di annunci pubblicitari in punti che non interrompono l'esperienza dello spettatore, ad esempio nel mezzo di una ripresa quando qualcuno sta parlando.
- Generazione di una serie di miniature di anteprima che evitano contenuti di transizione tra le riprese.

Un rilevamento delle riprese è contrassegnato nel fotogramma esatto in cui è presente un hard cut su una telecamera diversa. Se c'è una transizione graduale da una telecamera all'altra, Video Amazon Rekognition omette la transizione. Ciò garantisce che l'ora di inizio e di fine ripresa non includa sezioni senza contenuto effettivo.

Il diagramma seguente illustra i segmenti di rilevamento delle riprese su una pellicola. Si noti che ogni ripresa è identificata da un taglio da un'angolazione o posizione da una telecamera a quella successiva.

Informazioni sull'API di rilevamento dei segmenti di Video Amazon Rekognition

Per segmentare un video archiviato, si utilizzano le operazioni asincrone StartSegmentDetectione GetSegmentDetectionAPI per avviare un processo di segmentazione e recuperare i risultati. Il rilevamento dei segmenti accetta i video archiviati in un bucket Amazon S3 e restituisce un output JSON. È possibile scegliere di rilevare solo segnali d'azione tecnici, solo modifiche di ripresa, o entrambe le cose insieme, configurando la richiesta API `StartSegmentdetection`. È inoltre possibile filtrare i segmenti rilevati impostando le soglie per ottenere un livello minimo di affidabilità delle previsioni. Per ulteriori informazioni, consulta Utilizzo dell'API Amazon Rekognition Segment . Per il codice di esempio, consulta Esempio: rilevamento di segmenti in un video archiviato.

Utilizzo dell'API Amazon Rekognition Segment

Il rilevamento dei segmenti di Video Amazon Rekognition nei video archiviati è un'operazione asincrona di Video Amazon Rekognition. Video Amazon Rekognition Segment API è un'API composita in cui è possibile scegliere il tipo di analisi (segnali d'azione tecnici o rilevamento delle riprese) da una singola chiamata API. Per informazioni sulla chiamata di operazioni asincrone, consulta Chiamata delle operazioni Video Amazon Rekognition.

Argomenti

- Avvio dell'analisi dei segmenti
- Recupero dei risultati dell'analisi dei segmenti

Avvio dell'analisi dei segmenti

Per avviare il rilevamento di segmenti in una video chiamata archiviata StartSegmentDetection. I parametri di input sono gli stessi delle altre operazioni Video Amazon Rekognition con l'aggiunta della selezione del tipo di segmento e del filtraggio dei risultati. Per ulteriori informazioni, consulta Avvio di analisi video.

Di seguito è riportato un esempio JSON passato da `StartSegmentDetection`. La richiesta specifica che vengono rilevati sia i segnali d'azione tecnici che i segmenti di rilevamento delle riprese. Per i segmenti dei segnali d'azione tecnici (90%) e per i segmenti di rilevamento delle riprese (80%) sono richiesti filtri diversi per la sicurezza minima di rilevamento.

```
{
  "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
      }
  }
}
```

Scelta di un tipo di segmento

Utilizzare il parametro di input dell'array `SegmentTypes` per rilevare segnali d'azione tecnici e/o segmenti di rilevamento delle riprese nel video di input.

- TECHNICAL_CUE: identifica i timestamp accurati in base ai fotogrammi per l'inizio, la fine e la durata degli indizi tecnici (cornici nere, barre colorate, titoli di apertura, titoli di coda, loghi di studio e contenuto del programma principale) rilevati in un video. Ad esempio, è possibile utilizzare i

segnali d'azione tecnici per trovare l'inizio dei titoli di coda. Per ulteriori informazioni, consulta Segnali d'azione tecnici.

- RIPRESA: identifica l'inizio, la fine e la durata di una ripresa. Ad esempio, è possibile utilizzare il rilevamento delle riprese per identificare quelle candidate per l'editing finale di un video. Per ulteriori informazioni, consulta Rilevamento delle riprese.

Filtraggio dei risultati dell'analisi

È possibile utilizzare il parametro di input `Filters` (StartSegmentDetectionFilters) per specificare la sicurezza minima di rilevamento restituita nella risposta. All'interno di `Filters`, utilizzare `ShotFilter` (StartShotDetectionFilter) per filtrare le riprese rilevate. Utilizzate `TechnicalCueFilter` (StartTechnicalCueDetectionFilter) per filtrare i suggerimenti tecnici.

Per il codice di esempio, consulta Esempio: rilevamento di segmenti in un video archiviato.

Recupero dei risultati dell'analisi dei segmenti

Amazon Rekognition per video pubblica lo stato di completamento dell'analisi video in un argomento Amazon Simple Notification Service. Se l'analisi video ha esito positivo, chiama GetSegmentDetectionper ottenere i risultati dell'analisi video.

Di seguito è riportata una richiesta `GetSegmentDetection` di esempio. `JobId` è l'identificatore del processo restituito dalla chiamata a `StartSegmentDetection`. Per informazioni sugli altri parametri di input, consulta Ottenere i risultati dell'analisi di Video Amazon Rekognition.

```
{
    "JobId": "270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1784ca90fc180e3",
    "MaxResults": 10,
    "NextToken": "XfXnZKiyMOGDhzBzYUhS5puM+g1IgezqFeYpv/H/+5noP/LmM57FitUAwSQ5D6G4AB/
PNwolrw=="
}
```

`GetSegmentDetection` restituisce i risultati per l'analisi richiesta e le informazioni generali sul video archiviato.

Informazioni generali

`GetSegmentDection` restituisce le seguenti informazioni generali.

- Informazioni audio: la risposta include metadati audio in una matrice `AudioMetadata` di AudioMetadataoggetti. Possono essere presenti più flussi audio. Ogni oggetto `AudioMetadata` contiene metadati per un singolo flusso audio. Le informazioni audio contenute in un oggetto `AudioMetadata` includono il codec audio, il numero di canali audio, la durata del flusso audio e la frequenza di campionamento. I metadati audio vengono restituiti in ogni pagina di informazioni restituite da `GetSegmentDetection`.
- Informazioni video: attualmente, Amazon Rekognition Video restituisce un VideoMetadatasingolo oggetto nell'array. `VideoMetadata` L'oggetto contiene informazioni sul flusso video nel file di input che Video Amazon Rekognition ha scelto di analizzare. L'oggetto `VideoMetadata` include il codec video, il formato video e altre informazioni. I metadati video vengono restituiti in ogni pagina di informazioni restituite da `GetSegmentDetection`.
- Informazioni di paginazione – L'esempio illustra una pagina di informazioni di segmento. Puoi specificare il numero di elementi da restituire nel parametro di input `MaxResults` per `GetSegmentDetection`. Se esiste un numero di risultati maggiore di `MaxResults`, `GetSegmentDetection` restituisce un token (`NextToken`) utilizzato per ottenere la pagina di risultati successiva. Per ulteriori informazioni, consulta Ottenere i risultati dell'analisi di Video Amazon Rekognition.
- Richiedi informazioni: il tipo di analisi richiesto nella chiamata a `StartSegmentDetection` viene riportato nel campo `SelectedSegmentTypes`.

Segmenti

I segnali d'azione tecnici e le informazioni relative alle riprese rilevati in un video vengono restituiti in un array `Segments`, di oggetti SegmentDetection. La matrice viene ordinata in base ai tipi di segmento (TECHNICAL_CUE o SHOT) specificati nel parametro di input `SegmentTypes` di `StartSegmentDetection`. All'interno di ogni tipo di segmento l'array viene ordinato in base ai valori del timestamp. Ogni oggetto `SegmentDetection` include informazioni sul tipo di segmento rilevato (segnale d'azione tecnico o rilevamento delle riprese) e informazioni generali, quali l'ora di inizio, l'ora di fine e la durata del segmento.

Le informazioni sull'ora vengono restituite in tre formati.

- Millisecondi

  Il numero di millisecondi dall'inizio del video. I campi `DurationMillis`, `StartTimestampMillis`, e `EndTimestampMillis` sono in formato millisecondi.
- Timecode

I timecode Video Amazon Rekognition sono in formato SMPTE in cui ogni fotogramma del video ha un valore di timecode univoco. Il formato è hh:mm:ss:frame. Ad esempio, un valore di timecode 01:05:40:07 verrà letto come un'ora, cinque minuti, quaranta secondi e sette fotogrammi. I casi d'uso del drop frame rate sono supportati da Video Amazon Rekognition. Il timecode del formato drop rate hh:mm:ss; frame. I campi `DurationSMPTE`, `StartTimecodeSMPTE` e `EndTimecodeSMPTE` sono in formato timecode.

- Contatori per frame

  La durata di ogni segmento video viene espressa anche con il numero di fotogrammi. Il campo `StartFrameNumber` fornisce il numero di fotogramma all'inizio di un segmento video e `EndFrameNumber` il numero di fotogramma alla fine di un segmento video. `DurationFrames`fornisce il numero totale di fotogrammi in un segmento video. Questi valori vengono calcolati utilizzando un indice di frame che inizia con 0.

È possibile utilizzare il campo `SegmentType` per determinare il tipo di segmento restituito da Video Amazon Rekognition.

- Indicazioni tecniche: il **`TechnicalCueSegment`** campo è un TechnicalCueSegmentoggetto che contiene la confidenza di rilevamento e il tipo di segnale tecnico. I tipi di segnali tecnici sono `ColorBars`, `EndCredits`, `BlackFrames`, `OpeningCredits`, `StudioLogo`, `Slate` e `Content`.
- Ripresa: il `ShotSegment` campo è un ShotSegmentoggetto che contiene la confidenza di rilevamento e un identificatore per il segmento di ripresa all'interno del video.

Di seguito è riportato un esempio di risposta JSON dell'operazione `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
        }
    ]
}
```

Per il codice di esempio, consulta Esempio: rilevamento di segmenti in un video archiviato.

Esempio: rilevamento di segmenti in un video archiviato

La procedura seguente illustra come rilevare segmenti di segnali d'azione tecnici e segmenti di rilevamento delle riprese in un video archiviato in un bucket Amazon S3. La procedura illustra anche come filtrare i segmenti rilevati in base alla sicurezza che Video Amazon Rekognition ha nell'accuratezza del rilevamento.

L'esempio si espande nel codice in Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK), che utilizza una coda Amazon Simple Queue Service per ottenere lo stato di completamento di una richiesta di analisi video.

Per rilevare segmenti in un video archiviato in un bucket Amazon S3 (SDK)

1. Eseguire Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK).
2. Aggiungere quanto segue al codice utilizzato nella fase 1.

   Java

   1. Aggiungi le seguenti importazioni:

```
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. Aggiungere il codice seguente alla classe `VideoDetect`.

```
   //Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights
 Reserved.
   //PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)


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

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

       float minTechnicalCueConfidence = 80F;
       float minShotConfidence = 80F;

       StartSegmentDetectionRequest req = new
 StartSegmentDetectionRequest()
               .withVideo(new Video()
                       .withS3Object(new S3Object()
                               .withBucket(bucket)
                               .withName(video)))
               .withSegmentTypes("TECHNICAL_CUE" , "SHOT")
               .withFilters(new StartSegmentDetectionFilters()
                       .withTechnicalCueFilter(new
 StartTechnicalCueDetectionFilter()

 .withMinSegmentConfidence(minTechnicalCueConfidence))
                       .withShotFilter(new StartShotDetectionFilter()

 .withMinSegmentConfidence(minShotConfidence)))
               .withJobTag("DetectingVideoSegments")
               .withNotificationChannel(channel);

       StartSegmentDetectionResult startLabelDetectionResult =
 rek.startSegmentDetection(req);
       startJobId=startLabelDetectionResult.getJobId();

   }

   private static void GetSegmentDetectionResults() throws Exception{

       int maxResults=10;
```

```
        String paginationToken=null;
        GetSegmentDetectionResult segmentDetectionResult=null;
        Boolean firstTime=true;


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

            GetSegmentDetectionRequest segmentDetectionRequest= new
 GetSegmentDetectionRequest()
                    .withJobId(startJobId)
                    .withMaxResults(maxResults)
                    .withNextToken(paginationToken);

            segmentDetectionResult =
 rek.getSegmentDetection(segmentDetectionRequest);

            if(firstTime) {
                System.out.println("\nStatus\n------");
                System.out.println(segmentDetectionResult.getJobStatus());
                System.out.println("\nRequested features
\n------------------");
                 for (SegmentTypeInfo requestedFeatures :
 segmentDetectionResult.getSelectedSegmentTypes()) {
                    System.out.println(requestedFeatures.getType());
                }
                 int count=1;
                 List<VideoMetadata> videoMetaDataList =
 segmentDetectionResult.getVideoMetadata();
                 System.out.println("\nVideo Streams\n-------------");
                 for (VideoMetadata videoMetaData: videoMetaDataList) {
                     System.out.println("Stream: " + count++);
                     System.out.println("\tFormat: " +
 videoMetaData.getFormat());
                     System.out.println("\tCodec: " +
 videoMetaData.getCodec());
                     System.out.println("\tDuration: " +
 videoMetaData.getDurationMillis());
                     System.out.println("\tFrameRate: " +
 videoMetaData.getFrameRate());
                 }
```

```
                List<AudioMetadata> audioMetaDataList =
segmentDetectionResult.getAudioMetadata();
                System.out.println("\nAudio streams\n-------------");

                count=1;
                for (AudioMetadata audioMetaData: audioMetaDataList) {
                    System.out.println("Stream: " + count++);
                    System.out.println("\tSample Rate: " +
audioMetaData.getSampleRate());
                    System.out.println("\tCodec: " +
audioMetaData.getCodec());
                    System.out.println("\tDuration: " +
audioMetaData.getDurationMillis());
                    System.out.println("\tNumber of Channels: " +
audioMetaData.getNumberOfChannels());
                }
                System.out.println("\nSegments\n--------");

               firstTime=false;
           }


           //Show segment information

           List<SegmentDetection> detectedSegments=
segmentDetectionResult.getSegments();

           for (SegmentDetection detectedSegment: detectedSegments) {

              if
(detectedSegment.getType().contains(SegmentType.TECHNICAL_CUE.toString()))
{
                  System.out.println("Technical Cue");
                  TechnicalCueSegment
segmentCue=detectedSegment.getTechnicalCueSegment();
                  System.out.println("\tType: " + segmentCue.getType());
                  System.out.println("\tConfidence: " +
segmentCue.getConfidence().toString());
               }
              if
(detectedSegment.getType().contains(SegmentType.SHOT.toString())) {
                  System.out.println("Shot");
```

```
                    ShotSegment
segmentShot=detectedSegment.getShotSegment();
                    System.out.println("\tIndex " +
segmentShot.getIndex());
                    System.out.println("\tConfidence: " +
segmentShot.getConfidence().toString());
                }
                long seconds=detectedSegment.getDurationMillis();
                System.out.println("\tDuration : " + Long.toString(seconds)
+ " milliseconds");
                System.out.println("\tStart time code: " +
detectedSegment.getStartTimecodeSMPTE());
                System.out.println("\tEnd time code: " +
detectedSegment.getEndTimecodeSMPTE());
                System.out.println("\tDuration time code: " +
detectedSegment.getDurationSMPTE());
                System.out.println();

            }

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

   }
```

3. Nella funzione `main`, sostituisci le righe:

```
       StartLabelDetection(amzn-s3-demo-bucket, video);

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

con:

```
       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. Aggiungere il seguente codice alla classe `VideoDetect` creata nella fase 1.

```
# Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

    def StartSegmentDetection(self):

        min_Technical_Cue_Confidence = 80.0
        min_Shot_Confidence = 80.0
        max_pixel_threshold = 0.1
        min_coverage_percentage = 60

        response = self.rek.start_segment_detection(
            Video={"S3Object": {"Bucket": self.bucket, "Name": self.video}},
            NotificationChannel={
                "RoleArn": self.roleArn,
                "SNSTopicArn": self.snsTopicArn,
            },
            SegmentTypes=["TECHNICAL_CUE", "SHOT"],
            Filters={
                "TechnicalCueFilter": {
                    "BlackFrame": {
                        "MaxPixelThreshold": max_pixel_threshold,
                        "MinCoveragePercentage": min_coverage_percentage,
                    },
                    "MinSegmentConfidence": min_Technical_Cue_Confidence,
                },
                "ShotFilter": {"MinSegmentConfidence": min_Shot_Confidence},
            }
        )

        self.startJobId = response["JobId"]
        print(f"Start Job Id: {self.startJobId}")
```

```
   def GetSegmentDetectionResults(self):
       maxResults = 10
       paginationToken = ""
       finished = False
       firstTime = True

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

           if firstTime == True:
               print(f"Status\n------\n{response['JobStatus']}")
               print("\nRequested Types\n---------------")
               for selectedSegmentType in response['SelectedSegmentTypes']:
                   print(f"\tType: {selectedSegmentType['Type']}")
                   print(f"\t\tModel Version:
{selectedSegmentType['ModelVersion']}")

               print()
               print("\nAudio metadata\n--------------")
               for audioMetadata in response['AudioMetadata']:
                   print(f"\tCodec: {audioMetadata['Codec']}")
                   print(f"\tDuration: {audioMetadata['DurationMillis']}")
                   print(f"\tNumber of Channels:
{audioMetadata['NumberOfChannels']}")
                   print(f"\tSample rate: {audioMetadata['SampleRate']}")
               print()
               print("\nVideo metadata\n--------------")
               for videoMetadata in response["VideoMetadata"]:
                   print(f"\tCodec: {videoMetadata['Codec']}")
                   print(f"\tColor Range: {videoMetadata['ColorRange']}")
                   print(f"\tDuration: {videoMetadata['DurationMillis']}")
                   print(f"\tFormat: {videoMetadata['Format']}")
                   print(f"\tFrame rate: {videoMetadata['FrameRate']}")
                   print("\nSegments\n--------")

               firstTime = False

           for segment in response['Segments']:

               if segment["Type"] == "TECHNICAL_CUE":
```

```
                    print("Technical Cue")
                    print(f"\tConfidence: {segment['TechnicalCueSegment']
['Confidence']}")
                    print(f"\tType: {segment['TechnicalCueSegment']
['Type']}")

                if segment["Type"] == "SHOT":
                    print("Shot")
                    print(f"\tConfidence: {segment['ShotSegment']
['Confidence']}")
                    print(f"\tIndex: " + str(segment["ShotSegment"]
["Index"]))

                print(f"\tDuration (milliseconds):
 {segment['DurationMillis']}")
                print(f"\tStart Timestamp (milliseconds):
 {segment['StartTimestampMillis']}")
                print(f"\tEnd Timestamp (milliseconds):
 {segment['EndTimestampMillis']}")

                print(f"\tStart timecode: {segment['StartTimecodeSMPTE']}")
                print(f"\tEnd timecode: {segment['EndTimecodeSMPTE']}")
                print(f"\tDuration timecode: {segment['DurationSMPTE']}")

                print(f"\tStart frame number {segment['StartFrameNumber']}")
                print(f"\tEnd frame number: {segment['EndFrameNumber']}")
                print(f"\tDuration frames: {segment['DurationFrames']}")

                print()

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

2. Nella funzione `main`, sostituisci le righe:

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

con:

```
    analyzer.StartSegmentDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetSegmentDetectionResults()
```

> **Note**
>
> Se hai già eseguito un video di esempio diverso da Analisi di un video archiviato in un bucket Amazon S3 con Java o Python (SDK), il codice da sostituire potrebbe essere diverso.

3. Eseguire il codice. Vengono visualizzate informazioni sui segmenti rilevati nel video di input.

Rilevamento del riconoscimento facciale

Amazon Rekognition Face Liveness ti aiuta a verificare che un utente sottoposto a verifica facciale sia fisicamente presente davanti a una telecamera. Rileva lo spoofing rivolto a una telecamera o il tentativo di bypassare una telecamera. Gli utenti possono completare un controllo di riconoscimento facciale scattando un breve video selfie in cui seguono una serie di istruzioni volte a verificare la loro effettiva presenza.

Il riconoscimento facciale viene determinato con un calcolo probabilistico, quindi dopo il controllo viene restituito un punteggio di confidenza (compreso tra 0 e 100). Più alto è il punteggio, maggiore è la fiducia che la persona che effettua il controllo sia effettivamente presente sul posto. Face Liveness restituisce anche una cornice, chiamata immagine di riferimento, che può essere utilizzata per il confronto e la ricerca dei volti. Come con qualsiasi sistema basato sulla probabilità, Face Liveness non può garantire risultati perfetti. Usalo insieme ad altri fattori per prendere una decisione basata sul rischio sull'identità personale degli utenti.

Face Liveness utilizza più componenti:

- AWS Amplify SDK (React, Swift (iOS) e Android) con componente FaceLivenessDetector
- AWS SDKs
- AWS Cloud APIs

Quando configuri l'applicazione per l'integrazione con la funzionalità Face Liveness, utilizza le seguenti operazioni API:

- CreateFaceLivenessSession- Avvia una sessione di Face Liveness, permettendo di utilizzare il modello di rilevamento Face Liveness nell'applicazione. Restituisce un SessionId per la sessione creata. Consente inoltre di impostare il proprio ChallengePrefrence, in modo da poter utilizzare l' FaceMovementChallenge opzione.
- StartFaceLivenessSession- Chiamato dall' AWS Amplify. FaceLivenessDetector Avvia un flusso di eventi contenente informazioni sugli eventi e gli attributi rilevanti nella sessione corrente.
- GetFaceLivenessSessionResults- Recupera i risultati di una sessione specifica di Face Liveness, tra cui un punteggio di confidenza di Face Liveness, un'immagine di riferimento e immagini di audit.

Utilizzerai l'SDK AWS Amplify per integrare la funzionalità Face Liveness con i flussi di lavoro di verifica facciale per le applicazioni web. Quando gli utenti effettuano l'accesso o si autenticano

tramite la tua applicazione, inviali al flusso di lavoro Face Liveness check nell'SDK Amplify. L'Amplify SDK gestisce l'interfaccia utente e il feedback in tempo reale per gli utenti mentre scattano il loro video selfie.

Quando si utilizza FaceMovementAndLightChallenge il movimento del viso dell'utente verso l'ovale visualizzato sul dispositivo, Amplify SDK visualizza una sequenza di luci colorate sullo schermo. Quindi trasmette in modo sicuro il video del selfie sul cloud. APIs In alternativa, durante l'uso FaceMovementChallenge, il viso dell'utente si sposta nell'ovale visualizzato sul dispositivo, ma non c'è una sequenza di luci colorate. Sebbene 'FaceMovementAndLightChallenge' rimanga l'impostazione migliore per massimizzare la precisione, 'FaceMovementChallenge' consente ai clienti di dare priorità ai controlli più rapidi di vivacità rispetto alla precisione. Nella scelta tra queste impostazioni, i clienti devono considerare i requisiti del proprio caso d'uso, tra cui i tipi di attacco previsti, le percentuali di falsa accettazione e di falsi rifiuti desiderate e controlli aggiuntivi come la geolocalizzazione (ad esempio in base all'IP), i codici One Time Pass (), ecc. OTPs I clienti devono prendere questa decisione dopo aver testato le prestazioni di Liveness con varie soglie di punteggio di affidabilità sui loro contenuti specifici. Inoltre, con entrambi i tipi di liveness check, i proprietari delle applicazioni devono implementare controlli per proteggere il dispositivo da cui viene inviato lo streaming video. Una volta completata l'analisi, riceverai quanto segue sul backend:

- Un punteggio di confidenza di riconoscimento facciale (compreso tra 0 e 100)
- Un'immagine di alta qualità chiamata immagine di riferimento che può essere utilizzata per Face Match o Face Search
- Un set composto da un massimo di quattro immagini, denominate immagini di controllo, selezionate dal video del selfie

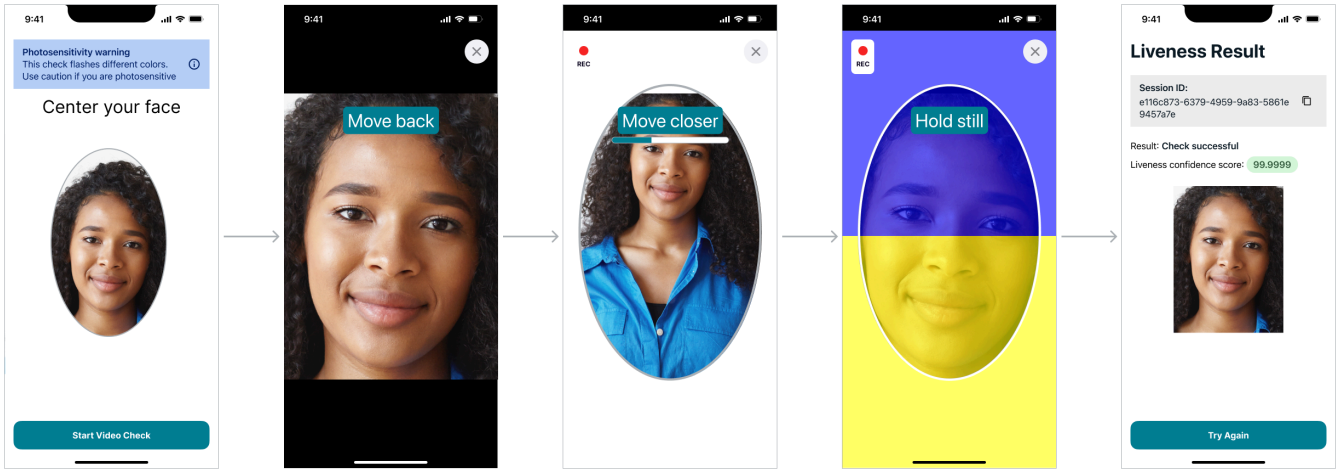
Face Liveness può essere utilizzato per una varietà di casi d'uso. Ad esempio, Face Liveness può essere utilizzato insieme al face matching (with CompareFacesand SearchFacesByImage) per la verifica dell'identità, per la stima dell'età su piattaforme con restrizioni di accesso basate sull'età e per rilevare utenti umani reali scoraggiando i bot.

Puoi saperne di più sui casi d'uso a cui è destinato il servizio, su come il machine learning (ML) viene utilizzato dal servizio e sulle considerazioni chiave per la progettazione e l'uso responsabili del servizio nella scheda di servizio per l'IA di Rekognition Face Liveness.

Puoi impostare delle soglie per i punteggi di confidenza di Face Liveness e Face Match. Le soglie scelte devono riflettere il caso d'uso. Quindi invii una verifica dell'identità all'utente in base approval/

denial al punteggio superiore o inferiore alle soglie. Se negato, chiedi all'utente di riprovare o indirizzalo a un altro metodo.

L'immagine seguente mostra il flusso dell'utente, dalle istruzioni al riconoscimento fino al risultato restituito:



Requisiti di riconoscimento facciale lato utente

Amazon Rekognition Face Liveness richiede le seguenti specifiche minime:

Dispositivi:

- Il dispositivo deve avere una fotocamera frontale
- Frequenza di aggiornamento minima del display del dispositivo: 60 Hz
- Dimensioni minime del display o dello schermo: 4 pollici
- Il dispositivo non deve essere sottoposto a jailbreak o rootato

Specifiche della fotocamera:

- Fotocamera a colori: la fotocamera frontale dovrebbe essere in grado di registrare i colori.
- Nessuna fotocamera virtuale o software per fotocamere.
- Capacità minima di registrazione: 15 fotogrammi al secondo.
- Risoluzione minima di registrazione video: 480x640px.

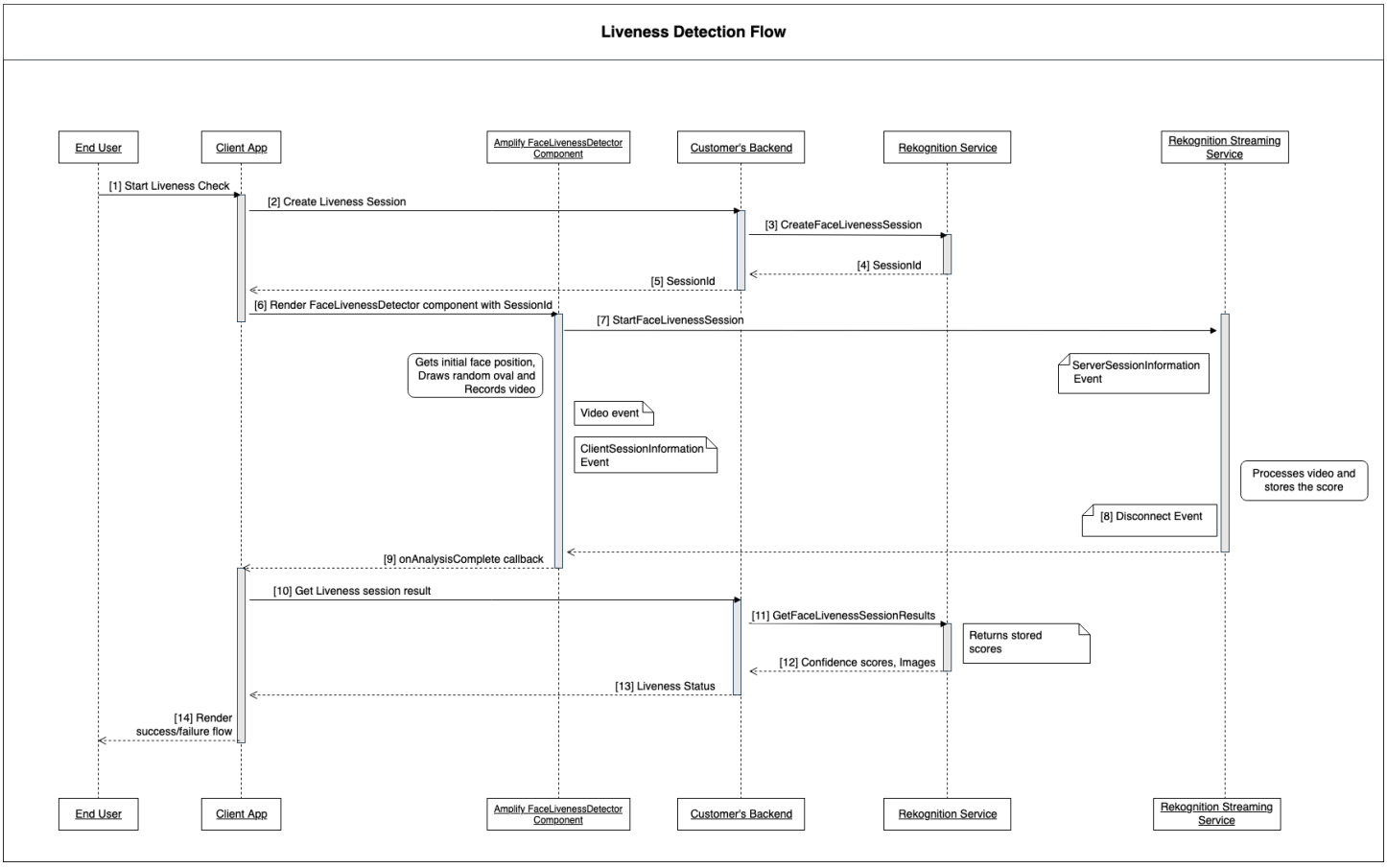
- Quando gli utenti utilizzano una webcam con un desktop per il controllo Face Liveness, è importante montare la webcam sopra lo stesso schermo in cui inizia il riconoscimento facciale.

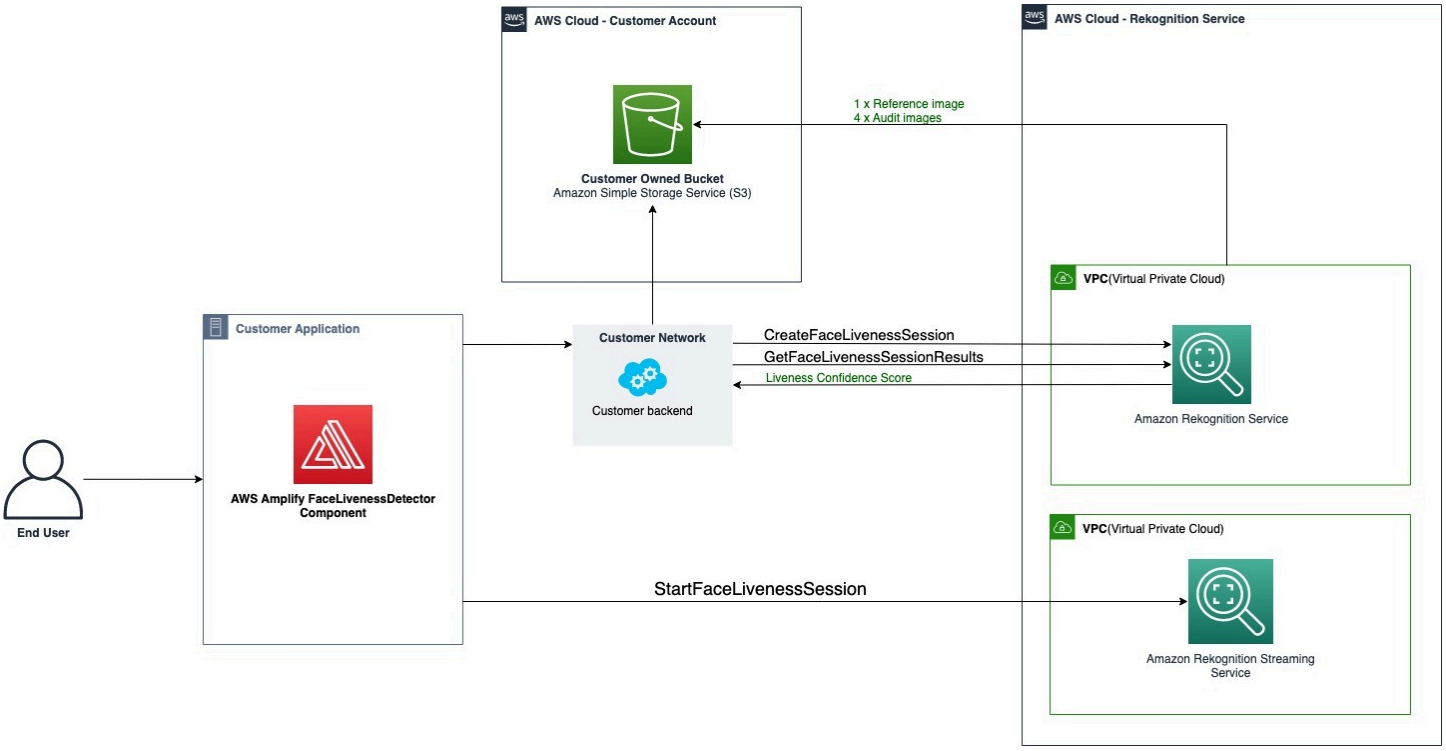
Larghezza di banda minima richiesta: 100 kbps

Browser supportati: Ultime tre versioni dei principali browser, come Google Chrome, Mozilla Firefox, Apple Safari e Microsoft Edge. Per informazioni sulle versioni dei browser Chrome e Firefox supportate, consulta Browser supportati per l'uso con la Console di gestione AWS.

Architettura e diagrammi di sequenza

I seguenti diagrammi descrivono in dettaglio il funzionamento di Amazon Rekognition Face Liveness per quanto riguarda l'architettura e la sequenza di operazioni della funzionalità:





Il processo di verifica di Face Liveness prevede diversi passaggi, come indicato di seguito:

1. L'utente avvia un controllo di riconoscimento facciale nell'app client.
2. L'app client chiama il backend del cliente, che a sua volta chiama il servizio Amazon Rekognition. Il servizio crea una sessione Face Liveness e ne restituisce una unica. SessionId Nota: dopo l'invio, una SessionId lettera scade dopo 3 minuti, quindi rimane solo una finestra di 3 minuti per completare i passaggi da 3 a 7 di seguito. È necessario utilizzare un nuovo SessionID per ogni controllo di Face Liveness. Se un determinato SessionID viene utilizzato per i successivi controlli Face Liveness, i controlli avranno esito negativo. Inoltre, un SessionId scade 3 minuti dopo l'invio, rendendo non disponibili tutti i dati Liveness associati alla sessione (ad esempio, SessionID, immagine di riferimento, immagini di controllo, ecc.).
3. L'app client esegue il rendering del componente FaceLivenessDetector Amplify utilizzando i callback ottenuti e appropriati SessionId .
4. Il FaceLivenessDetector componente stabilisce una connessione al servizio di streaming Amazon Rekognition, visualizza un ovale sullo schermo dell'utente e visualizza una sequenza di luci colorate. FaceLivenessDetector registra e trasmette video in tempo reale al servizio di streaming Amazon Rekognition.

5. Il servizio di streaming Amazon Rekognition elabora il video in tempo reale, archivia i risultati e DisconnectEvent restituisce un messaggio FaceLivenessDetector al componente quando lo streaming è completo.
6. Il FaceLivenessDetector componente richiama il `onAnalysisComplete` callback per segnalare all'app client che lo streaming è completo e che i punteggi sono pronti per il recupero.
7. L'app client chiama il backend del cliente per ottenere un flag booleano che indica se l'utente era attivo o meno. Il backend del cliente invia la richiesta al servizio Amazon Rekognition per ottenere il punteggio di fiducia, i riferimenti e le immagini di controllo. Il backend del cliente utilizza questi attributi per determinare se l'utente è attivo e restituisce una risposta appropriata all'app client.
8. Infine, l'app client trasmette la risposta al FaceLivenessDetector componente, che esegue il rendering appropriato del success/failure messaggio per completare il flusso.

Prerequisiti

I prerequisiti per l'utilizzo di Amazon Rekognition Face Liveness includono quanto segue:

1. Configura un account AWS
2. Configura Face Liveness AWS SDKs
3. Configura le risorse AWS Amplify

Fase 1: Impostazione di un account AWS

Se non hai ancora un AWS account, completa i passaggi indicati Crea un AWS account e un utente per crearne uno.

Passaggio 2: configura Face Liveness AWS SDKs

Se non l'hai già fatto, installa e configura il AWS CLI AWS SDKs. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

Esistono diversi modi per autenticare le chiamate AWS SDK. Gli esempi in questa guida presuppongono che tu stia utilizzando un profilo di credenziali predefinito per chiamare AWS i comandi CLI AWS e le operazioni dell'API SDK.

Consultate la pagina Concessione dell'accesso programmatico per ulteriori informazioni su come concedere al vostro account utente l'accesso all'SDK scelto. AWS La pagina spiega anche come utilizzare un profilo sul computer locale e come eseguire il codice di esempio negli ambienti. AWS

Assicurati che l'utente che richiede le operazioni di Face Liveness disponga delle autorizzazioni corrette per richiedere le operazioni, come le `AmazonRekognitionFullAccess` e le `AmazonS3FullAccess` autorizzazioni.

Fase 3: Configurare AWS Amplify Resources

Per integrare Amazon Rekognition Face Liveness nella tua app, devi configurare l'SDK Amplify per utilizzare AWS il componente Amplify. FaceLivenessDetector

Se non l'hai già fatto, segui le istruzioni per configurare l'Interfaccia della linea di comando AWS (AWS CLI) in Getting Started with the AWS CLI. Dopo aver installato la CLI, completa la procedura di configurazione dell'autenticazione visualizzata nel sito dei documenti dell'interfaccia utente di Amplify per configurare le risorse Amplify. AWS

Best practice del rilevamento del riconoscimento facciale

Consigliamo di seguire diverse best practice per l'utilizzo di Amazon Rekognition Face Liveness. Le best practice di riconoscimento facciale includono linee guida su dove condurre i controlli di riconoscimento facciale, l'uso di immagini di controllo e la scelta delle soglie di punteggio di confidenza.

Consulta Raccomandazioni per l'uso di Face Liveness l'elenco completo delle best practice.

Programmazione di Amazon Rekognition Face Liveness APIs

Per utilizzare l'API Amazon Rekognition Face Liveness, devi creare un backend che esegua i seguenti passaggi:

1. Chiama CreateFaceLivenessSessionper avviare una sessione di Face Liveness. Al termine dell'`CreateFaceLivenessSession`operazione, l'interfaccia utente richiede all'utente di inviare un video selfie. Il FaceLivenessDetector componente di AWS Amplify chiama quindi StartFaceLivenessSessionper eseguire il rilevamento di Liveness.
2. Chiama GetFaceLivenessSessionResultsper restituire i risultati di rilevamento associati a una sessione di Face Liveness.
3. Procedi a configurare la tua applicazione React per utilizzare il FaceLivenessDetector componente seguendo i passaggi della guida Amplify Liveness.

Prima di usare Face Liveness, assicurati di aver creato un account AWS, configurato l'interfaccia a riga di comando di AWS e AWS e SDKs AWS Amplify. Dovresti inoltre assicurarti che la policy IAM per la tua API di backend disponga di autorizzazioni che coprano quanto segue: `GetFaceLivenessSessionResults`, e `CreateFaceLivenessSession`. Per maggiori informazioni, consulta la sezione Prerequisiti.

Fase 1: CreateFaceLivenessSession

CreateFaceLivenessSession L'operazione API crea una sessione Face Liveness e ne restituisce una unica. `SessionId`

Come parte dell'input per questa operazione, è anche possibile specificare una posizione del bucket Amazon S3. Ciò consente la memorizzazione di un'immagine di riferimento e di controllare le immagini generate durante la sessione di riconoscimento facciale. Il bucket Amazon S3 deve trovarsi nell'account AWS del chiamante e nella stessa regione dell'endpoint Face Liveness. Inoltre, le chiavi degli oggetti S3 vengono generate dal sistema Face Liveness.

È anche possibile fornire un`AuditImagesLimit`, che è un numero compreso tra 0 e 4. Per impostazione predefinita, è impostato su 0. Il numero di immagini restituite è il massimo possibile e si basa sulla durata del video selfie.

Infine, puoi fornire una ChallengePreference, inclusi il tipo e la versione della sfida su cui desideri impostare la sessione come impostazione predefinita. Per impostazione predefinita, verrà utilizzato FaceLivenessMovementAndLightChallenge ma può essere impostato su FaceMovementChallenge.

Esempio di richiesta

```
{
   "ClientRequestToken": "string",
   "KmsKeyId": "string",
   "Settings": {
      "AuditImagesLimit": number,
      "ChallengePreferences": [
         {
            "Type": "string",
            "Versions": {
               "Maximum": "string",
               "Minimum": "string"
            }
```

```
         }
      ],
      "OutputConfig": {
         "S3Bucket": "string",
         "S3KeyPrefix": "string"
      }
   }
}
```

Esempio di risposta

```
{
    {"SessionId": "0f959dbb-37cc-45d8-a08d-dc42cce85fa8"}
}
```

Fase 2: StartFaceLivenessSession

Al termine dell'operazione CreateFaceLivenessSession API, il componente AWS Amplify esegue StartFaceLivenessSession l'operazione API. All'utente viene richiesto di scattare un video selfie. Per una corretta verifica, l'utente deve posizionare il volto all'interno dell'ovale sullo schermo mantenendo una buona illuminazione. Per ulteriori informazioni, consulta Raccomandazioni per l'uso di Face Liveness.

Questa operazione API richiede il video acquisito durante la sessione di Face Liveness, il sessionID ottenuto dall'operazione API e un CreateFaceLivenessSession callback. `onAnalysisComplete` Il callback può essere utilizzato per segnalare al backend di chiamare l'operazione GetFaceLivenessSessionResults API, che restituisce un punteggio di confidenza, un riferimento e immagini di controllo.

Tieni presente che questo passaggio viene eseguito dal componente AWS FaceLivenessDetector Amplify sull'applicazione client. Non è necessario eseguire configurazioni aggiuntive per chiamare`StartFaceLivenessSession`.

Fase 3: GetFaceLivenessSessionResults

L'operazione GetFaceLivenessSessionResults API recupera i risultati di una sessione specifica di Face Liveness. Richiede il SessionID come input e restituisce il punteggio di confidenza Face Liveness corrispondente. Fornisce inoltre un'immagine di riferimento che include un riquadro di

delimitazione facciale e immagini di controllo che contengono anche riquadri di delimitazione facciale. Il punteggio di confidenza di Face Liveness varia da 0 a 100.

Esempio di richiesta

```
{"SessionId": "0f959dbb-37cc-45d8-a08d-dc42cce85fa8"}
```

Esempio di risposta

```
{
    "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"
}
```

Fase 4: rispondere ai risultati

Dopo la sessione di riconoscimento facciale, confronta il punteggio di confidenza del controllo con la soglia specificata. Se il punteggio è superiore alla soglia, l'utente può passare alla schermata o all'attività successiva. Se il controllo fallisce, l'utente verrà avvisato e gli verrà richiesto di riprovare.

Chiamata Face Liveness APIs

Puoi testare Amazon Rekognition Face Liveness con qualsiasi SDK AWS supportato, come l'SDK AWS Python Boto3 o l'SDK AWS per Java. Puoi chiamare il and con l'SDK che hai scelto. `CreateFaceLivenessSession` `GetFaceLivenessSessionResults` APIs La sezione seguente mostra come chiamarli APIs con Python e Java. SDKs

Per chiamare Face Liveness: APIs

- Se non l'hai già fatto, crea o aggiorna un utente con `AmazonRekognitionFullAccess` autorizzazioni. Per ulteriori informazioni, consulta Fase 1: Configurare un account AWS e creare un utente.
- Se non l'hai già fatto, installa e configura l'AWS CLI e AWS. SDKs Per ulteriori informazioni, consulta Fase 2: Configurazione dell'interfaccia a riga di comando di AWS e di AWS. SDKs

Python

Il seguente frammento mostra come chiamarli APIs nelle applicazioni Python. Tieni presente che per eseguire questo esempio dovrai utilizzare almeno la versione 1.26.110 dell'SDK Boto3, sebbene sia consigliata la versione più recente dell'SDK.

```
import boto3

session = boto3.Session(profile_name='default')
client = session.client('rekognition')

def create_session():

    response = client.create_face_liveness_session()

    session_id = response.get("SessionId")
    print('SessionId: ' + session_id)
```

```
    return session_id


def get_session_results(session_id):

    response = client.get_face_liveness_session_results(SessionId=session_id)

    confidence = response.get("Confidence")
    status = response.get("Status")

    print('Confidence: ' + "{:.2f}".format(confidence) + "%")
    print('Status: ' + status)

    return status


def main():
    session_id = create_session()
    print('Created a Face Liveness Session with ID: ' + session_id)

    status = get_session_results(session_id)
    print('Status of Face Liveness Session: ' + status)


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

Java

Il seguente frammento mostra come chiamarli APIs nelle applicazioni Java:

```
package aws.example.rekognition.liveness;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.CreateFaceLivenessSessionRequest;
import com.amazonaws.services.rekognition.model.CreateFaceLivenessSessionResult;
import
 com.amazonaws.services.rekognition.model.GetFaceLivenessSessionResultsRequest;
```

```
import
 com.amazonaws.services.rekognition.model.GetFaceLivenessSessionResultsResult;

public class DemoLivenessApplication {

    static AmazonRekognition rekognitionClient;

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

        rekognitionClient = AmazonRekognitionClientBuilder.defaultClient();

        try {
            String sessionId = createSession();
            System.out.println("Created a Face Liveness Session with ID: " +
 sessionId);

            String status = getSessionResults(sessionId);
            System.out.println("Status of Face Liveness Session: " + status);

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

    private static String createSession() throws Exception {

        CreateFaceLivenessSessionRequest request = new
 CreateFaceLivenessSessionRequest();
        CreateFaceLivenessSessionResult result =
 rekognitionClient.createFaceLivenessSession(request);

        String sessionId = result.getSessionId();
        System.out.println("SessionId: " + sessionId);

        return sessionId;
    }

    private static String getSessionResults(String sessionId) throws Exception {

        GetFaceLivenessSessionResultsRequest request = new
 GetFaceLivenessSessionResultsRequest().withSessionId(sessionId);
        GetFaceLivenessSessionResultsResult result =
 rekognitionClient.getFaceLivenessSessionResults(request);
```

```
        Float confidence = result.getConfidence();
        String status = result.getStatus();

        System.out.println("Confidence: " + confidence);
        System.out.println("status: " + status);

        return status;
    }
}
```

Java V2

Il seguente frammento mostra come chiamare Face Liveness con l'SDK Java V2 APIs : 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

Il seguente frammento mostra come chiamare Face Liveness con l'SDK Node.Js: APIs 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)

Il seguente frammento mostra come chiamare Face APIs Liveness con l'SDK Node.Js per Javascript v3: AWS

```
import { RekognitionClient, CreateFaceLivenessSessionCommand } from "@aws-sdk/
client-rekognition"; // ES Modules
import const { RekognitionClient, CreateFaceLivenessSessionCommand } =
 require("@aws-sdk/client-rekognition"); // CommonJS import
const client = new RekognitionClient(config);
const input = {
  KmsKeyId: "STRING_VALUE",
  Settings: {
    OutputConfig: { // LivenessOutputConfig
      S3Bucket: "STRING_VALUE", // required
      S3KeyPrefix: "STRING_VALUE",
    },
    AuditImagesLimit: Number("int"),
  },
  ClientRequestToken: "STRING_VALUE",
};
const command = new CreateFaceLivenessSessionCommand(input);
const response = await client.send(command);
// { // CreateFaceLivenessSessionResponse
//   SessionId: "STRING_VALUE", // required
// };
```

Configurazione e personalizzazione dell'applicazione

Configurazione dell'applicazione

L'applicazione Face Liveness può funzionare su dispositivi mobili o browser web desktop. Ti consigliamo di configurare i componenti di Face Liveness per integrarli con la soluzione scelta. È inoltre necessario assicurarsi che l'applicazione sia autorizzata a utilizzare la fotocamera del dispositivo. La guida Amplify Liveness fornisce istruzioni dettagliate su come:

- Installare e configurare AWS Amplify
- Importa FaceLivenessDetector e renderizza il componente
- Ascoltare i callback
- Esempio di messaggio di errore di Render Amplify

Personalizzare la tua applicazione

Puoi personalizzare determinati componenti della tua applicazione liveness utilizzando AWS Amplify.

Per informazioni sulla traduzione, consulta la documentazione di Amplify Authenticator.

Per informazioni sulla personalizzazione dei componenti e dei temi di Amplify, consulta la documentazione di Amplify relativa ai temi.

Modello di responsabilità condivisa per il riconoscimento facciale

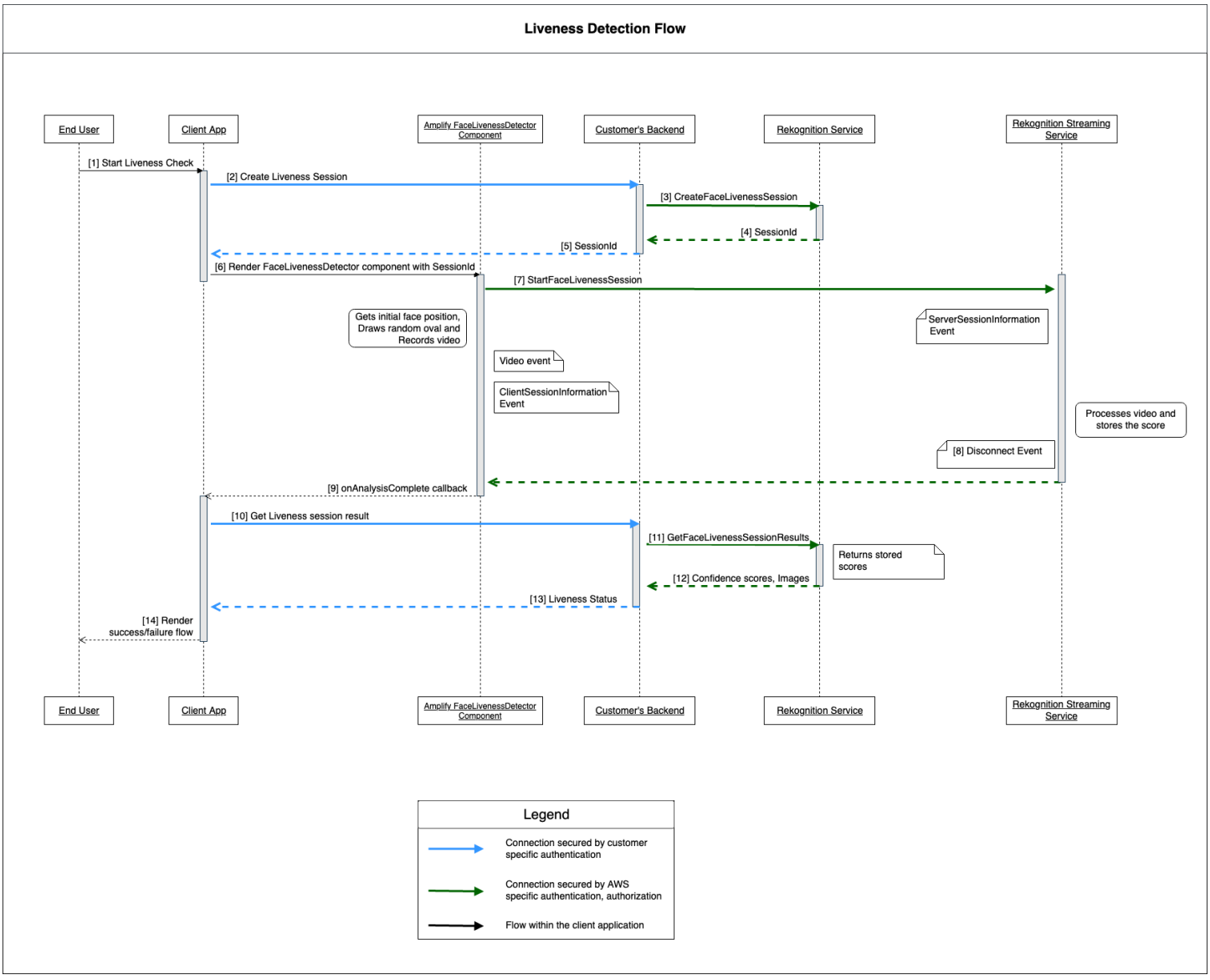
La sicurezza e la conformità sono una responsabilità condivisa tra te AWS e te, il cliente. Scopri di più sul modello di responsabilità AWS condivisa qui.

1. Tutte le chiamate al AWS servizio (tramite l'applicazione client o il backend del cliente) sono autenticate e autorizzate con AWS Auth (AWS autenticazione). È responsabilità dei proprietari dei servizi di riconoscimento facciale garantire che ciò avvenga.
2. Tutte le chiamate al backend del cliente (dall'applicazione client) vengono autenticate e autorizzate tramite il cliente. Questa responsabilità ricade sul cliente. Il cliente deve assicurarsi che le chiamate provenienti dall'applicazione client siano autenticate e non siano state manipolate in alcun modo.
3. Il backend del cliente deve identificare l'utente finale che esegue la richiesta di riconoscimento facciale. È responsabilità del cliente collegare un utente finale a una sessione di riconoscimento facciale. Il servizio di riconoscimento facciale non fa distinzione tra gli utenti finali. È in grado di identificare solo l' AWS identità chiamante (gestita dal cliente).
4. AWS consiglia ai clienti di applicare controlli di convalida aggiuntivi, come la geolocalizzazione della posizione (ad esempio in base all'IP), i codici One Time Pass (OTPs) ecc. oltre a Face Liveness, in linea con i requisiti dei casi d'uso e il livello di sicurezza.

L'impostazione 'FaceMovementAndLightChallenge' offre la massima precisione per Rekognition Liveness richiedendo agli utenti di spostare il viso verso lo schermo e di tenerlo fermo per una serie di luci lampeggianti. Consigliamo ai clienti di utilizzare questa impostazione predefinita. In alternativa, i clienti possono abilitare l'impostazione FaceMovementChallenge '', che riduce il tempo di controllo di alcuni secondi eliminando le luci lampeggianti. Sebbene «FaceMovementAndLightChallenge» rimanga l'impostazione migliore per massimizzare la precisione, «FaceMovementChallenge» consente ai clienti di dare priorità ai controlli di vivibilità più rapidi. Nella scelta tra queste impostazioni, i clienti devono considerare i requisiti del proprio caso d'uso, tra cui i tipi di attacco

previsti, le percentuali di falsa accettazione e di falsi rifiuti desiderate, e inoltre implementare controlli aggiuntivi come la geolocalizzazione (ad esempio in base all'IP), i codici One Time Pass (), ecc. OTPs I clienti devono prendere questa decisione dopo aver testato le prestazioni di Liveness con diverse soglie di punteggio di confidenza a seconda del loro caso d'uso. I clienti hanno la responsabilità di implementare i controlli per proteggere il dispositivo da cui viene inviato il video

Il seguente diagramma di flusso mostra quali chiamate sono autenticate dal servizio AWS o dal cliente:



Tutte le chiamate al servizio Amazon Rekognition Face Liveness sono AWS protette da Auth (tramite meccanismo di firma). AWS Ciò include le seguenti chiamate:

- [3] Chiamata CreateFaceLivenessSessionAPI (dal backend del cliente)

- [7] Chiamata StartFaceLivenessSessionAPI (dall'applicazione client)
- [11] Chiamata GetFaceLivenessSessionResultsAPI (dal backend del cliente)

Tutte le chiamate al backend del cliente devono disporre di un meccanismo di autenticazione e autorizzazione. I clienti devono garantire che le terze parti code/library/etc utilizzate vengano mantenute e sviluppate attivamente. I clienti devono inoltre assicurarsi che l'utente finale corretto stia effettuando chiamate alla sessione di riconoscimento facciale corretta. I clienti devono autenticare e autorizzare i seguenti flussi:

- [2] Crea una sessione Face Liveness (dall'applicazione client)
- [10] Ottieni i risultati della sessione Face Liveness (dall'applicazione client)

I clienti possono seguire il modello di sicurezza STRIDE per assicurarsi che le loro chiamate API siano protette.

Tipo	Descrizione	Controllo di sicurezza
Spoofing	Azione di minaccia finalizzata all'accesso e all'utilizzo delle credenziali di un altro utente, come nome utente e password.	Autenticazione
Manomissione	Azione di minaccia volta a modificare o modificare intenzionalmente i dati persistenti. Gli esempi includono i record in un database e l'alterazione dei dati in transito tra due computer su una rete aperta, come Internet.	Integrità
Ripudio	Azione di minaccia finalizzata all'esecuzione di operazioni proibite in un sistema che	Non ripudio

	non è in grado di tracciare le operazioni.	
Divulgazione di informazioni	Azione di minaccia intesa a leggere un file a cui non è stato concesso l'accesso o a leggere dati in transito.	La riservatezza
Negazione del servizio	Azione di minaccia consisten te nel tentativo di negare l'accesso a utenti validi, ad esempio rendendo un server Web temporaneamente non disponibile o inutilizzabile.	Disponibilità
Elevazione dei privilegi	Azione di minaccia volta a ottenere un accesso privilegi ato alle risorse per ottenere un accesso non autorizzato alle informazioni o compromettere un sistema.	Autorizzazione

AWS protegge le sue connessioni nei seguenti modi:

1. Calcolo della firma richiesta e successiva verifica della firma lato servizio. Le richieste vengono autenticate utilizzando questa firma.
2. AWS i clienti sono tenuti a configurare ruoli IAM appropriati per autorizzare determinate azioni/operazioni. Questi ruoli IAM sono necessari per effettuare chiamate ad AWS service.
3. Sono consentite solo le richieste HTTPS al servizio. AWS Le richieste vengono crittografate nella rete aperta tramite TLS. Ciò protegge la riservatezza delle richieste e ne mantiene l'integrità.
4. AWS il servizio registra dati sufficienti per identificare le chiamate effettuate dai clienti. In questo modo si evitano attacchi di ripudio .
5. AWS il servizio possiede il mantenimento di una disponibilità sufficiente

Il cliente è responsabile della protezione del servizio e delle chiamate API nei seguenti modi:

1. Il cliente deve assicurarsi di seguire un meccanismo di autenticazione adeguato. Esistono vari meccanismi di autenticazione che possono essere utilizzati per autenticare una richiesta. I clienti possono esplorare l'autenticazione basata su digest _OAuth , la connessione OpenID e altri meccanismi.
2. I clienti devono assicurarsi che il proprio servizio supporti canali di crittografia adeguati (come TLS/HTTPS) per effettuare chiamate API di servizio.
3. I clienti devono assicurarsi di registrare i dati necessari per identificare in modo univoco una chiamata API e il chiamante. Dovrebbero essere in grado di identificare il client che chiama la loro API con parametri definiti e l'ora delle chiamate.
4. I clienti devono assicurarsi che il loro sistema sia disponibile e che siano protetti dagli attacchi DDoS. Ecco alcuni esempi di tecniche di difesa contro gli attacchi DDo S.

I clienti hanno la responsabilità di conservare le proprie applicazioni up-to-date. Per ulteriori informazioni, consulta Linee guida per l'aggiornamento di Face Liveness.

Linee guida per l'aggiornamento di Face Liveness

AWS aggiorna regolarmente Face Liveness AWS SDKs (utilizzato nel backend del cliente) e FaceLivenessDetector i componenti di AWS Amplify SDKs (utilizzati nelle applicazioni client) per fornire nuove funzionalità, sicurezza aggiornata e migliorata, correzioni di bug APIs, miglioramenti dell'usabilità e altro ancora. Ti consigliamo di mantenerlo SDKs up-to-date per garantire un funzionamento ottimale della funzionalità. Se continui a utilizzare versioni precedenti di SDKs, le richieste potrebbero essere bloccate per motivi di manutenibilità e sicurezza.

Face Liveness richiede l'utilizzo del FaceLivenessDetector componente, incluso in AWS SDKs Amplify (React, iOS, Android).

Controllo delle versioni e tempistiche

Stiamo effettuando il controllo delle versioni dei seguenti componenti chiave della funzionalità Face Liveness. Seguiamo un formato di controllo delle versioni semantico. Ad esempio, un formato di versione di X.Y.Z in cui X rappresenta la versione principale, Y rappresenta la versione secondaria e Z rappresenta la versione della patch.

- Le sfide degli utenti di Face Liveness (ad esempio, FaceMovementAndLightChallenge challenge) fanno parte dell'API StartFaceLivenessSession

- FaceLivenessDetector i componenti forniti tramite AWS SDKs Amplify vengono utilizzati nelle applicazioni client

Versioni principali: riserviamo gli aggiornamenti delle versioni principali per importanti aggiornamenti di sicurezza, interruzioni delle API e straordinari aggiornamenti sull'usabilità. Le applicazioni e il backend per i clienti devono essere aggiornati il prima possibile per continuare a utilizzare le funzionalità di Face Liveness. Una volta rilasciata una nuova versione principale, supportiamo la versione principale precedente per 120 giorni dalla data della nuova versione. Potremmo bloccare le richieste provenienti dalla versione principale precedente dopo 120 giorni.

Versioni minori: riserviamo aggiornamenti delle versioni minori per importanti funzionalità e miglioramenti di sicurezza e usabilità. Consigliamo vivamente di applicare questi aggiornamenti. Sebbene ci impegniamo a garantire che gli aggiornamenti minori siano retrocompatibili il più a lungo possibile, potremmo annunciare end-of-support una versione secondaria precedente 180 giorni dopo il rilascio di una nuova versione secondaria.

Versioni delle patch: riserviamo gli aggiornamenti delle versioni patch per correzioni di bug e miglioramenti opzionali. Sebbene ti consigliamo di mantenere la tua versione up-to-date per la migliore sicurezza e esperienza utente, ci impegniamo a garantire che gli aggiornamenti delle patch siano completamente compatibili con le versioni precedenti fino al rilascio di una nuova versione principale o secondaria.

La finestra temporale per il controllo delle versioni (120 giorni per le versioni principali e 180 giorni per le minori) si applica all'aggiornamento dell'SDK dell'app, al caricamento dell'app sull'app store o sul sito Web e al download dell'ultima versione dell'app da parte degli utenti.

Rilascio delle versioni e matrice di compatibilità

Il rilascio di una versione principale per un FaceLivenessDetector componente o una sfida per l'utente spesso coincidono. Per aiutarti a tenere traccia delle dipendenze tra le versioni, consulta le risorse collegate nelle tabelle seguenti.

Versioni e changelog dell'SDK:

FaceLivenessDetector per web SDK	FaceLivenessDetector per iOS SDK	FaceLivenessDetector per Android SDK

versione attuale	Registro delle modifiche	Versione attuale/Changelog	Versione corrente/Changelog

Sfide per gli utenti:

Nome della sfida	Versione	Data di rilascio	Data di pensionamento
FaceMovementAndLightChallenge	v1.0.0	4/10/2023	N/D
FaceMovementChallenge	v1.0.0	30/04/2025	N/D

Comunicazione di nuove versioni

AWS comunica le nuove uscite attraverso i seguenti canali:

- Notifiche e-mail di aggiornamento sullo stato del servizio inviate all'indirizzo e-mail dell'account associato all'ID dell'account Face Liveness.
- Aggiornamenti pubblicati AWS SDKs e notifiche associate nei rispettivi GitHub repository.
- Aggiornamenti pubblicati per AWS SDKs Amplify e notifiche associate nei rispettivi repository. GitHub

Ti consigliamo di iscriverti a questi canali per rimanere. up-to-date

Domande frequenti su Face Liveness

Utilizza le seguenti domande frequenti per trovare le risposte alle domande più frequenti su Rekognition Face Liveness.

- Quali sono i risultati di un test di riconoscimento facciale?

  Rekognition Face Liveness fornisce i seguenti risultati per ogni riconoscimento:

  - Punteggio di fiducia: viene restituito un punteggio numerico compreso tra 0 e 100. Questo punteggio indica la probabilità che il video selfie provenga da una persona reale e non da un malintenzionato che falsifica l'identità.
  - Immagine di alta qualità: dal video selfie viene estratta un'unica immagine di alta qualità. Questa cornice può essere utilizzata per vari scopi come il confronto dei volti, la stima dell'età o la ricerca facciale.
  - Immagini di controllo: dal video selfie vengono restituite fino a quattro immagini, che possono essere utilizzate per scopi di audit trail.
- Rekognition Face Liveness è conforme ai test iBeta Presentation Attack Detection (PAD)?

  Il test Presentation Attack Detection (PAD) di iBeta Quality Assurance viene condotto in conformità alla norma ISO/IEC 30107-3. iBeta è accreditata dal NIST/NVLAP per testare e fornire risultati in base a questo standard PAD. Rekognition Face Liveness ha superato i test di conformità iBeta Presentation Attack Detection (PAD) di livello 1 e 2 con un punteggio PAD perfetto. Il rapporto può essere trovato sulla pagina web di iBeta qui.
- Come posso ottenere frame di alta qualità e frame aggiuntivi?

  Il frame di alta qualità e i frame aggiuntivi possono essere restituiti come byte non elaborati o caricati in un bucket Amazon S3 specificato, a seconda delle configurazioni della richiesta API. CreateFaceLivenessSession
- Posso cambiare la posizione dell'ovale e delle luci colorate?

  No. La posizione ovale e le luci colorate sono caratteristiche che consentono una maggiore precisione e pertanto non possono essere personalizzate.
- Posso personalizzare l'interfaccia utente secondo la nostra applicazione?

  Sì, puoi personalizzare la maggior parte dei componenti dello schermo come tema, colore, lingua, contenuto del testo e carattere per allinearli all'applicazione. I dettagli su come personalizzare questi componenti sono disponibili nella documentazione dei nostri componenti dell'interfaccia utente React, Swift e Android.
- Posso personalizzare il conto alla rovescia e il tempo per adattare un volto a un ovale?

No, il conto alla rovescia e il tempo dedicato all'allenamento facciale sono stati predeterminati sulla base di studi interni su larga scala condotti su migliaia di utenti, con l'obiettivo di fornire un equilibrio ottimale tra ciò che consente una maggiore precisione e una maggiore latenza. Per questo motivo, le impostazioni delle tempistiche non possono essere personalizzate.

- Le luci di diverso colore soddisfano le linee guida sull'accessibilità?

  Sì, le luci di colore diverso del nostro prodotto rispettano le linee guida sull'accessibilità delineate nelle WCAG 2.1. Come verificato con oltre 1000 controlli degli utenti, l'esperienza utente mostra circa due colori al secondo, il che è conforme alla raccomandazione di limitare i colori a tre al secondo. Ciò riduce la probabilità di innescare crisi epilettiche nella maggior parte della popolazione.

- L'SDK regola la luminosità dello schermo per ottenere risultati ottimali?

  Il cellulare Face Liveness SDKs (per Android e iOS) regola automaticamente la luminosità all'avvio del controllo. Tuttavia, per il web SDK esistono limitazioni sulle pagine web che impediscono la regolazione automatica della luminosità. In questi casi, ci aspettiamo che l'applicazione web indichi agli utenti finali di aumentare manualmente la luminosità dello schermo per ottenere risultati ottimali.

- Deve essere un ovale? Potremmo usare altre forme simili?

  No, le dimensioni, la forma e la posizione dell'ovale non sono personalizzabili. Lo specifico design ovale è stato scelto con cura per la sua efficacia nel catturare e analizzare accuratamente i movimenti del volto. Pertanto, la forma ovale non può essere modificata.

- Qual è la latenza end-to-end?

  Misuriamo end-to-end la latenza dal momento in cui l'utente avvia l'azione richiesta per completare il liveness check fino al momento in cui l'utente ottiene il risultato (esito positivo o negativo). Nel migliore dei casi, la latenza è di 5 secondi. In media, ci aspettiamo che sia di circa 7 secondi. Nel peggiore dei casi, la latenza è di 11 secondi. Consideriamo la variazione della end-to-end latenza in quanto dipende da: il tempo impiegato dall'utente per completare l'azione richiesta (ad esempio, sposta il viso nell'ovale), dalla connettività di rete, dalla latenza dell'applicazione, ecc.

- Posso usare la funzionalità Face Liveness senza Amplify SDK?

No, è necessario l'Amplify SDK per utilizzare la funzionalità Rekognition Face Liveness.

- Dove posso trovare gli stati di errore associati a Face Liveness?

  Puoi vedere i diversi stati di errore di Face Liveness qui.

- Face Liveness non è disponibile nella mia regione. Come posso utilizzare questa funzionalità?

  Puoi scegliere di chiamare Face Liveness in tutte le regioni in cui è disponibile, a seconda del carico di traffico e della vicinanza. Face Liveness è attualmente disponibile nelle seguenti regioni: AWS

  - Stati Uniti orientali (Virginia settentrionale)
  - US West (Oregon)
  - Europa (Irlanda)
  - Asia Pacifico (Tokyo, Mumbai)

  Anche se il tuo AWS account si trova in un'altra regione, non si prevede che la differenza di latenza sia significativa. Puoi ottenere cornici per selfie di alta qualità e controllare immagini tramite la posizione di Amazon S3 o come byte non elaborati, ma il tuo bucket Amazon S3 deve corrispondere alla regione di Face Liveness. AWS Se sono diversi, devi ricevere le immagini come byte grezzi.

- Amazon Rekognition Liveness Detection utilizza i contenuti dei clienti per migliorare il servizio?

  Puoi scegliere di non utilizzare i tuoi input di immagini e video per migliorare o sviluppare la qualità di Rekognition e di altre tecnologie di apprendimento automatico/intelligenza artificiale di Amazon utilizzando una politica di rifiuto dei servizi di AWS Organizations. Per informazioni su come rinunciare, consulta la politica di rifiuto dei servizi di Managing AI Services.

Analisi in blocco

Amazon Rekognition Bulk Analysis consente di elaborare un'ampia raccolta di immagini in modo asincrono utilizzando un file manifest con l'operazione. StartMediaAnalysisJob L'output di ogni singola immagine corrisponde all'output restituito dall'operazione utilizzata per l'analisi.

Attualmente, Rekognition supporta l'analisi con l'operazione. DetectModerationLabels

Ti verrà addebitato il numero di immagini che sono state elaborate correttamente dal processo. I risultati di un processo terminato vengono emessi in un bucket Amazon S3 specificato.

Tieni presente che l'analisi in blocco non supporta l'integrazione con Amazon A2I.

L'API è in grado di rilevare tipi di contenuto animato o illustrato e le informazioni sul tipo di contenuto rilevato vengono restituite come parte della risposta.

Argomenti

- Elaborazione di immagini in blocco
- Manifesti dei risultati dell'analisi di massa
- Tipo di contenuto
- Verifica delle previsioni e adattatori di formazione

Elaborazione di immagini in blocco

Puoi iniziare un nuovo processo di analisi di massa inviando un file manifest e richiamando l'operazione. StartMediaAnalysisJob Il file manifest di input contiene riferimenti alle immagini in un bucket Amazon S3 ed è formattato come segue:

```
{"source-ref": "s3://foo/bar/1.jpg"}
```

Per creare un processo di analisi in blocco (CLI)

1. Se non lo hai già fatto:

    a. Crea o aggiorna un utente con le autorizzazioni `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Per ulteriori informazioni, consulta Fase 1: impostazione di un account AWS e creazione di un utente.

b. Installa e configura il AWS CLI e il. AWS SDKs Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Carica immagini nel bucket S3.

   Per le istruzioni, consulta Caricamento di oggetti in Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

3. Utilizza i seguenti comandi per creare e recuperare processi di analisi in blocco.

CLI

Utilizzate il comando seguente per richiamare l'StartMediaAnalysisJoboperazione per l'analisi con l' DetectModerationLabels operazione:

```
# 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"
```

È possibile ottenere informazioni su un determinato processo, ad esempio il percorso Amazon S3 del bucket in cui sono archiviati i risultati e i file di riepilogo, utilizzando l'operazione. GetMediaAnalysisJob Lo fornisci con un ID di lavoro restituito da StartMediaAnalysisJob o. ListMediaAnalysisJob I dettagli sui singoli processi vengono conservati solo per un anno.

```
# Request
aws rekognition get-media-analysis-job \
--job-id customer-job-id
```

Puoi elencare tutte le tue analisi di massa utilizzando l'operazione di ListMediaAnalysisJobslavoro, che restituisce pagine di lavori. Con l'argomento `max-results`, è possibile specificare il numero massimo di processi da restituire per pagina, limitato al valore di `max-results`. Vengono restituiti un massimo di 100 risultati per pagina. I dettagli sui singoli processi vengono conservati solo per un anno.

```
# Request
# Specify number of jobs to return per page, limited to max-results.
aws rekognition list-media-analysis-jobs --max-results 1
```

Manifesti dei risultati dell'analisi di massa

Il processo di analisi in blocco genera un file manifesto di output che contiene i risultati del processo, nonché un riepilogo del manifest che contiene statistiche e dettagli su eventuali errori durante l'elaborazione delle voci del manifest di input.

Se nel manifest di input sono state incluse voci duplicate, il processo non tenterà di filtrare gli input univoci e elaborerà invece tutte le voci fornite.

Il file manifesto di output è formattato come segue:

```
// 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"}]}}
```

Il riepilogo del manifesto di output è formattato come segue:

```
{
    "version": "1.0",               # Schema version, 1.0 for GA.
    "statistics": {
        "total-json-lines": Number,  # Total number json lines (images) in the input
 manifest.
        "valid-json-lines": Number,  # Total number of JSON Lines (images) that contain
 references to valid images.
        "invalid-json-lines": Number # Total number of invalid JSON Lines. These lines
 were not handled.
    },
    "errors": [
        {
            "line-numer": Number,   # The number of the line in the manifest where the
 error occured.
            "source-ref": "String", # Optional. Name of the file if was parsed.
            "code": "String",       # Error code.
            "message": "String"     # Description of the error.
        }
     ]
}
```

Tipo di contenuto

Le informazioni sul tipo di contenuto multimediale analizzato per StartMediaAnalysisJob operazione vengono restituite dall'operazione. GetMediaAnalysisJob ContentType può essere una delle due diverse categorie:

- Contenuti animati, che includono videogiochi e animazioni (ad esempio cartoni animati, fumetti, manga, anime).
- Contenuti illustrati, che includono disegni, dipinti e schizzi.

Verifica delle previsioni e adattatori di formazione

L'analisi in blocco può anche essere sfruttata tramite la console Rekognition per ottenere previsioni per un batch di immagini, verificarle e quindi creare un adattatore utilizzando le previsioni verificate. Gli adattatori consentono di migliorare la precisione di qualsiasi operazione Rekognition supportata.

Attualmente, puoi creare adattatori da utilizzare con la funzionalità di moderazione personalizzata di Rekognition. Creando un adattatore e fornendolo all'DetectModerationLabelsoperazione, è possibile ottenere una maggiore precisione per le attività di moderazione dei contenuti relative al caso d'uso specifico.

Per ulteriori informazioni sulla moderazione personalizzata, consulta Migliorare la precisione con la moderazione personalizzata. Per una spiegazione su come verificare le previsioni fatte con l'analisi in blocco, consulta Analisi e verifica di massa. Per un tutorial su come utilizzare la console Rekognition per verificare le previsioni e creare un adattatore, consulta Tutorial sull'adattatore di moderazione personalizzato.

Tutorial

Questi tutorial interservizi dimostrano come utilizzare le operazioni API di Rekognition insieme ad altri AWS servizi per creare applicazioni di esempio e svolgere una serie di attività. La maggior parte di questi tutorial utilizza Amazon S3 per archiviare immagini o video. Altri servizi di uso comune includono. AWS Lambda

Argomenti

- Archiviazione dei dati di Amazon Rekognition con Amazon RDS e DynamoDB
- Utilizzo di Amazon Rekognition e Lambda per etichettare gli asset in un bucket Amazon S3
- Creazione di AWS applicazioni di analisi video
- Creazione di una funzione Lambda Amazon Rekognition
- Utilizzo di Amazon Rekognition per la verifica dell'identità
- Utilizzo di Amazon Rekognition (REK) per rilevare le etichette per le applicazioni di marketing
- Rilevamento di etichette in un'immagine usando Lambda e Python

Archiviazione dei dati di Amazon Rekognition con Amazon RDS e DynamoDB

Quando si utilizza Amazon Rekognition APIs, è importante ricordare che le operazioni API non salvano nessuna delle etichette generate. Puoi salvare queste etichette inserendole nel database, insieme agli identificatori per le rispettive immagini.

Questo tutorial dimostra il rilevamento delle etichette e il salvataggio di quelle etichette rilevate in un database. L'applicazione di esempio sviluppata in questo tutorial leggerà le immagini da un bucket Amazon S3, richiamerà l'DetectLabelsoperazione su queste immagini e memorizzerà le etichette risultanti in un database. L'applicazione memorizzerà i dati in un'istanza di database Amazon RDS o in un database DynamoDB, a seconda del tipo di database che desideri utilizzare.

Utilizzerai l'AWS SDK per Python per questo tutorial. Puoi anche vedere il GitHub repository degli esempi di AWS Documentation SDK per altri tutorial su Python.

Argomenti

- Prerequisiti
- Ottenere etichette per immagini in un bucket Amazon S3

- Creazione di una tabella Amazon DynamoDB
- Importazione di dati in DynamoDB
- Creazione di un database MySQL in Amazon RDS
- Caricamento di dati su una tabella Amazon RDS MySQL

Prerequisiti

Prima di iniziare questo tutorial, devi installare Python e completare i passaggi necessari per configurare Python SDK. AWS Oltre a ciò, assicurati di:

Aver creato un account AWS e un ruolo IAM

Aver installato l'SDK Python (Boto3)

Hai configurato correttamente le tue credenziali di accesso AWS

Aver creato il bucket Amazon S3 e riempito di immagini

Aver creato un'istanza di database RDS, se si utilizza RDS per archiviare dati

Ottenere etichette per immagini in un bucket Amazon S3

Inizia scrivendo una funzione che assuma il nome di un'immagine nel tuo bucket Amazon S3 e recupera quell'immagine. Questa immagine verrà visualizzata per confermare che le immagini corrette vengano passate a una chiamata a DetectLabelscui è presente anche la funzione.

1. Trova il bucket Amazon S3 che desideri utilizzare e scrivine il nome. Effettuerai chiamate a questo bucket Amazon S3 e leggerai le immagini al suo interno. Assicurati che il bucket contenga alcune immagini da passare all'DetectLabelsoperazione.
2. Scrivi il codice per effettuare la connessione al bucket Amazon S3. Puoi connetterti alla risorsa Amazon S3 con Boto3 per recuperare un'immagine da un bucket Amazon S3. Una volta connesso alla risorsa Amazon S3, puoi accedere al bucket fornendo al metodo Bucket il nome del bucket Amazon S3. Dopo la connessione al bucket Amazon S3, recuperi le immagini dal bucket utilizzando il metodo Object. Utilizzando Matplotlib, puoi utilizzare questa connessione per visualizzare le tue immagini mentre vengono elaborate. Boto3 viene utilizzato anche per connettersi al client Rekognition.

   Nel codice seguente, fornisci la tua regione al parametro region_name. Passerai il nome del bucket Amazon S3 e il nome dell'immagine a DetectLabels, che restituirà le etichette per

l'immagine corrispondente. Dopo aver selezionato solo le etichette dalla risposta, vengono restituiti sia il nome dell'immagine che le etichette.

```
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. Salva questo codice in un file chiamato get_images.py.

Creazione di una tabella Amazon DynamoDB

Il codice seguente utilizza Boto3 per connettersi a DynamoDB e utilizza il metodo DynamoDB `CreateTable` per creare una tabella denominata Images. La tabella ha una chiave primaria composita costituita da una chiave di partizione chiamata Image e da una chiave di ordinamento chiamata Labels. La chiave Image contiene il nome dell'immagine, mentre la chiave Labels memorizza le etichette assegnate a quell'immagine.

```
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)
```

Salva questo codice in un editor ed eseguilo una volta per creare una tabella DynamoDB.

Importazione di dati in DynamoDB

Ora che il database DynamoDB è stato creato e hai una funzione per ottenere le etichette per le immagini, puoi memorizzare le etichette in DynamoDB. Il codice seguente recupera tutte le immagini in un bucket S3, recupera le etichette per esse e archivia i dati in DynamoDB.

1. Dovrai scrivere il codice per caricare i dati su DynamoDB. Una funzione chiamata `get_image_names` viene utilizzata per connettersi al bucket Amazon S3 e restituisce i nomi di tutte le immagini nel bucket sotto forma di elenco. Passerai questo elenco alla funzione `read_image_from_S3`, che viene importata dal file `get_images.py` che hai creato.

   ```
   import boto3
   import json
   from get_images import read_image_from_s3

   boto3 = boto3.Session()

   def get_image_names(name_of_bucket):

       s3_resource = boto3.resource('s3')
       my_bucket = s3_resource.Bucket(name_of_bucket)
       file_list = []
       for file in my_bucket.objects.all():
           file_list.append(file.key)
       return file_list
   ```

2. La funzione `read_image_from_S3` che abbiamo creato in precedenza restituirà il nome dell'immagine in fase di elaborazione e il dizionario delle etichette associate a quell'immagine. Una funzione chiamata `find_values` viene utilizzata per ottenere solo le etichette dalla risposta. Il nome dell'immagine e le relative etichette sono quindi pronti per essere caricati nella tabella DynamoDB.

   ```
   def find_values(id, json_repr):
       results = []
   ```

```
    def _decode_dict(a_dict):
        try:
            results.append(a_dict[id])
        except KeyError:
            pass
        return a_dict

    json.loads(json_repr, object_hook=_decode_dict) # Return value ignored.
    return results
```

3. Utilizzerai una terza funzione, chiamata `load_data`, per caricare effettivamente le immagini e le etichette nella tabella DynamoDB che hai creato.

```
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. Qui vengono chiamate le tre funzioni che abbiamo definito in precedenza e dove vengono eseguite le operazioni. Aggiungi le tre funzioni sopra definite, insieme al codice seguente, a un file Python. Eseguire il codice.

```
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!")
```

Hai appena generato etichette per le tue immagini e le hai archiviate in un'istanza di DynamoDB. DetectLabels Assicurati di eliminare tutte le risorse che hai creato durante questo tutorial. In questo modo eviterai che ti vengano addebitate risorse che non utilizzi.

Creazione di un database MySQL in Amazon RDS

Prima di andare oltre, assicurati di aver completato la procedura di configurazione per Amazon RDS e di aver creato un'istanza DB MySQL utilizzando Amazon RDS.

Il codice seguente utilizza la libreria PyMySQL e l'istanza database Amazon RDS. Crea una tabella per contenere i nomi delle immagini e le etichette associate a tali immagini. Amazon RDS riceve comandi per creare tabelle e inserire dati nelle tabelle. Per utilizzare Amazon RDS, devi connetterti all'host Amazon RDS utilizzando il nome host, il nome utente e la password. Potrai connetterti ad Amazon RDS fornendo questi argomenti alla `connect` funzione di PyMy SQL e creando un'istanza di cursore.

1. Nel codice seguente, sostituisci il valore di host con il tuo endpoint host Amazon RDS e sostituisci il valore di user con il nome utente principale associato all'istanza Amazon RDS. Dovrai inoltre sostituire la password con la password principale dell'utente principale.

   ```
   import pymysql

   host = "host-endpoint"
   user = "username"
   password = "master-password"
   ```

2. Crea un database e una tabella in cui inserire i dati dell'immagine e dell'etichetta. Fatelo eseguendo e commettendo una query di creazione. Usa il seguente codice per creare un database. Esegui questo codice solo una volta.

   ```
   conn = pymysql.connect(host=host, user=user, passwd=password)
   print(conn)
   cursor = conn.cursor()
   print("Connection successful")

   # run once
   create_query = "create database rekogDB1"
   ```

```
print("Creation successful!")
cursor.execute(create_query)
cursor.connection.commit()
```

3. Una volta creato il database, è necessario creare una tabella in cui inserire i nomi e le etichette delle immagini. Per creare una tabella, devi prima passare il comando use SQL, insieme al nome del tuo database, alla `execute` funzione. Dopo aver effettuato la connessione, viene eseguita una query per creare una tabella. Il codice seguente si connette al database e quindi crea una tabella con una chiave primaria, chiamata`image_id`, e un attributo di testo che memorizza le etichette. Usa le importazioni e le variabili che hai definito in precedenza ed esegui questo codice per creare una tabella nel tuo database.

```
# 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!")
```

Caricamento di dati su una tabella Amazon RDS MySQL

Dopo aver creato il database Amazon RDS e una tabella nel database, puoi ottenere etichette per le tue immagini e memorizzarle nel database Amazon RDS.

1. Connettiti al tuo bucket Amazon S3 e recupera i nomi di tutte le immagini nel bucket. Questi nomi di immagini verranno passati alla `read_image_from_s3` funzione creata in precedenza per ottenere le etichette per tutte le immagini. Il codice seguente si collega al tuo bucket Amazon S3 e restituisce un elenco di tutte le immagini nel bucket.

```
import pymysql
from get_images import read_image_from_s3
import json
import boto3

host = "host-endpoint"
user = "username"
password = "master-password"

conn = pymysql.connect(host=host, user=user, passwd=password)
print(conn)
```

```
cursor = conn.cursor()
print("Connection successful")

def get_image_names(name_of_bucket):

    s3_resource = boto3.resource('s3')
    my_bucket = s3_resource.Bucket(name_of_bucket)
    file_list = []
    for file in my_bucket.objects.all():
        file_list.append(file.key)
    return file_list
```

2. La risposta dell'DetectLabelsAPI non contiene solo le etichette, quindi scrivi una funzione per estrarre solo i valori delle etichette. La seguente funzione restituisce un elenco completo delle sole etichette.

```
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. Avrai bisogno di una funzione per inserire i nomi e le etichette delle immagini nella tua tabella. La seguente funzione esegue una query di inserimento e inserisce una determinata coppia di nomi ed etichette di immagini.

```
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. Infine, è necessario eseguire le funzioni definite in precedenza. Nel codice seguente, i nomi di tutte le immagini nel bucket vengono raccolti e forniti alla funzione che chiama DetectLabels. Successivamente, le etichette e il nome dell'immagine a cui si riferiscono vengono caricate nel tuo database Amazon RDS. Copia le tre funzioni sopra definite, insieme al codice seguente, in un file Python. Esegui il file Python.

```
bucket = "bucket-name"
file_list = get_image_names(bucket)

for file in file_list:
    file_name = file
    print("Getting labels for " + file_name)
    image_name, image_labels = read_image_from_s3(bucket, file_name)
    image_json = json.dumps(image_labels, indent=4)
    labels=set(find_values("Name", image_json))
    print("Labels found: " + str(labels))
    unique_labels=set(find_values("Name", image_json))
    print(unique_labels)
    image_name_string = str(image_name)
    labels_string = str(unique_labels)
    upload_data(image_name_string, labels_string)
    print("Success!")
```

Hai generato con successo etichette per le tue immagini e le hai archiviate in un database MySQL utilizzando Amazon RDS. DetectLabels Assicurati di eliminare tutte le risorse che hai creato durante questo tutorial. Questo eviterà che ti vengano addebitate risorse che non utilizzi.

Per altri esempi AWS multiservizio, consulta l'archivio degli esempi di Documentation SDK. AWS GitHub

Utilizzo di Amazon Rekognition e Lambda per etichettare gli asset in un bucket Amazon S3

In questo tutorial, crei una AWS Lambda funzione che tagga automaticamente le risorse digitali che si trovano in un bucket Amazon S3. La funzione Lambda legge tutti gli oggetti da un determinato bucket Amazon S3. Per ogni oggetto nel bucket, passa l'immagine al servizio Amazon Rekognition per generare una serie di etichette. Ogni etichetta viene utilizzata per creare un tag da applicare

all'immagine. Dopo aver eseguito la funzione Lambda, crea automaticamente tag basati su tutte le immagini in un determinato bucket Amazon S3 e li applica alle immagini.

Ad esempio, supponiamo di eseguire la funzione Lambda e di avere questa immagine in un bucket Amazon S3.

L'applicazione crea quindi automaticamente i tag e li applica all'immagine.

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**
>
> I servizi utilizzati in questo tutorial fanno parte del piano gratuito. AWS Quando hai finito con il tutorial, ti consigliamo di eliminare tutte le risorse che hai creato durante il tutorial in modo da non ricevere alcun addebito.

Questo tutorial utilizza l' AWS SDK for Java versione 2. Consulta l' GitHub archivio degli esempi di AWS Documentation SDK per ulteriori tutorial su Java V2.

Argomenti

- Prerequisiti
- Configurazione del ruolo IAM Lambda
- Creazione del progetto
- Scrivi il codice
- Package del progetto
- Distribuire la funzione Lambda
- Test del metodo Lambda

Prerequisiti

Prima di iniziare, devi completare i passaggi descritti in Configurazione dell' AWS SDK for Java. Poi assicurati di disporre di:

- Java 1.8 JDK.
- Maven 3.6 o versione successiva.
- Un bucket Amazon S3 con 5-7 immagini della natura. Queste immagini vengono lette dalla funzione Lambda.

Configurazione del ruolo IAM Lambda

Questo tutorial utilizza i servizi Amazon Rekognition e Amazon S3. Configura il ruolo lambda-support per disporre di politiche che gli consentano di richiamare questi servizi da una funzione Lambda.

Per configurare il ruolo

1. Accedi AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione seleziona Ruoli, quindi scegli Crea ruolo.
3. Scegliere AWS Servizio, quindi scegliere Lambda.
4. Scegli la scheda Autorizzazioni.
5. Cercare AWSLambdaBasicExecutionRole.
6. Scegli Tag successivi.
7. Scegli Rivedi.
8. Assegna un nome al ruolo lambda-support.

9. Scegliere Crea ruolo.
10. Scegli lambda-support per visualizzare la pagina di panoramica.
11. Scegli Collega policy.
12. Scegli AmazonRekognitionFullAccessdall'elenco delle politiche.
13. Scegli Collega policy.
14. Cerca AmazonS3 FullAccess, quindi scegli Allega politica.

Creazione del progetto

Crea un nuovo progetto Java, quindi configura Maven pom.xml con le impostazioni e le dipendenze richieste. Assicurati che il file pom.xml sia simile al seguente:

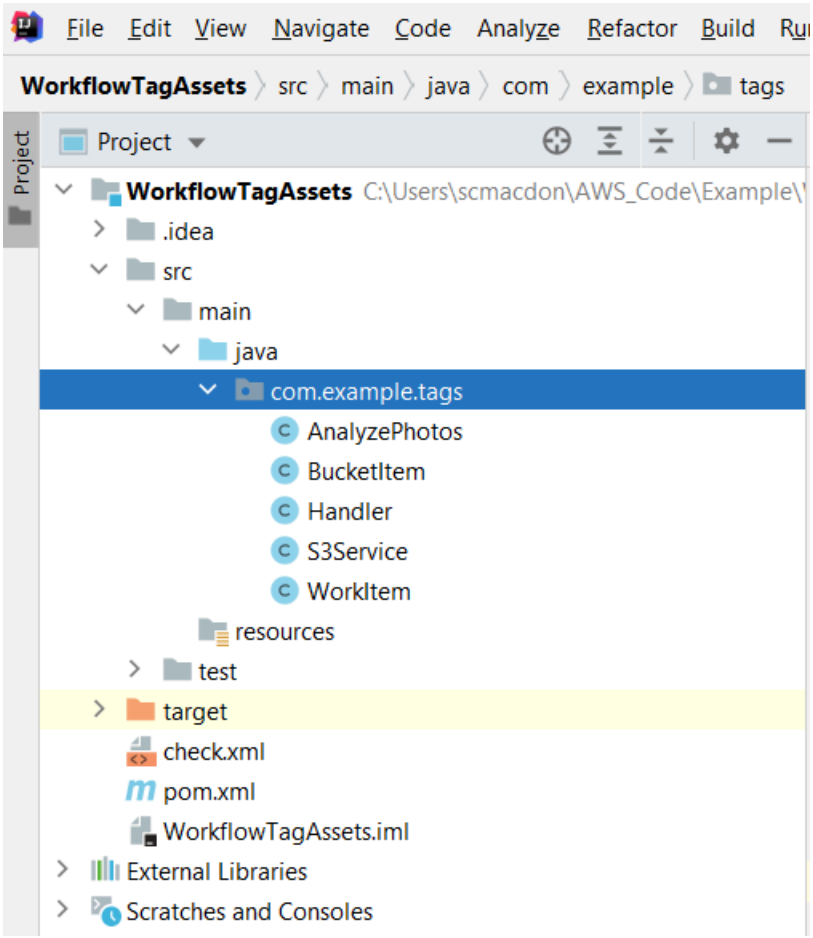
```
<?xml version="1.0" encoding="UTF-8"?>
 <project xmlns="http://maven.apache.org/POM/4.0.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>org.example</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>
```

Scrivi il codice

Utilizzate l'API Java AWS Lambda runtime per creare la classe Java che definisce la funzione Lambda. In questo esempio, esiste una classe Java per la funzione Lambda denominata Handler e classi aggiuntive richieste per questo caso d'uso. La figura seguente mostra le classi Java del progetto. Notate che tutte le classi Java si trovano in un pacchetto denominato com.example.tags.



Create le seguenti classi Java per il codice:

- Handler utilizza l'API run-time Lambda Java ed esegue lo use case descritto in questo tutorial. AWS La logica dell'applicazione che viene eseguita si trova nel metodo handleRequest.
- S3Service utilizza l'API Amazon S3 per eseguire operazioni S3.
- AnalyzePhotosutilizza l'API Amazon Rekognition per analizzare le immagini.
- BucketItemdefinisce un modello che memorizza le informazioni sui bucket Amazon S3.
- WorkItemdefinisce un modello che archivia i dati di Amazon Rekognition.

Classe gestore

Questo codice Java rappresenta la classe Gestore. La classe legge un flag passato alla funzione Lambda. Il servizio S3. ListBucketObjectsmetodo restituisce un oggetto List in cui ogni elemento è un valore di stringa che rappresenta la chiave dell'oggetto. Se il valore del flag è true, i tag vengono applicati scorrendo l'elenco e applicando i tag a ciascun oggetto chiamando il metodo s3Service.tagAssets. Se il valore del flag è false, allora S3Service. deleteTagFromViene richiamato il metodo Object che elimina i tag. Inoltre, tieni presente che puoi registrare i messaggi CloudWatch nei log di Amazon utilizzando un LambdaLoggeroggetto.

> **Note**
>
> Assicurati di assegnare il nome del bucket alla variabile bucketName.

```
package com.example.tags;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

public class Handler implements RequestHandler<Map<String,String>, String> {

@Override
public String handleRequest(Map<String, String> event, Context context) {
    LambdaLogger logger = context.getLogger();
    String delFlag = event.get("flag");
    logger.log("FLAG IS: " + delFlag);
    S3Service s3Service = new S3Service();
    AnalyzePhotos photos = new AnalyzePhotos();

    String bucketName = "<Enter your bucket name>";
    List<String> myKeys = s3Service.listBucketObjects(bucketName);
    if (delFlag.compareTo("true") == 0) {

        // Create a List to store the data.
        List<ArrayList<WorkItem>> myList = new ArrayList<>();
```

```
        // loop through each element in the List and tag the assets.
        for (String key : myKeys) {

            byte[] keyData = s3Service.getObjectBytes(bucketName, key);

            // Analyze the photo and return a list where each element is a WorkItem.
            ArrayList<WorkItem> item = photos.detectLabels(keyData, key);
            myList.add(item);
        }

        s3Service.tagAssets(myList, bucketName);
        logger.log("All Assets in the bucket are tagged!");

    } else {

        // Delete all object tags.
        for (String key : myKeys) {
            s3Service.deleteTagFromObject(bucketName, key);
            logger.log("All Assets in the bucket are deleted!");
        }
     }
    return delFlag;
 }
}
```

Classe S3Service

La classe seguente utilizza l'API Amazon S3 per eseguire operazioni S3. Ad esempio, il getObjectBytesmetodo restituisce un array di byte che rappresenta l'immagine. Allo stesso modo, il listBucketObjectsmetodo restituisce un oggetto List in cui ogni elemento è un valore di stringa che specifica il nome della chiave.

```
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 classe

Il seguente codice Java rappresenta la AnalyzePhotosclasse. Questa classe utilizza l'API Amazon Rekognition per analizzare le immagini.

```
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 classe

Il seguente codice Java rappresenta la BucketItemclasse che memorizza i dati degli oggetti Amazon S3.

```
package com.example.tags;

public class BucketItem {

 private String key;
 private String owner;
 private String date ;
 private String size ;


 public void setSize(String size) {
    this.size = size ;
 }
```

```
 public String getSize() {
    return this.size ;
 }

 public void setDate(String date) {
    this.date = date ;
 }

 public String getDate() {
    return this.date ;
 }

 public void setOwner(String owner) {
    this.owner = owner ;
 }

 public String getOwner() {
    return this.owner ;
 }

 public void setKey(String key) {
    this.key = key ;
 }

 public String getKey() {
    return this.key ;
 }
}
```

WorkItem classe

Il seguente codice Java rappresenta la WorkItemclasse.

```
 package com.example.tags;

 public class WorkItem {

 private String key;
 private String name;
 private String confidence ;

public void setKey (String key) {
```

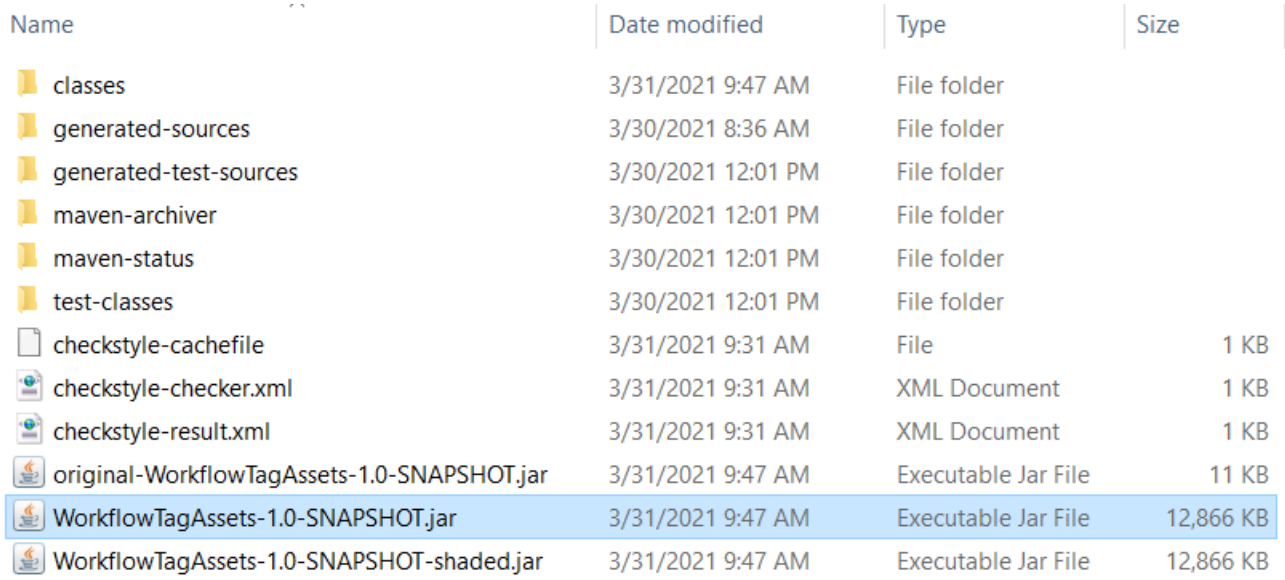
```
    this.key = key;
}

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;
}

}
```

Package del progetto

Package del progetto in un file.jar (JAR) utilizzando il seguente comando Maven.

```
mvn package
```

Il file JAR si trova nella cartella di destinazione (che è una cartella secondaria della cartella del progetto).



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**
>
> Notate l'uso di nel file POM del progetto. maven-shade-plugin Questo plugin è responsabile della creazione di un JAR che contiene le dipendenze richieste. Se tenti di impacchettare il progetto senza questo plugin, le dipendenze richieste non sono incluse nel file JAR e incontrerai un. ClassNotFoundException

Distribuire la funzione Lambda

1. Aprire la console Lambda.
2. Selezionare Create function (Crea funzione).
3. Scegli Author from scratch (Crea da zero).
4. Nella sezione Informazioni di base, inserisci cron come nome.
5. In Runtime, selezionare Java 8.
6. Scegliere Usa un ruolo esistente, quindi scegliere lambda-support (il ruolo IAM creato).
7. Scegli Crea funzione.
8. In Tipo di voce del codice, scegli Carica un file .zip o un file .jar.
9. Scegli Carica, quindi naviga fino al file JAR che hai creato.
10. Per Handler, inserite il nome completo della funzione, ad esempio com.example.tags.Handler:handleRequest (com.example.tags specifica il pacchetto, Handler è la classe seguita da :: e il nome del metodo).

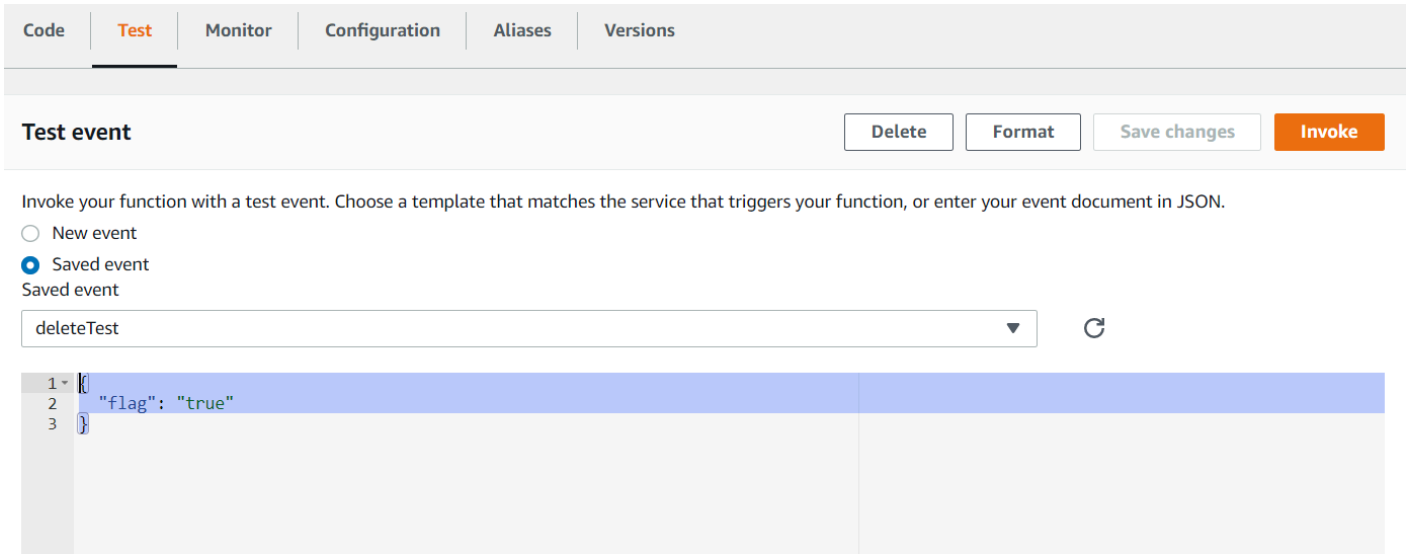
11. Seleziona Salva.

Test del metodo Lambda

A questo punto del tutorial, è possibile testare la funzione Lambda.

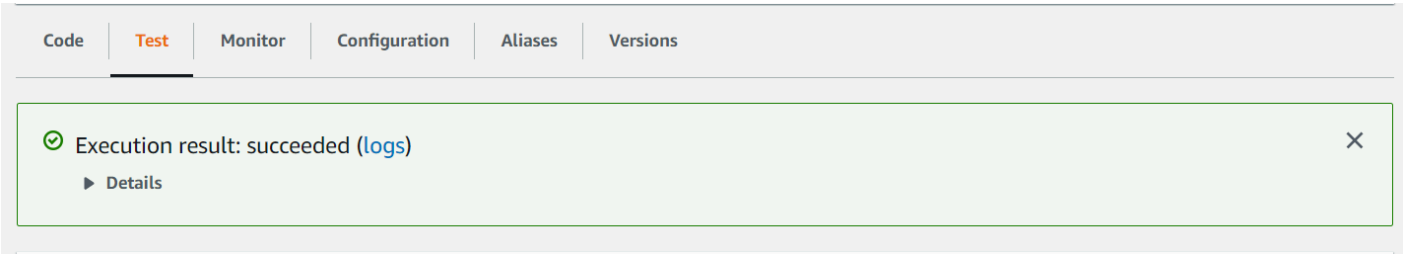
1. Nella console Lambda, fai clic sulla scheda Test, quindi inserisci il seguente codice JSON.

```
                {
"flag": "true"
 }
```



> **Note**
>
> Se si passa a true tags alle risorse digitali e si passa a false, i tag vengono eliminati.

2. Scegliete il pulsante Invoke. Dopo aver richiamato la funzione Lambda, viene visualizzato un messaggio di successo.



Congratulazioni, hai creato una AWS Lambda funzione che applica automaticamente i tag agli asset digitali che si trovano in un bucket Amazon S3. Come indicato all'inizio di questo tutorial, assicurati di terminare tutte le risorse che hai creato durante lo svolgimento di questo tutorial per assicurarti che non ti vengano addebitati costi.

Per altri esempi AWS multiservizio, consulta l'archivio degli esempi di Documentation SDK.AWS GitHub

Creazione di AWS applicazioni di analisi video

È possibile creare un'applicazione Web Java che analizza i video per il rilevamento delle etichette utilizzando l' AWS SDK for Java versione 2. L'applicazione creata in questo AWS tutorial consente di caricare un video (MP4 file) in un bucket Amazon S3. Quindi l'applicazione utilizza il servizio Amazon Rekognition per analizzare il video. I risultati vengono utilizzati per compilare un modello di dati, quindi viene generato un report che viene inviato via e-mail a un utente specifico utilizzando Amazon Simple Email Service.

L'illustrazione seguente mostra un report generato dopo che l'applicazione ha completato l'analisi del video. Le colonne della tabella seguente mostrano la fascia d'età, la barba, gli occhiali e gli occhi aperti, oltre ai valori di confidenza per le previsioni dei diversi attributi.

	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 questo tutorial, crei un'applicazione Spring Boot che richiama vari servizi. AWS Gli Spring Boot APIs vengono utilizzati per creare un modello, diverse viste e un controller. Per ulteriori informazioni, consulta Spring Boot.

Questo servizio utilizza i seguenti AWS servizi:

- Amazon Rekognition
- Amazon S3
- Amazon SES
- AWS Elastic Beanstalk

I AWS servizi inclusi in questo tutorial sono inclusi nel piano AWS gratuito. Ti consigliamo di eliminare tutte le risorse che crei nel tutorial quando hai finito di utilizzarle per evitare che ti vengano addebitate.

Prerequisiti

Prima di iniziare, devi completare i passaggi descritti in Configurazione dell' AWS SDK for Java. Poi assicurati di disporre di:

- Java 1.8 JDK.
- Maven 3.6 o versione successiva.
- Un bucket Amazon S3 denominato video [somevalue]. Assicurati di utilizzare questo nome di bucket nel codice Java di Amazon S3. Per ulteriori informazioni, consulta Creazione di un bucket.
- Un ruolo IAM. Ne hai bisogno per la VideoDetectFacesclasse che creerai. Per ulteriori informazioni, consulta Configurazione di Video Amazon Rekognition.
- Argomento Amazon SNS valido. Ne hai bisogno per la VideoDetectFacesclasse che creerai. Per ulteriori informazioni, consulta Configurazione di Video Amazon Rekognition.

Procedura

Nel corso del tutorial, esegui le seguenti operazioni:

1. Crea un progetto
2. Aggiungi le dipendenze POM al tuo progetto
3. Crea le classi Java
4. Crea i file HTML
5. Crea i file di script
6. Pacchetto del progetto in un file JAR
7. Distribuisci l'applicazione su AWS Elastic Beanstalk

Per procedere con il tutorial, segui le istruzioni dettagliate nell'archivio degli esempi GitHub di AWS Documentation SDK.

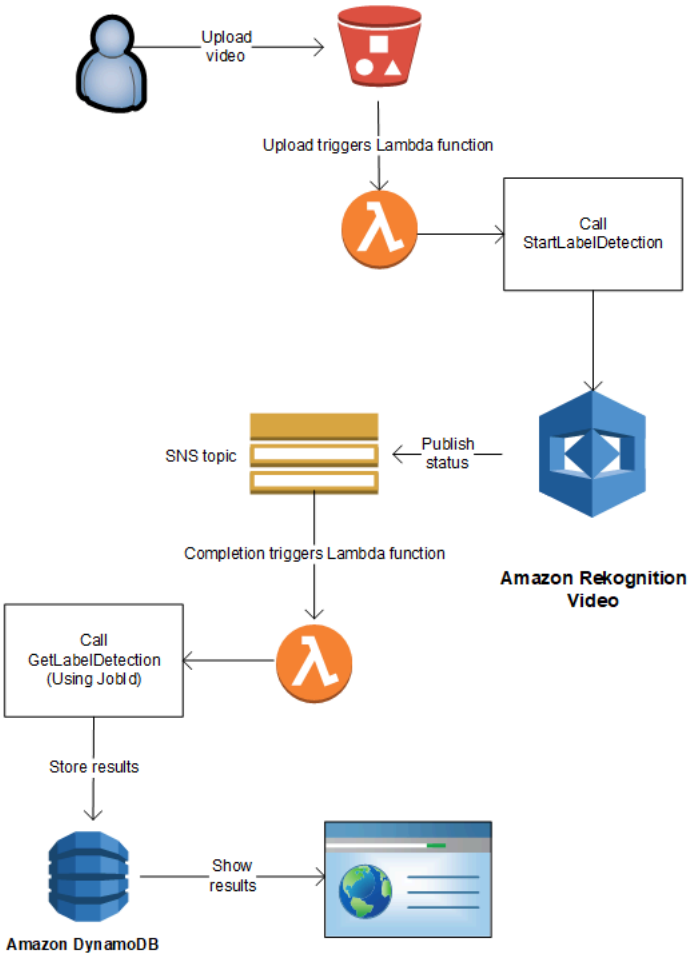
Creazione di una funzione Lambda Amazon Rekognition

Questa esercitazione mostra come ottenere i risultati di un'operazione di analisi video per il rilevamento di etichette utilizzando una funzione Lambda Java.

> **Note**
>
> Questo tutorial utilizza l' AWS SDK for Java 1.x. Per un tutorial su Rekognition e l'SDK for AWS Java versione 2, consulta l'archivio degli esempi di Documentation SDK.AWS GitHub

Puoi usare le funzioni Lambda con le operazioni di Video Amazon Rekognition. Ad esempio, il seguente diagramma mostra un sito Web che utilizza una funzione Lambda per avviare automaticamente l'analisi di un video quando viene caricato in un bucket Amazon S3. Quando la funzione Lambda viene attivata, chiama StartLabelDetectionper iniziare a rilevare le etichette nel video caricato. Per informazioni sull'utilizzo di Lambda per elaborare le notifiche degli eventi da un bucket Amazon S3, consulta Utilizzo di AWS Lambda con gli eventi Amazon S3.

Una seconda funzione Lambda viene attivata quando lo stato di completamento dell'analisi viene inviato all'argomento Amazon SNS registrato. La seconda funzione Lambda chiama GetLabelDetectionper ottenere i risultati dell'analisi. I risultati vengono quindi archiviati in un database in preparazione alla visualizzazione su una pagina Web. Questa seconda funzione lambda è l'argomento principale di questo tutorial.



In questa esercitazione, la funzione Lambda viene attivata quando Video Amazon Rekognition invia lo stato di completamento per l'analisi video all'argomento Amazon SNS registrato. Quindi raccoglie i risultati dell'analisi video chiamando GetLabelDetection. A scopo dimostrativo, questo tutorial scrive i risultati del rilevamento delle etichette in un registro. CloudWatch Nella funzione Lambda della tua applicazione, è opportuno archiviare i risultati dell'analisi per un utilizzo successivo. Ad esempio, è possibile usare Amazon DynamoDB per salvare i risultati dell'analisi. Per ulteriori informazioni, vedi Lavorare con DynamoDB.

Le procedure seguenti mostrano come:

- Creare l'argomento Amazon SNS e impostare le autorizzazioni.
- Crea la funzione Lambda utilizzando AWS Management Console e sottoscrivila all'argomento Amazon SNS.
- Configurare la funzione Lambda tramite la AWS Management Console.
- Aggiungi codice di esempio a un AWS Toolkit for Eclipse progetto e caricalo nella funzione Lambda.

- Test della funzione Lambda tramite AWS CLI.

> **Note**
>
> Usa la stessa AWS regione per tutto il tutorial.

Prerequisiti

Questa esercitazione presuppone che tu sia esperto di AWS Toolkit for Eclipse. Per ulteriori informazioni, consulta AWS Toolkit for Eclipse.

Creazione dell'argomento SNS

Lo stato di completamento di un'operazione di un'analisi video di Amazon Rekognition per video viene inviato a un argomento Amazon SNS. Questa procedura consente di creare l'argomento Amazon SNS e il ruolo di servizio IAM che offre l'accesso a Video Amazon Rekognition ai tuoi argomenti Amazon SNS. Per ulteriori informazioni, consulta Chiamata delle operazioni Video Amazon Rekognition.

Come creare un argomento Amazon SNS

1. Se non l'hai ancora fatto, crea un ruolo di servizio IAM per fornire a Video Amazon Rekognition l'accesso ai tuoi argomenti Amazon SNS. Prendi nota del nome della risorsa Amazon (ARN). Per ulteriori informazioni, consulta Accesso a più argomenti Amazon SNS.
2. Creare un argomento Amazon SNS utilizzando la console Amazon SNS. È sufficiente specificare il nome dell'argomento. Anteponi il nome dell'argomento con. AmazonRekognition Prendere nota dell'ARN dell'argomento.

Creazione della funzione Lambda

Puoi creare la funzione Lambda tramite la AWS Management Console. Quindi, puoi utilizzare un progetto AWS Toolkit for Eclipse per caricare il pacchetto della funzione Lambda su AWS Lambda. È inoltre possibile creare la funzione Lambda con AWS Toolkit for Eclipse. Per ulteriori informazioni, consulta Esercitazione: Come creare, caricare e chiamare una funzione Lambda AWS.

Creazione della funzione Lambda

1. Accedi alla Console di gestione AWS e apri la console di AWS Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
2. Selezionare Create function (Crea funzione).
3. Scegli Author from scratch (Crea da zero).
4. In Function name (Nome funzione), immettere un nome per la funzione.
5. In Runtime, selezionare Java 8.
6. Scegliere Choose or create an execution role (Scegliere o creare un ruolo di esecuzione).
7. In Execution role (Ruolo di esecuzione), scegliere Create a new role with basic Lambda permissions (Crea un nuovo ruolo con le autorizzazioni Lambda di base).
8. Annotare il nome del nuovo ruolo visualizzato nella parte inferiore della sezione Basic Information (Informazioni di base).
9. Scegli Crea funzione.

Configura la funzione Lambda

Una volta creata la funzione Lambda, la configuri per essere attivata dall'argomento Amazon SNS creato in Creazione dell'argomento SNS. Devi inoltre adeguare i requisiti di memoria e il periodo di timeout per la funzione Lambda.

Per configurare la funzione Lambda

1. In Function Code (Codice funzione), digita `com.amazonaws.lambda.demo.JobCompletionHandler` per Handler (Gestore).
2. Per Basic settings (Impostazioni di base), scegliere Edit (Modifica). Viene visualizzata la finestra di dialogo Edit basic settings (Modifica impostazioni di base).
   a. Scegliere 1024 per Memory (Memoria).
   b. Scegliere 10 secondi per Timeout.
   c. Scegli Save (Salva).
3. In Designer, scegliere + Add trigger (+ Aggiungi trigger). Viene visualizzata la finestra di dialogo Add trigger (Aggiungi trigger).
4. In Trigger configuration (Configurazione trigger, scegliere SNS.

In Argomento SNS, scegliere l'argomento Amazon SNS creato in Creazione dell'argomento SNS.

5. Scegli Enable trigger (Abilita trigger).
6. Per aggiungere il trigger, scegli Add (Aggiungi).
7. Selezionare Salva per salvare la funzione Lambda.

Configurazione del ruolo IAM Lambda

Per chiamare le operazioni di Amazon Rekognition Video, aggiungi la policy gestita di AmazonRekognitionFullAccessAWS al ruolo IAM Lambda. Le operazioni di avvio, ad esempio StartLabelDetection, richiedono anche le autorizzazioni di passaggio per il ruolo di servizio IAM utilizzato da Amazon Rekognition Video per accedere all'argomento Amazon SNS.

Per configurare il ruolo

1. Accedi AWS Management Console e apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
2. Nel riquadro di navigazione, seleziona Ruoli.
3. Nell'elenco, scegliere il nome del ruolo di esecuzione creato in Creazione della funzione Lambda.
4. Scegli la scheda Autorizzazioni.
5. Scegli Collega policy.
6. Scegli AmazonRekognitionFullAccessdall'elenco delle politiche.
7. Scegli Collega policy.
8. Scegliere nuovamente il ruolo di esecuzione.
9. Scegliere Add inline policy (Aggiungi policy inline).
10. Scegliere la scheda JSON.
11. Sostituisci la policy esistente con la seguente policy. Sostituisci `servicerole` con il ruolo del servizio IAM; creato in Creazione dell'argomento SNS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "mysid",
            "Effect": "Allow",
            "Action": "iam:PassRole",
```

```
            "Resource": "arn:servicerole"
        }
    ]
}
```

12. Scegli Verifica policy.
13. In Name* (Nome) digita un nome per la policy.
14. Scegliere Create Policy (Crea policy).

Crea il progetto AWS Toolkit for Eclipse Lambda

Quando viene attivata la funzione Lambda, il codice seguente ottiene lo stato di completamento dall'argomento Amazon SNS e le chiamate GetLabelDetectionper ottenere i risultati dell'analisi. Un numero di etichette rilevate e un elenco di etichette rilevate viene scritto in un registro. CloudWatch La funzione Lambda deve archiviare i risultati dell'analisi video per un utilizzo successivo.

Per creare il progetto AWS Toolkit for Eclipse Lambda

1. Crea un progetto AWS Toolkit for EclipseAWS Lambda.
   - In Project name: (Nome progetto), digita un nome per il progetto a tua scelta.
   - Per Nome classe:, inserisci JobCompletionHandler.
   - Per Input type: (Tipo input:), seleziona SNS Event (Evento SNS).
   - Lascia gli altri campi invariati.
2. In Eclipse Project Explorer, apri il metodo di gestione Lambda generato JobCompletionHandler (.java) e sostituisci il contenuto con quanto segue:

```
//Copyright 2018 Amazon.com, Inc. or its 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. I namespace Rekognition non sono stati risolti. Per risolvere il problema:
   - Ferma i tuo mouse sulla porzione sottolineata della riga `import com.amazonaws.services.rekognition.AmazonRekognition;`.
   - Scegli Fix project set up... (Correggi configurazione progetto...).
   - Scegli la versione più recente dell'archivio Amazon Rekognition.
   - Fai clic su OK per aggiungere l'archivio al progetto.
4. Salvare il file.
5. Fai clic con il pulsante destro del mouse nella finestra del codice Eclipse, scegli AWS Lambda, quindi scegli Upload function to AWS Lambda (Carica funzione su AWS Lambda).
6. Nella pagina Select Target Lambda Function (Seleziona funzione Lambda di destinazione), scegli la regione AWS da utilizzare.
7. Seleziona Choose an existing lambda function (Scegli una funzione Lambda esistente) e scegli al funzione Lambda che hai creato in Creazione della funzione Lambda.
8. Scegli Next (Successivo). Viene visualizzata la finestra di dialogo Function Configuration (Configurazione funzione).
9. In IAM Role (Ruolo IAM), selezionare il ruolo IAM creato in Creazione della funzione Lambda.

10. Seleziona Fine e la funzione Lambda viene caricata su AWS.

Test della funzione Lambda

Utilizzate il AWS CLI comando seguente per testare la funzione Lambda avviando l'analisi del rilevamento delle etichette di un video. Una volta completata l'analisi, la funzione Lambda viene attivata. Verifica che l'analisi sia riuscita controllando i registri di CloudWatch Logs.

Verifica della funzione Lambda

1. Carica un file video in formato MOV o MPEG-4 nel bucket S3. A scopo di verifica, carica un video non più lungo di 30 secondi.

   Per le istruzioni, consulta Caricamento di oggetti in Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

2. Eseguite il AWS CLI comando seguente per iniziare a rilevare le etichette in un video.

   ```
   aws rekognition start-label-detection --video
    "S3Object={Bucket="bucketname",Name="videofile"}" \
   --notification-channel "SNSTopicArn=TopicARN,RoleArn=RoleARN" \
   --region Region
   ```

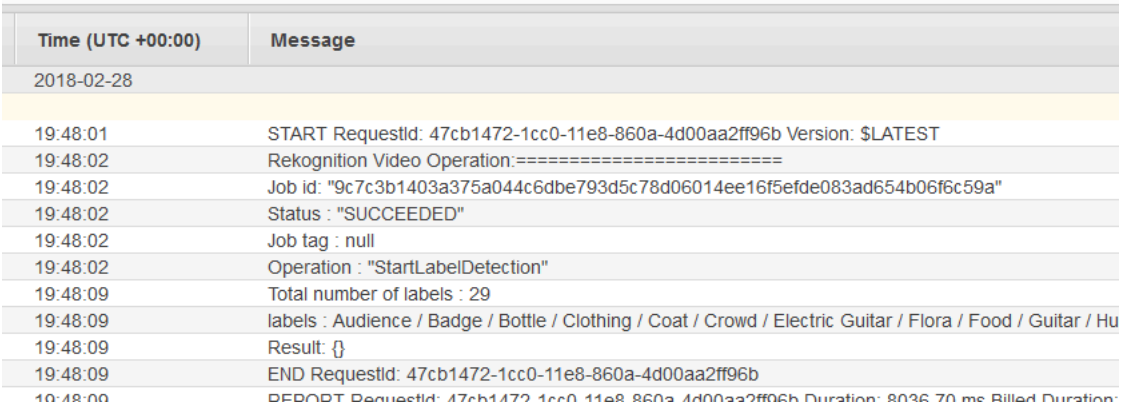
   Aggiorna i seguenti valori:

   - Modifica `bucketname` e `videofile` nel nome del bucket Amazon S3 e nel nome del file video in cui desideri rilevare le etichette.
   - Cambia `TopicARN` l'ARN dell'argomento Amazon SNS creato in Creazione dell'argomento SNS.
   - Cambia `RoleARN` l'ARN del ruolo IAM creato in Creazione dell'argomento SNS.
   - Passa `Region` alla AWS regione che stai utilizzando.

3. Prendi nota del valore di `JobId` nella risposta. La risposta si presenta in maniera analoga all'esempio di struttura JSON riportato di seguito.

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

4. Aprire la console https://console.aws.amazon.com/cloudwatch/.
5. Quando l'analisi viene completata, appare una voce di registro per la funzione Lambda nel Gruppo di log.
6. Scegli la funzione Lambda per vedere i flussi di log.
7. Scegli l'ultimo flusso di log per visualizzare le voci di registro effettuate dalla funzione Lambda. Se l'operazione è riuscita, è simile all'output seguente, che mostra i dettagli dell'operazione di riconoscimento video, tra cui l'ID del lavoro, il tipo di operazione "StartLabelDetection«e un elenco di categorie di etichette rilevate come Bottle, Clothing, Crowd e Food:



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:

   Il valore di Job id (Id lavoro) deve corrispondere al valore di `JobId` annotato al passaggio 3.

Utilizzo di Amazon Rekognition per la verifica dell'identità

Amazon Rekognition offre agli utenti diverse operazioni che consentono la semplice creazione di sistemi di verifica dell'identità. Amazon Rekognition consente all'utente di rilevare i volti in un'immagine e quindi confrontare i volti rilevati con altri volti confrontando i dati dei volti. Questi dati relativi ai volti vengono archiviati in container lato server denominati Raccolte. Sfruttando le operazioni di riconoscimento facciale, confronto facciale e gestione delle raccolte di Amazon Rekognition, puoi creare un'applicazione con una soluzione di verifica dell'identità.

Questo tutorial illustrerà due flussi di lavoro comuni per la creazione di applicazioni che richiedono la verifica dell'identità.

Il primo flusso di lavoro prevede la registrazione di un nuovo utente in una raccolta. Il secondo flusso di lavoro prevede la ricerca di una raccolta esistente allo scopo di accedere a un utente abituale.

Utilizzerai l'AWS SDK per Python per questo tutorial. Puoi anche vedere il GitHub repository degli esempi di AWS Documentation SDK per altri tutorial su Python.

Argomenti

- Prerequisiti
- Creazione di una raccolta
- Registrazione di un nuovo utente
- Accesso utente esistente

Prerequisiti

Prima di iniziare questo tutorial, devi installare Python e completare i passaggi necessari per configurare Python SDK. AWS Oltre a ciò, assicurati di:

- Aver creato un account AWS e un ruolo IAM
- Aver installato l'SDK Python (Boto3)
- Hai configurato correttamente le tue credenziali di accesso AWS
- Aver creato un bucket Amazon Simple Storage Service e caricato un'immagine che desideri utilizzare come ID ai fini della verifica dell'identità.
- Aver selezionato una seconda immagine da utilizzare come immagine di destinazione per la verifica dell'identità.

Creazione di una raccolta

Prima di poter registrare un nuovo utente o cercare un utente in una raccolta, è necessario disporre di una raccolta su cui lavorare. Una raccolta Amazon Rekognition è un container lato server utilizzato per archiviare informazioni sui volti rilevati.

Creazione della raccolta

Inizierai scrivendo una funzione che crei una raccolta da utilizzare con la tua applicazione. Amazon Rekognition archivia le informazioni sui volti rilevati in un container lato server chiamato Raccolte. Puoi cercare volti noti nelle informazioni sui volti archiviate in una raccolta. Per memorizzare le informazioni sul volto, devi prima creare una raccolta utilizzando l'operazione `CreateCollection`.

1. Scegli un nome per la raccolta da creare. Nel codice seguente, sostituisci il valore di `collection_id` con il nome della raccolta che desideri creare e sostituisci il valore di `region` con il nome della regione definita nelle credenziali utente. Puoi usare l'argomento `Tags` per

applicare tutti i tag che desideri alla raccolta, sebbene ciò non sia obbligatorio. L'operazione `CreateCollection` restituirà informazioni sulla raccolta che hai creato, incluso il relativo Arn. Prendi nota dell'Arn che ricevi come risultato dell'esecuzione del codice.

```
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. Salva ed esegui il codice. Copia l'Arn della raccolta.

   Ora che la raccolta Rekognition è stata creata, puoi memorizzare informazioni facciali e identificatori in quella raccolta. Potrai anche confrontare i volti con le informazioni memorizzate a scopo di verifica.

Registrazione di un nuovo utente

Ti consigliamo di poter registrare nuovi utenti e aggiungere le loro informazioni a una raccolta. Il processo di registrazione di un nuovo utente prevede in genere i seguenti passaggi:

Richiama l'operazione **`DetectFaces`**

Scrivi il codice per verificare la qualità dell'immagine del volto tramite l'operazione `DetectFaces`. Utilizzerai l'operazione `DetectFaces` per determinare se un'immagine catturata dalla fotocamera è adatta per l'elaborazione da parte dell'operazione `SearchFacesByImage`. L'immagine deve contenere solo un volto. Fornirai un file di immagine di input locale all'operazione `DetectFaces` e riceverai i dettagli per i volti rilevati nell'immagine. Il codice di esempio seguente fornisce l'immagine di input a `DetectFaces` e quindi verifica se nell'immagine è stato rilevato solo un volto.

1. Nel seguente esempio di codice, sostituisci `photo` con il nome dell'immagine di destinazione in cui desideri rilevare i volti. Dovrai anche sostituire il valore di `region` con il nome della regione associata al tuo account.

```
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. Salva ed esegui il codice in corso.

Richiama l'operazione **`CompareFaces`**

L'applicazione dovrà essere in grado di registrare nuovi utenti in una raccolta e confermare l'identità degli utenti abituali. Creerai prima le funzioni utilizzate per registrare un nuovo utente. Inizierai utilizzando l'`CompareFaces`operazione per confrontare un'input/target image of the user and a ID/ storedimmagine locale. Se viene rilevata una corrispondenza tra il volto in entrambe le immagini, puoi cercare nella raccolta per vedere se l'utente è registrato.

Inizia scrivendo una funzione che confronti un'immagine di input con l'immagine ID che hai archiviato nel tuo bucket Amazon S3. Nel seguente esempio di codice, dovrai fornire tu stesso l'immagine di input, che dovrebbe essere acquisita dopo aver utilizzato una qualche forma di rilevatore di vivacità. Dovrai anche passare il nome di un'immagine archiviata nel bucket Amazon S3.

1. Sostituisci il valore di `bucket` con il nome del bucket Amazon S3 che contiene il file sorgente. Dovrai inoltre sostituire il valore di `source_file` con il nome dell'immagine sorgente che stai utilizzando. Sostituisci il valore di `target_file` con il nome del file di destinazione che hai fornito. Sostituisci il valore di `region` con il nome della `region` definita nelle credenziali utente.

   Per specificare il livello minimo di affidabilità della corrispondenza che desideri ottenere nella risposta, utilizza `similarityThreshold` nella richiesta. I volti rilevati verranno restituiti nell'array `FaceMatches` solo se la confidenza è superiore a questa soglia. La scelta `similarityThreshold` deve riflettere la natura del processo di utilizzo specifico. Qualsiasi caso d'uso che coinvolga applicazioni di sicurezza critiche deve utilizzare 99 come soglia selezionata.

```
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. Salva ed esegui il codice in corso.

   Ti verrà restituito un oggetto di risposta contenente informazioni sul volto corrispondente e sul livello di confidenza.

Richiama l'operazione **`SearchFacesByImage`**

Se il livello di confidenza dell'operazione `CompareFaces` è superiore a quello scelto `SimilarityThreshold`, ti consigliamo di cercare nella raccolta un volto che potrebbe corrispondere all'immagine di input. Se viene trovata una corrispondenza nella tua raccolta, significa che probabilmente l'utente è già registrato in essa e non è necessario registrare un nuovo utente nella raccolta. Se non c'è nessuna corrispondenza, puoi registrare il nuovo utente nella raccolta.

1. Inizia scrivendo il codice che richiamerà l'operazione `SearchFacesByImage`. L'operazione prenderà come argomento un file di immagine locale e quindi cercherà un volto che corrisponda ai `Collection` volti più grandi rilevati nell'immagine fornita.

   Nel seguente esempio di codice, modifica il valore di `collectionId` nella raccolta in cui desideri cercare. Sostituisci il valore di `region` con il nome della regione associata al tuo account. Dovrai inoltre sostituire il valore di `photo` con il nome del file di input. Dovrai anche

specificare una soglia di somiglianza sostituendo il valore di `threshold` con un percentile scelto.

```
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. Salva ed esegui il codice in corso. Se c'è stata una corrispondenza, significa che la persona riconosciuta nell'immagine fa già parte della raccolta e non è necessario passare ai passaggi successivi. In questo caso, puoi semplicemente consentire all'utente di accedere all'applicazione.

Richiama l'operazione **`IndexFaces`**

Supponendo che non sia stata trovata alcuna corrispondenza nella raccolta che hai cercato, ti consigliamo di aggiungere il volto dell'utente alla tua raccolta. Puoi farlo chiamando l'operazione `IndexFaces`. Quando chiami IndexFaces, Amazon Rekognition estrae i tratti del viso di un volto identificato nell'immagine di input, archiviando i dati nella raccolta specificata.

1. Inizia scrivendo il codice per chiamare `IndexFaces`. Sostituisci il valore di `image` con il nome del file locale che desideri utilizzare come immagine di input per l'operazione. IndexFaces Dovrai inoltre sostituire il valore di `photo_name` con il nome desiderato per l'immagine di

input. Assicurati di sostituire il valore di `collection_id` con l'IDE della raccolta creata in precedenza. Poi, sostituisci il valore di `region` con il nome della regione associata al tuo account. Dovrai anche specificare un valore per il parametro `MaxFaces` di input, che definisce il numero massimo di volti in un'immagine che devono essere indicizzati. Il valore predefinito per questo parametro è 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. Salva ed esegui il codice in corso. Determina se desideri salvare i dati restituiti dall'operazione `IndexFaces`, ad esempio il FaceID assegnato alla persona nell'immagine. La prossima sezione esaminerà come salvare questi dati. Copia i valori restituiti `FaceId`, `ImageId` e `Confidence` prima di procedere.

Salva immagini e dati FaceID in Amazon S3 e Amazon DynamoDB

Una volta ottenuto il Face ID per l'immagine di input, i dati dell'immagine possono essere salvati in Amazon S3, mentre i dati del volto e l'URL dell'immagine possono essere inseriti in un database come DynamoDB.

1. Scrivi il codice per caricare l'immagine di input nel database Amazon S3. Nell'esempio di codice che segue, sostituisci il valore di `bucket` con il nome del bucket in cui desideri caricare il file, quindi sostituisci il valore di `file_name` con il nome del file locale che desideri archiviare nel tuo bucket Amazon S3. Fornisci un nome chiave che identifichi il file nel bucket Amazon S3 sostituendo il valore di `key_name` con un nome che desideri assegnare al file di immagine. Il file da caricare è lo stesso definito negli esempi di codice precedenti, ovvero il file di input per cui è stato utilizzato IndexFaces. Infine, sostituisci il valore di `region` con il nome della regione associata al tuo account.

```
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. Salva ed esegui il seguente esempio di codice per caricare l'immagine di input su Amazon S3.
3. Ti consigliamo di salvare anche il Face ID restituito in un database. Questo può essere fatto creando una tabella di database DynamoDB e quindi caricando il Face ID su quella tabella. Nell'esempio di codice seguente viene creata una tabella DynamoDB. Nota che devi eseguire il codice che crea questa tabella solo una volta. Nel codice seguente, sostituisci il valore di `region` con il valore della regione associata al tuo account. Dovrai anche sostituire il valore di `database_name` con il nome che desideri assegnare alla tabella DynamoDB.

```
import boto3

# Create DynamoDB database with image URL and face data, face ID

def create_dynamodb_table(table_name, region):
    dynamodb = boto3.client("dynamodb", region_name=region)

    table = dynamodb.create_table(
        TableName=table_name,
        KeySchema=[{
                'AttributeName': 'FaceID', 'KeyType': 'HASH'  # Partition key
                },],
            AttributeDefinitions=[
            {
                'AttributeName': 'FaceID', 'AttributeType': 'S'  }, ],
                ProvisionedThroughput={
            'ReadCapacityUnits': 10, 'WriteCapacityUnits': 10  }
    )
    print(table)
    return table

region = "region-name"
database_name = 'database-name'
dynamodb_table = create_dynamodb_table(database_name, region)
print("Table status:", dynamodb_table)
```

4. Salva ed esegui il codice procedente per creare la tua tabella.
5. Dopo aver creato la tabella, puoi FaceId caricarvi i dati restituiti. Per fare ciò, stabilirai una connessione alla tabella con la funzione Tabella e quindi utilizzerai la funzione `put_item` per caricare i dati.

Nel seguente esempio di codice, sostituisci il valore di `bucket` con il nome del bucket contenente l'immagine di input che hai caricato su Amazon S3. Dovrai inoltre sostituire il valore di `file_name` con il nome del file di input che hai caricato nel tuo bucket Amazon S3 e il valore di `key_name` con la chiave che hai usato in precedenza per identificare il file di input. Infine, sostituisci il valore di `region` con il nome della regione associata al tuo account. Questi valori devono corrispondere a quelli forniti nel passaggio 1.

`AddDBEntry`Memorizza i valori FaceId ImageId, e Confidence assegnati a un volto in una raccolta. Fornisci alla funzione seguente i valori che hai salvato durante la fase 2 della sezione successiva `IndexFaces`.

```
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. Salva ed esegui l'esempio di codice procedente per memorizzare i dati Face ID restituiti in una tabella.

Accesso utente esistente

Dopo che un utente è stato registrato in una raccolta, può essere autenticato al suo ritorno utilizzando l'operazione `SearchFacesByImage`. È necessario ottenere un'immagine di input e quindi verificare la qualità dell'immagine di input utilizzata `DetectFaces`. Ciò determina se è stata utilizzata un'immagine adatta prima di eseguire l'operazione `SearchFacesbyImage`.

Chiama l' DetectFaces operazione

1. Utilizzerai l'operazione `DetectFaces` per controllare la qualità dell'immagine del volto e determinare se un'immagine catturata dalla fotocamera è adatta per l'elaborazione da parte dell'operazione `SearchFacesByImage`. L'immagine deve contenere solo un volto. Il seguente esempio di codice prende un'immagine di input e la fornisce all'operazione `DetectFaces`.

   Nell'esempio di codice seguente, sostituisci il valore di `photo` con il nome dell'immagine di destinazione e sostituisci il valore di `region` con il nome della regione associata al tuo account.

```
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. Salva ed esegui il codice.

Chiama l' SearchFacesByImage operazione

1. Scrivi il codice per confrontare il volto rilevato con i volti della raccolta con `SearchFacesByImage`. Utilizzerai il codice mostrato nella sezione Procedura di registrazione di un nuovo utente e fornirai l'immagine di input per l'operazione `SearchFacesByImage`.

   Nel seguente esempio di codice, modifica il valore di `collectionId` nella raccolta in cui desideri cercare. Modificherai anche il valore di `bucket` con il nome di un bucket Amazon S3 e il valore di un file `fileName` di immagine in quel bucket. Sostituisci il valore di `region` con il nome della regione associata al tuo account. Dovrai anche specificare una soglia di somiglianza sostituendo il valore di `threshold` con un percentile scelto.

   ```
   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. Salva ed esegui il codice.

Verifica il FaceID restituito e il livello di confidenza

Ora puoi verificare le informazioni sulla corrispondenza FaceId stampando elementi di risposta come gli FaceId attributi Similarity e Confidence.

```
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']))
```

Utilizzo di Amazon Rekognition (REK) per rilevare le etichette per le applicazioni di marketing

Questo tutorial ti guida nella creazione di un'applicazione Python di esempio che potrebbe essere utilizzata per inviare e-mail a persone, sulla base di immagini caricate su un sito Web. L'applicazione di esempio è progettata per coinvolgere gli utenti in una campagna di marketing sui social media inviando loro e-mail personalizzate sulle offerte di viaggio se nelle loro foto vengono riconosciuti dei punti di riferimento.

La soluzione integra vari AWS servizi, tra cui Amazon Rekognition, Amazon S3, DynamoDB e Amazon SES CloudWatch. L'applicazione utilizza Amazon Rekognition per rilevare le etichette nelle

immagini caricate da account riconosciuti in un database DynamoDB, quindi al titolare dell'account viene inviata un'e-mail di marketing basata sulle etichette rilevate nell'immagine. L'architettura completa della soluzione è la seguente:

- Archivia i dati degli utenti nel database DynamoDB.
- L'utente carica i dati delle immagini e i metadati (numero di account utente) su Amazon S3.
- Una funzione Lambda richiama DetectLabelsper identificare e registrare le etichette nell'immagine caricata, cercare l'e-mail dell'utente in DynamoDB e inviare un'e-mail di marketing all'utente che ha caricato l'immagine.
- I risultati della funzione Lambda vengono registrati CloudWatch per una revisione successiva.

Ecco una panoramica di tutti i passaggi del tutorial:

1. Crea e compila una tabella DynamoDB.
2. Scrivi una funzione Lambda con Logging and Notification.
3. Configura la funzione Lambda e le autorizzazioni.
4. Carica immagini e metadati con Amazon S3.
5. Registri dei sondaggi CloudWatch .

Argomenti

- Prerequisiti
- Creazione e compilazione di una tabella DynamoDB
- Creazione di una funzione Lambda con registrazione e notifica
- Impostazione delle autorizzazioni e della funzione Lambda
- Caricamento di immagini su Amazon S3 con metadati
- CloudWatch Registri dei sondaggi del client

Prerequisiti

Prima di iniziare questo tutorial, avrai bisogno di:

- Un AWS account e autorizzazioni IAM appropriate.
- Python e Boto3 installati nel tuo ambiente di sviluppo.

- Conoscenze di base di Lambda, Amazon S3, DynamoDB, Amazon SES e. CloudWatch
- Familiarità di base con Lambda e Amazon SES.

Creazione e compilazione di una tabella DynamoDB

Per iniziare questo tutorial, creeremo una tabella DynamoDB per archiviare i dati dei clienti come: indirizzo e-mail, età, numero di telefono e stato di iscrizione, utilizzando una variabile AccountNumber "" come chiave primaria. Inseriremo anche alcuni dati di esempio nella tabella.

Creazione della tabella DynamoDB

Per prima cosa, creeremo la tabella DynamoDB. Ecco come configurarlo usando Python e Boto3. Creiamo una funzione chiamata `create_user_table` e ci connettiamo alla risorsa DynamoDB al suo interno. Nell'esempio di codice riportato di seguito, sostituisci il valore di «region_name» con il nome della regione in cui opera l'account, quindi esegui la cella di codice per creare la tabella.

```
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()
```

L'esecuzione di questo script ha impostato una tabella DynamoDB `CustomerDataTable` denominata `AccountNumber` con come chiave primaria.

Inserimento di dati di esempio

Ora vorremo inserire alcuni dati di esempio nella tabella. Questi dati di esempio ci aiuteranno a testare la funzionalità completa dell'applicazione.

Creeremo una seconda funzione che aggiunge dati di esempio alla stessa che `CustomerDataTable` abbiamo creato in precedenza. Il codice seguente crea tre voci di esempio e ogni voce include un numero di account, un indirizzo e-mail, un'età, un numero di telefono e lo stato dell'iscrizione. Nell'esempio di codice riportato di seguito, sostituisci il valore di `region_name` con il nome della regione in cui opera l'account, quindi esegui la cella di codice per creare la tabella. Se desideri testare la parte dell'app relativa alla consegna delle e-mail, sostituisci il valore riportato `EmailAddress` nella prima voce relativa al cliente di seguito con un indirizzo e-mail a cui puoi ricevere e-mail. Salva ed esegui il codice.

```
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()
```

Con la tabella DynamoDB impostata e compilata, possiamo ora procedere all'integrazione di questo recupero dei dati nella funzione Lambda. Ciò consente alla nostra applicazione di recuperare i dettagli dell'utente in base al numero di account che verrà identificato nel prossimo flusso di lavoro di elaborazione delle immagini.

Creazione di una funzione Lambda con registrazione e notifica

Ora possiamo creare una funzione Lambda. Vogliamo assicurarci che la funzione Lambda, che viene attivata al caricamento dell'immagine, possa leggere sia i dati dell'immagine che i metadati dell'account e utilizzarla per eseguire una ricerca nella tabella DynamoDB per i dati utente associati. Ciò significa che non solo avremo bisogno di richiamare la funzione di Amazon Rekognition, ma anche di una DetectLabels funzione che la accolga AccountNumber e la utilizzi per recuperare l'indirizzo e-mail associato da DynamoDB. Inoltre, abbiamo bisogno di una funzione che invii un'e-mail all'indirizzo e-mail utilizzando Amazon SES. Infine, utilizziamo un logger per registrare le informazioni sul processo. Questi dati verranno visualizzati da CloudWatch.

Creazione di una funzione Lambda

Ecco uno schema di come una funzione Lambda potrebbe gestire questi requisiti. Nel seguente esempio di codice, assicuratevi che sia specificata la tabella DynamoDB corretta, se avete usato qualcosa di CustomerDataTable diverso da quello utilizzato come nome della tabella. Inoltre, nella funzione «send_marketing_email», dovreste sostituire il valore dell'argomento «Source» con un indirizzo e-mail a cui avete accesso che funga da e-mail di invio.

```
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")
```

Ora che abbiamo scritto la funzione Lambda, dobbiamo configurare le autorizzazioni per Lambda e creare un'istanza della nostra funzione Lambda nella console di gestione. AWS

Impostazione delle autorizzazioni e della funzione Lambda

Crea o aggiorna il ruolo IAM

Prima di poter utilizzare una funzione Lambda per gestire qualsiasi immagine caricata dagli utenti, dobbiamo gestirne le autorizzazioni. La funzione Lambda necessita di un ruolo IAM specifico con policy che le consentano di interagire con Amazon S3, DynamoDB, Amazon SES, Amazon Rekognition e. CloudWatch

Per configurare il tuo ruolo IAM per la funzione Lambda:

1. Console AWS di gestione aperta.
2. Vai a IAM > Ruoli > Crea ruolo.

   Scegli «Crea ruolo».
3. Seleziona Lambda come servizio che utilizzerà questo ruolo. Fai clic su «Avanti».
4. Allega le seguenti politiche (nota che queste politiche sono selezionate esclusivamente a scopo dimostrativo, in un ambiente di produzione reale dovresti limitare le autorizzazioni solo a quelle necessarie.):

   - Amazon S3 ReadOnlyAccess
   - AmazonRekognitionReadOnlyAccess
   - AmazonDynamoDBFullAccesso
   - SESFullAccesso Amazon
   - AWSLambdaEsegui
   - AWSLambdaBasicExecutionRole (per la CloudWatch registrazione)

   Fai clic su «Avanti».
5. Assegna un nome al ruolo e una descrizione, quindi crea il ruolo scegliendo «Crea ruolo».
6. Dopo aver configurato le autorizzazioni appropriate per Lambda, possiamo creare un'istanza della funzione Lambda utilizzando la console di gestione. AWS

   Accedi al servizio Lambda nella Console di gestione AWS.
7. Fai clic su «Crea funzione». Scegli «Autore da zero».

8. Immettete il nome di una funzione e selezionate il runtime di Python. In «Modifica il ruolo di esecuzione predefinito», seleziona «Usa un ruolo esistente», quindi scegli il ruolo IAM che hai creato in precedenza.
9. Ora devi creare e aggiornare il codice nella scheda della funzione lambda. Vai alla scheda chiamata «lambda_function» e sostituisci il codice con l'esempio di codice Lambda precedente.

   Salva le modifiche e distribuiscile.
10. Ora devi configurare un evento Amazon S3 come Lambda Trigger.

    Nella scheda/pagina di configurazione della funzione Lambda, vai su Trigger, fai clic su «Aggiungi trigger».
11. Seleziona Amazon S3 dall'elenco dei trigger disponibili.
12. Per configurare il trigger: seleziona il bucket da cui deve essere attivata la funzione.

    Scegli il tipo di evento, PUT. Facoltativamente, è possibile specificare un prefisso o un suffisso se si desidera elaborare solo file con determinati nomi o tipi.
13. Attiva il trigger facendo clic su «Aggiungi» e salva la configurazione.

Verifica degli indirizzi e-mail di Amazon SES

Prima di poter utilizzare Amazon SES per inviare e-mail, devi verificare gli indirizzi e-mail del mittente e del destinatario. Per farlo:

1. Vai alla console Amazon SES. Vai a «Gestione delle identità», quindi a «Indirizzi e-mail».
2. Fai clic su «Verifica un nuovo indirizzo e-mail». Inserisci l'indirizzo email che desideri verificare e fai clic su «Verifica questo indirizzo e-mail». Riceverai un'email con un link di verifica. Fai clic sul link per completare la procedura di verifica.
3. Dopo aver verificato gli indirizzi e-mail di entrambi gli account, impostato la funzione Lambda con le autorizzazioni corrette e creato alcuni dati cliente di esempio, puoi testare la funzione Lambda caricando un'immagine di prova nel bucket Amazon S3 scelto.

Caricamento di immagini su Amazon S3 con metadati

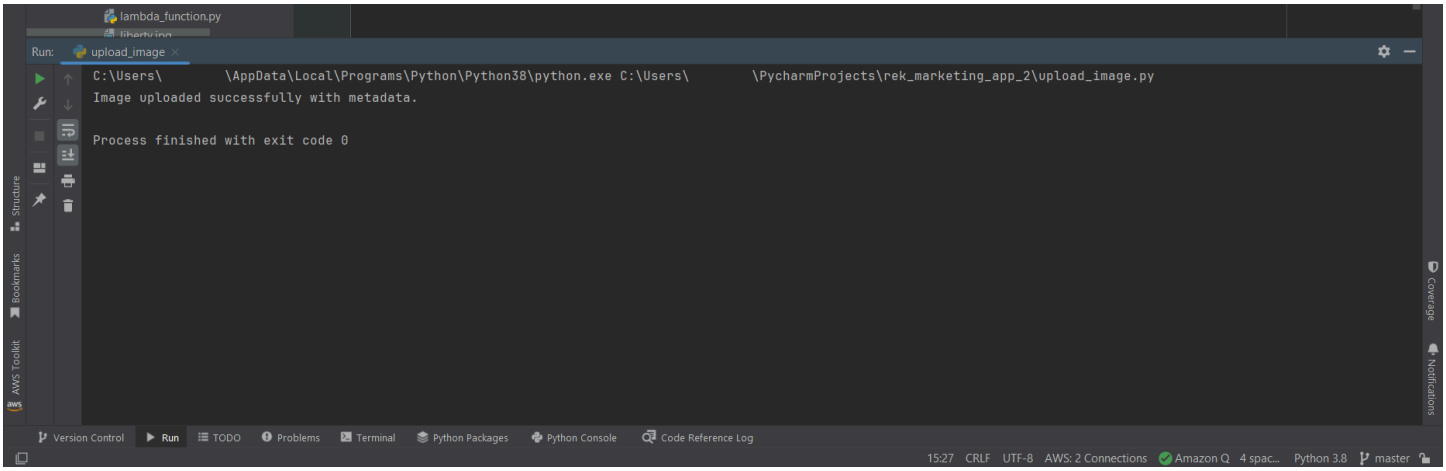
Ora possiamo caricare un'immagine di prova nel bucket Amazon S3 che abbiamo scelto utilizzando la AWS console o lo script che abbiamo preparato in precedenza. Scrivi uno script che carichi un'immagine nel bucket Amazon S3 precedentemente specificato nella funzione Lambda.

Nell'esempio di codice riportato di seguito, specifica il percorso dell'immagine come primo argomento di «upload_image_to_s3", il nome del bucket come secondo argomento e il numero di account dell'utente che carica l'immagine come argomento finale.

```
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 questa funzione, il `ExtraArgs` parametro del `upload_fileobj` metodo viene utilizzato per includere i metadati definiti dall'utente () insieme all'immagine. `account_number` Questi metadati possono essere successivamente utilizzati per elaborare l' AWS immagine di conseguenza.

Salva ed esegui lo script. Questo caricherà l'immagine.



Pochi minuti dopo aver caricato l'immagine, dovresti ricevere un'email all'indirizzo che hai precedentemente associato all'account specificato qui.

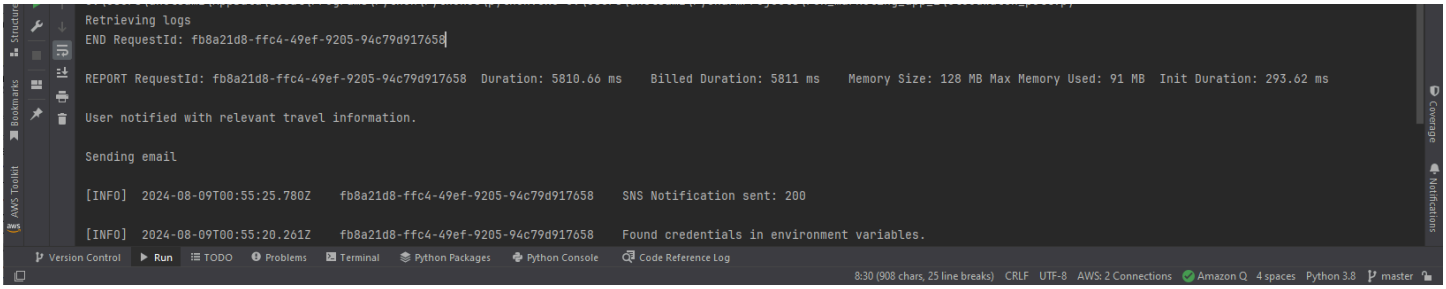
CloudWatch Registri dei sondaggi del client

Controlla CloudWatch i log della tua funzione Lambda per vedere se è stata attivata ed eseguita come previsto. I log sono disponibili in CloudWatch > Logs > Log groups >//lambda/your_function_name.AWS Puoi anche scrivere uno script per accedere e stampare i log. Il seguente esempio di codice analizza i log del gruppo Lambda, stampando i log generati nell'ultima ora. Salva ed esegui il codice.

```
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'])
```

L'esecuzione del codice dovrebbe stampare i log e dovrebbe essere possibile vedere che l'utente è stato avvisato tramite un'e-mail contenente le informazioni di viaggio pertinenti.



Hai creato con successo un'applicazione in grado di rilevare le etichette nelle immagini caricate in un bucket Amazon S3 e quindi inviare un'e-mail all'utente che ha caricato l'immagine con un messaggio promozionale. Assicurati di eliminare tutte le risorse che non ti servono più in modo che non ti vengano addebitati costi.

Rilevamento di etichette in un'immagine usando Lambda e Python

AWS Lambda è un servizio di elaborazione che è possibile utilizzare per eseguire codice senza effettuare il provisioning o gestire server. Puoi chiamare le operazioni dell'API Rekognition dall'interno di una funzione Lambda. Le seguenti istruzioni mostrano come creare una funzione Lambda in Python che chiama `DetectLabels`.

La funzione Lambda chiama `DetectLabels` e restituisce una matrice di etichette rilevate nell'immagine e anche il livello di affidabilità con cui sono state rilevate.

Le istruzioni includono un codice Python di esempio che mostra come chiamare la funzione Lambda e fornirle un'immagine proveniente da un bucket Amazon S3 o dal computer locale.

Assicurati che le immagini scelte soddisfino i limiti di Rekognition. Consulta le linee guida e le quote in Rekognition e l'API Reference per informazioni sui limiti di tipo e dimensione DetectLabels dei file di immagine.

Creazione di una funzione Lambda (console)

In questo passaggio, crea una funzione Lambda vuota e un ruolo di esecuzione IAM che consente alla funzione Lambda di chiamare l'operazione `DetectLabels`. Nei passaggi successivi aggiungete il codice sorgente e, facoltativamente, aggiungete un livello alla funzione Lambda.

Se utilizzi documenti archiviati in un bucket Amazon S3, questo passaggio dimostra anche come concedere l'accesso al bucket in cui sono archiviati i documenti.

Per creare una funzione (console AWS Lambda)

1. Accedi a AWS Management Console e apri la AWS Lambda console all'indirizzo https://console.aws.amazon.com/lambda/.
2. Scegli Crea funzione. Per ulteriori informazioni, consulta Creare una funzione Lambda con la console.
3. Scegliere le opzioni seguenti:
   - Scegli Crea da zero.
   - Inserire un valore per Nome della funzione.
   - Per Runtime, scegliere la versione più recente di Python.
   - In Architecture (Architettura), scegli x86_64.
4. Scegliere Crea funzione per creare una funzione AWS Lambda .
5. Nella pagina della funzione, scegli la scheda Configurazione.
6. Nel riquadro Autorizzazioni, in Ruolo di esecuzione, scegli il nome del ruolo per aprire il ruolo nella console IAM.
7. Nella scheda Autorizzazioni, scegli Aggiungi autorizzazioni, poi Crea policy inline.
8. Scegliere la scheda JSON e sostituire la policy con la policy seguente:

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

```
    ]
}
```

9. Scegli Verifica policy.
10. Inserisci un nome per la policy, ad esempio DetectLabels-access.
11. Scegli Create Policy (Crea policy).
12. Se stai archiviando documenti per l'analisi in un bucket Amazon S3, devi aggiungere una policy di accesso Amazon S3. A tale scopo, ripeti i passaggi da 7 a 11 nella console AWS Lambda e apporta le seguenti modifiche.

    a. Per il passaggio 8, utilizza la seguente policy. Sostituisci *bucket/folder path* con il bucket Amazon S3 e il percorso della cartella i documenti che desideri analizzare.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3Access",
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::bucket/folder path/*"
        }
    ]
}
```

    b. Per la fase 10, scegli un nome di policy diverso, ad esempio S3Bucket-access.

(Opzionale) Creazione di un livello (console)

Non è necessario eseguire questo passaggio per utilizzare una funzione Lambda e una chiamata `DetectLabels`.

L'`DetectLabels`operazione è inclusa nell'ambiente Lambda Python predefinito come parte di AWS SDK for Python (Boto3).

Se altre parti della tua funzione Lambda richiedono aggiornamenti recenti del AWS servizio che non si trovano nell'ambiente Lambda Python predefinito, puoi eseguire questo passaggio per aggiungere la versione più recente di Boto3 SDK come livello alla tua funzione.

Per aggiungere l'SDK come livello, devi prima creare un archivio di file zip contenente l'SDK Boto3. Quindi, crea un livello e aggiungi l'archivio di file zip al livello. Per ulteriori informazioni, consulta la sezione Utilizzo dei livelli con la funzione Lambda.

Per creare un livello (console)

1. Apri un prompt dei comandi e inserisci i seguenti comandi per creare un pacchetto di distribuzione con la versione più recente dell'SDK. AWS

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

2. Prendete nota del nome del file zip (boto3-layer.zip), utilizzato nel passaggio 8 di questa procedura.
3. Apri la AWS Lambda console all'indirizzo. https://console.aws.amazon.com/lambda/
4. Nel riquadro di navigazione scegli Layers (Livelli).
5. Scegli Create layer (Crea livello).
6. Immetti i valori per Nome (Nome) e Description (Descrizione).
7. In Tipo di voce del codice, scegli Carica un file .zip e seleziona Carica.
8. Nella finestra di dialogo, scegliete l'archivio di file zip (boto3-layer.zip) che avete creato nel passaggio 1 di questa procedura.
9. Per Compatible Runtimes, scegliere la versione più recente di Python.
10. Scegliere Crea per creare il livello.
11. Scegli l'icona del menu del pannello di navigazione.
12. Nel riquadro di navigazione, seleziona Funzioni.
13. Nell'elenco delle risorse, scegliere la funzione creata in precedenza in???.
14. Scegli la scheda Codice.
15. Nella sezione Livelli, scegli Aggiungi un livello.
16. Scegliete Livelli personalizzati.
17. In Livelli personalizzati, scegli il nome del livello che hai inserito nel passaggio 6.
18. In Versione scegli la versione del livello, che dovrebbe essere 1.
19. Scegli Aggiungi.

Aggiungere codice Python (console)

In questo passaggio, aggiungi il codice Python alla funzione Lambda tramite l'editor di codice della console Lambda. Il codice rileva le etichette in un'immagine utilizzando l'operazione `DetectLabels`. Restituisce una matrice di etichette rilevate nell'immagine e anche il livello di affidabilità nelle etichette rilevate.

Il documento fornito per l'operazione `DetectLabels` può trovarsi in un bucket Amazon S3 o in un computer locale.

Aggiungere codice Python (console)

1. Passa alla scheda Codice.
2. Nell'editor di codice, sostituisci il codice in lambda_function.py con il seguente codice:

```
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. Per distribuire la funzione Lambda, scegli Deploy.

Aggiungere codice Python (console)

Ora che hai creato la tua funzione Lambda, puoi richiamarla per rilevare le etichette in un'immagine.

In questo passaggio, esegui il codice Python sul tuo computer, che passa un'immagine locale o un'immagine in un bucket Amazon S3 alla tua funzione Lambda.

Assicurati di eseguire il codice nella stessa AWS regione in cui hai creato la funzione Lambda. Puoi visualizzare la AWS regione per la tua funzione Lambda nella barra di navigazione della pagina dei dettagli della funzione nella console Lambda.

Se la funzione Lambda restituisce un errore di timeout, estendi il periodo di timeout per la funzione Lambda. Per ulteriori informazioni, consulta Configurazione del timeout della funzione (console).

Per ulteriori informazioni sulla chiamata di una funzione Lambda dal codice, consulta Chiamata delle funzioni AWS Lambda.

Prova la funzione Lambda

1. Se non lo hai ancora fatto, esegui queste operazioni:

   a. Assicurati che l'utente disponga dell'autorizzazione `lambda:InvokeFunction`. Si più usare la seguente policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "InvokeLambda",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "ARN for lambda function"
        }
    ]
}
```

Puoi ottenere l'ARN per la tua funzione Lambda dalla panoramica delle funzioni nella console Lambda.

Per fornire l'accesso, aggiungi autorizzazioni agli utenti, gruppi o ruoli:

- Utenti e gruppi in: AWS IAM Identity Center

  Crea un set di autorizzazioni. Segui le istruzioni riportate nella pagina Create a permission set (Creazione di un set di autorizzazioni) nella Guida per l'utente di AWS IAM Identity Center .

- Utenti gestiti in IAM tramite un provider di identità:

  Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate nella pagina Create a role for a third-party identity provider (federation) della Guida per l'utente IAM.

- Utenti IAM:
  - Crea un ruolo che l'utente possa assumere. Segui le istruzioni riportate nella pagina Create a role for an IAM user della Guida per l'utente IAM.
  - (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate nella pagina Aggiunta di autorizzazioni a un utente (console) nella Guida per l'utente IAM.

b. Installa e configura AWS SDK per Python. Per ulteriori informazioni, consulta Passaggio 2: configura AWS CLI e AWS SDKs.

2. Salva il seguente codice in un file denominato `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. Eseguire il codice. Se il documento si trova in un bucket Amazon S3, assicurati che sia lo stesso bucket specificato in precedenza nel passaggio 12 di ???.

   In caso di successo, il codice restituisce una risposta JSON parziale per ogni tipo di blocco rilevato nel documento.

Esempi di codice per l'utilizzo di Amazon Rekognition AWS SDKs

I seguenti esempi di codice mostrano come usare Amazon Rekognition AWS con un kit di sviluppo software (SDK). Gli esempi di codice in questo capitolo hanno lo scopo di integrare gli esempi di codice presenti nel resto di questa guida.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Per un elenco completo di guide ed esempi di AWS codice per sviluppatori SDK, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Nozioni di base

Buongiorno Amazon Rekognition

Il seguente esempio di codice mostra come iniziare a usare Amazon Rekognition.

C++

SDK per C++

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Codice per il CMake file 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})
```

Codice per il file sorgente 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;
}
```

- Per i dettagli sull'API, consulta la ListCollectionssezione AWS SDK per C++ API Reference.

Esempi di codice

- Esempi di base per l'utilizzo di Amazon Rekognition AWS SDKs
  - Buongiorno Amazon Rekognition
  - Azioni per l'utilizzo di Amazon Rekognition AWS SDKs
    - Utilizzo CompareFaces con un AWS SDK o una CLI
    - Utilizzo CreateCollection con un AWS SDK o una CLI
    - Utilizzo DeleteCollection con un AWS SDK o una CLI
    - Utilizzo DeleteFaces con un AWS SDK o una CLI
    - Utilizzo DescribeCollection con un AWS SDK o una CLI
    - Utilizzo DetectFaces con un AWS SDK o una CLI
    - Utilizzo DetectLabels con un AWS SDK o una CLI
    - Utilizzo DetectModerationLabels con un AWS SDK o una CLI
    - Utilizzo DetectText con un AWS SDK o una CLI
    - Utilizzo DisassociateFaces con un AWS SDK o una CLI
    - Utilizzo GetCelebrityInfo con un AWS SDK o una CLI
    - Utilizzo IndexFaces con un AWS SDK o una CLI
    - Utilizzo ListCollections con un AWS SDK o una CLI
    - Utilizzo ListFaces con un AWS SDK o una CLI
    - Utilizzo RecognizeCelebrities con un AWS SDK o una CLI
    - Utilizzo SearchFaces con un AWS SDK o una CLI
    - Utilizzo SearchFacesByImage con un AWS SDK o una CLI

- Scenari per l'utilizzo di Amazon Rekognition AWS SDKs
  - Crea una raccolta Amazon Rekognition e trova i volti al suo interno utilizzando un SDK AWS
  - Creazione di un'applicazione di gestione delle risorse fotografiche che consente agli utenti di gestire le foto utilizzando etichette
  - Rileva PPE nelle immagini con Amazon Rekognition utilizzando un SDK AWS
  - Rileva e visualizza elementi nelle immagini con Amazon Rekognition utilizzando un SDK AWS
  - Rileva i volti in un'immagine utilizzando un SDK AWS
  - Rileva le informazioni nei video utilizzando Amazon Rekognition e l'SDK AWS
  - Rileva oggetti nelle immagini con Amazon Rekognition utilizzando un SDK AWS
  - Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un SDK AWS
  - Salva EXIF e altre informazioni sull'immagine utilizzando un SDK AWS

Esempi di base per l'utilizzo di Amazon Rekognition AWS SDKs

I seguenti esempi di codice mostrano come utilizzare le nozioni di base di Amazon Rekognition con. AWS SDKs

Esempi

- Buongiorno Amazon Rekognition
- Azioni per l'utilizzo di Amazon Rekognition AWS SDKs
  - Utilizzo CompareFaces con un AWS SDK o una CLI
  - Utilizzo CreateCollection con un AWS SDK o una CLI
  - Utilizzo DeleteCollection con un AWS SDK o una CLI
  - Utilizzo DeleteFaces con un AWS SDK o una CLI
  - Utilizzo DescribeCollection con un AWS SDK o una CLI
  - Utilizzo DetectFaces con un AWS SDK o una CLI
  - Utilizzo DetectLabels con un AWS SDK o una CLI
  - Utilizzo DetectModerationLabels con un AWS SDK o una CLI
  - Utilizzo DetectText con un AWS SDK o una CLI
  - Utilizzo DisassociateFaces con un AWS SDK o una CLI
  - Utilizzo GetCelebrityInfo con un AWS SDK o una CLI
  - Utilizzo IndexFaces con un AWS SDK o una CLI

- Utilizzo ListCollections con un AWS SDK o una CLI
- Utilizzo ListFaces con un AWS SDK o una CLI
- Utilizzo RecognizeCelebrities con un AWS SDK o una CLI
- Utilizzo SearchFaces con un AWS SDK o una CLI
- Utilizzo SearchFacesByImage con un AWS SDK o una CLI

Buongiorno Amazon Rekognition

Il seguente esempio di codice mostra come iniziare a usare Amazon Rekognition.

C++

SDK per C++

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Codice per il CMake file 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})
```

Codice per il file sorgente 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;
}
```

- Per i dettagli sull'API, consulta la ListCollectionssezione AWS SDK per C++ API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUsare Rekognition con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Azioni per l'utilizzo di Amazon Rekognition AWS SDKs

I seguenti esempi di codice mostrano come eseguire singole azioni di Amazon Rekognition con. AWS SDKs Ogni esempio include un collegamento a GitHub, dove puoi trovare le istruzioni per la configurazione e l'esecuzione del codice.

Questi estratti richiamano l'API Amazon Rekognition e sono estratti di codice da programmi più grandi che devono essere eseguiti nel contesto. È possibile visualizzare le azioni nel contesto in Scenari per l'utilizzo di Amazon Rekognition AWS SDKs .

Gli esempi seguenti includono solo le operazioni più comunemente utilizzate. Per un elenco completo, consulta la Documentazione di riferimento delle API Amazon Rekognition.

Esempi

- Utilizzo CompareFaces con un AWS SDK o una CLI
- Utilizzo CreateCollection con un AWS SDK o una CLI
- Utilizzo DeleteCollection con un AWS SDK o una CLI
- Utilizzo DeleteFaces con un AWS SDK o una CLI
- Utilizzo DescribeCollection con un AWS SDK o una CLI
- Utilizzo DetectFaces con un AWS SDK o una CLI
- Utilizzo DetectLabels con un AWS SDK o una CLI
- Utilizzo DetectModerationLabels con un AWS SDK o una CLI
- Utilizzo DetectText con un AWS SDK o una CLI
- Utilizzo DisassociateFaces con un AWS SDK o una CLI
- Utilizzo GetCelebrityInfo con un AWS SDK o una CLI
- Utilizzo IndexFaces con un AWS SDK o una CLI
- Utilizzo ListCollections con un AWS SDK o una CLI
- Utilizzo ListFaces con un AWS SDK o una CLI
- Utilizzo RecognizeCelebrities con un AWS SDK o una CLI
- Utilizzo SearchFaces con un AWS SDK o una CLI
- Utilizzo SearchFacesByImage con un AWS SDK o una CLI

Utilizzo **`CompareFaces`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `CompareFaces`.

Per ulteriori informazioni, consulta Confronto dei volti nelle immagini.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

    /// <summary>
    /// Uses the Amazon Rekognition Service to compare faces in two images.
    /// </summary>
    public class CompareFaces
    {
        public static async Task Main()
        {
            float similarityThreshold = 70F;
            string sourceImage = "source.jpg";
            string targetImage = "target.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            Amazon.Rekognition.Model.Image imageSource = new
Amazon.Rekognition.Model.Image();

            try
            {
                using FileStream fs = new FileStream(sourceImage, FileMode.Open,
FileAccess.Read);
                byte[] data = new byte[fs.Length];
```

```
                fs.Read(data, 0, (int)fs.Length);
                imageSource.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine($"Failed to load source image: {sourceImage}");
                return;
            }

            Amazon.Rekognition.Model.Image imageTarget = new
Amazon.Rekognition.Model.Image();

            try
            {
                using FileStream fs = new FileStream(targetImage, FileMode.Open,
FileAccess.Read);
                byte[] data = new byte[fs.Length];
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageTarget.Bytes = new MemoryStream(data);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Failed to load target image: {targetImage}");
                Console.WriteLine(ex.Message);
                return;
            }

            var compareFacesRequest = new CompareFacesRequest
            {
                SourceImage = imageSource,
                TargetImage = imageTarget,
                SimilarityThreshold = similarityThreshold,
            };

            // Call operation
            var compareFacesResponse = await
rekognitionClient.CompareFacesAsync(compareFacesRequest);

            // Display results
            compareFacesResponse.FaceMatches.ForEach(match =>
            {
                ComparedFace face = match.Face;
                BoundingBox position = face.BoundingBox;
```

```
                Console.WriteLine($"Face at {position.Left} {position.Top}
matches with {match.Similarity}% confidence.");
            });

            Console.WriteLine($"Found {compareFacesResponse.UnmatchedFaces.Count}
face(s) that did not match.");
        }
    }
```

- Per i dettagli sull'API, CompareFacesconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per confrontare i volti in due immagini

Il `compare-faces` comando seguente confronta i volti in due immagini archiviate in un bucket Amazon S3.

```
aws rekognition compare-faces \
    --source-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"source.jpg"}}' \
    --target-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"target.jpg"}}'
```

Output:

```
{
    "UnmatchedFaces": [],
    "FaceMatches": [
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.12368916720151901,
                    "Top": 0.16007372736930847,
                    "Left": 0.5901257991790771,
                    "Height": 0.25140416622161865
                },
                "Confidence": 100.0,
```

```
                "Pose": {
                    "Yaw": -3.7351467609405518,
                    "Roll": -0.10309021919965744,
                    "Pitch": 0.8637830018997192
                },
                "Quality": {
                    "Sharpness": 95.51618957519531,
                    "Brightness": 65.29893493652344
                },
                "Landmarks": [
                    {
                        "Y": 0.26721030473709106,
                        "X": 0.6204193830490112,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.26831310987472534,
                        "X": 0.6776827573776245,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.3514654338359833,
                        "X": 0.6241428852081299,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.35258132219314575,
                        "X": 0.6713621020317078,
                        "Type": "mouthRight"
                    },
                    {
                        "Y": 0.3140771687030792,
                        "X": 0.6428444981575012,
                        "Type": "nose"
                    }
                ]
            },
            "Similarity": 100.0
        }
    ],
    "SourceImageFace": {
        "BoundingBox": {
            "Width": 0.12368916720151901,
            "Top": 0.16007372736930847,
```

```
            "Left": 0.5901257991790771,
            "Height": 0.25140416622161865
        },
        "Confidence": 100.0
    }
}
```

Per ulteriori informazioni, consulta Comparazione dei volti nelle immagini nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta Command CompareFacesReference AWS CLI .

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

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

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

```
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <sourceKey> <targetKey>

            Where:
                bucketName - The name of the S3 bucket where the images are
 stored.
                sourceKey  - The S3 key (file name) for the source image.
                targetKey  - The S3 key (file name) for the target image.
           """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceKey = args[1];
        String targetKey = args[2];

        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
        compareTwoFaces(rekClient, bucketName, sourceKey, targetKey);
    }

   /**
    * Compares two faces from images stored in an Amazon S3 bucket using AWS
 Rekognition.
    *
    * <p>This method takes two image keys from an S3 bucket and compares the
 faces within them.
    * It prints out the confidence level of matched faces and reports the number
 of unmatched faces.</p>
    *
    * @param rekClient   The {@link RekognitionClient} used to call AWS
 Rekognition.
    * @param bucketName  The name of the S3 bucket containing the images.
    * @param sourceKey   The object key (file path) for the source image in the
 S3 bucket.
    * @param targetKey   The object key (file path) for the target image in the
 S3 bucket.
```

```
     * @throws RuntimeException If the Rekognition service returns an error.
     */
    public static void compareTwoFaces(RekognitionClient rekClient, String
bucketName, String sourceKey, String targetKey) {
        try {
            Float similarityThreshold = 70F;
            S3Object s3ObjectSource = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceKey)
                    .build();

            Image sourceImage = Image.builder()
                    .s3Object(s3ObjectSource)
                    .build();

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

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

            CompareFacesRequest facesRequest = CompareFacesRequest.builder()
                    .sourceImage(sourceImage)
                    .targetImage(targetImage)
                    .similarityThreshold(similarityThreshold)
                    .build();

            // Compare the two images.
            CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
            List<CompareFacesMatch> faceDetails =
compareFacesResult.faceMatches();

            for (CompareFacesMatch match : faceDetails) {
                ComparedFace face = match.face();
                BoundingBox position = face.boundingBox();
                System.out.println("Face at " + position.left().toString()
                        + " " + position.top()
                        + " matches with " + face.confidence().toString()
                        + "% confidence.");
            }
```

```
            List<ComparedFace> unmatchedFaces =
 compareFacesResult.unmatchedFaces();
            System.out.println("There were " + unmatchedFaces.size() + " face(s)
 that did not match.");

        } catch (RekognitionException e) {
            System.err.println("Error comparing faces: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Per i dettagli sull'API, CompareFacesconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun compareTwoFaces(
    similarityThresholdVal: Float,
    sourceImageVal: String,
    targetImageVal: String,
) {
    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage =
        Image {
            bytes = sourceBytes
        }

    val tarImage =
```

```
        Image {
            bytes = targetBytes
        }

    val facesRequest =
        CompareFacesRequest {
            sourceImage = souImage
            targetImage = tarImage
            similarityThreshold = similarityThresholdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->

        val compareFacesResult = rekClient.compareFaces(facesRequest)
        val faceDetails = compareFacesResult.faceMatches

        if (faceDetails != null) {
            for (match: CompareFacesMatch in faceDetails) {
                val face = match.face
                val position = face?.boundingBox
                if (position != null) {
                    println("Face at ${position.left} ${position.top} matches
 with ${face.confidence} % confidence.")
                }
            }
        }

        val uncompared = compareFacesResult.unmatchedFaces
        if (uncompared != null) {
            println("There was ${uncompared.size} face(s) that did not match")
        }

        println("Source image rotation:
 ${compareFacesResult.sourceImageOrientationCorrection}")
        println("target image rotation:
 ${compareFacesResult.targetImageOrientationCorrection}")
    }
}
```

- Per i dettagli sull'API, CompareFacesconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def compare_faces(self, target_image, similarity):
        """
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value
 greater
                           than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                 reference image. The second element is the list of faces that
 have
                 a similarity value below the specified threshold.
```

```
        """
        try:
            response = self.rekognition_client.compare_faces(
                SourceImage=self.image,
                TargetImage=target_image.image,
                SimilarityThreshold=similarity,
            )
            matches = [
                RekognitionFace(match["Face"]) for match in
response["FaceMatches"]
            ]
            unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
            logger.info(
                "Found %s matched faces and %s unmatched faces.",
                len(matches),
                len(unmatches),
            )
        except ClientError:
            logger.exception(
                "Couldn't match faces from %s to %s.",
                self.image_name,
                target_image.image_name,
            )
            raise
        else:
            return matches, unmatches
```

- Per i dettagli sull'API, consulta CompareFaces AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`CreateCollection`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `CreateCollection`.

Per ulteriori informazioni, consulta Creazione di una raccolta.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses Amazon Rekognition to create a collection to which you can add
/// faces using the IndexFaces operation.
/// </summary>
public class CreateCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine("Creating collection: " + collectionId);

        var createCollectionRequest = new CreateCollectionRequest
        {
            CollectionId = collectionId,
        };

        CreateCollectionResponse createCollectionResponse = await
rekognitionClient.CreateCollectionAsync(createCollectionRequest);
        Console.WriteLine($"CollectionArn :
{createCollectionResponse.CollectionArn}");
        Console.WriteLine($"Status code :
{createCollectionResponse.StatusCode}");
    }
}
```

- Per i dettagli sull'API, CreateCollectionconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per creare una collezione

Il `create-collection` comando seguente crea una raccolta con il nome specificato.

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

Output:

```
{
    "CollectionArn": "aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
    "FaceModelVersion": "4.0",
    "StatusCode": 200
}
```

Per ulteriori informazioni, consulta Creating a Collection nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta Command CreateCollectionReference AWS CLI .

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

            Usage: <collectionName>\s

            Where:
                collectionName - The name of the collection.\s
            """;

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

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

        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Creates a new Amazon Rekognition collection.
     *
```

```
     * @param rekClient    the Amazon Rekognition client used to interact with
 the Rekognition service
     * @param collectionId the unique identifier for the collection to be created
     */
    public static void createMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            CreateCollectionRequest collectionRequest =
 CreateCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            CreateCollectionResponse collectionResponse =
 rekClient.createCollection(collectionRequest);
            System.out.println("CollectionArn: " +
 collectionResponse.collectionArn());
            System.out.println("Status code: " +
 collectionResponse.statusCode().toString());

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

- Per i dettagli sull'API, CreateCollectionconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
```

```
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- Per i dettagli sull'API, CreateCollectionconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
 API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def create_collection(self, collection_id):
        """
        Creates an empty collection.
```

```
        :param collection_id: Text that identifies the collection.
        :return: The newly created collection.
        """
        try:
            response = self.rekognition_client.create_collection(
                CollectionId=collection_id
            )
            response["CollectionId"] = collection_id
            collection = RekognitionCollection(response, self.rekognition_client)
            logger.info("Created collection %s.", collection_id)
        except ClientError:
            logger.exception("Couldn't create collection %s.", collection_id)
            raise
        else:
            return collection
```

- Per i dettagli sull'API, consulta CreateCollection AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DeleteCollection`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DeleteCollection`.

Per ulteriori informazioni, consulta Eliminazione di una raccolta.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to delete an existing collection.
    /// </summary>
    public class DeleteCollection
    {
        public static async Task Main()
        {
            var rekognitionClient = new AmazonRekognitionClient();

            string collectionId = "MyCollection";
            Console.WriteLine("Deleting collection: " + collectionId);

            var deleteCollectionRequest = new DeleteCollectionRequest()
            {
                CollectionId = collectionId,
            };

            var deleteCollectionResponse = await
rekognitionClient.DeleteCollectionAsync(deleteCollectionRequest);
            Console.WriteLine($"{collectionId}:
{deleteCollectionResponse.StatusCode}");
        }
    }
```

- Per i dettagli sull'API, DeleteCollectionconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per eliminare una raccolta

Il `delete-collection` comando seguente elimina la raccolta specificata.

```
aws rekognition delete-collection \
```

```
    --collection-id MyCollection
```

Output:

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

Per ulteriori informazioni, consulta Eliminazione di una raccolta nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta DeleteCollectionCommand Reference.AWS CLI

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

```
        final String usage = """
            Usage: <collectionId>\s

            Where:
                collectionId - The id of the collection to delete.\s
            """;

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

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

        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Deletes an Amazon Rekognition collection.
     *
     * @param rekClient      An instance of the {@link RekognitionClient} class,
which is used to interact with the Amazon Rekognition service.
     * @param collectionId   The ID of the collection to be deleted.
     */
    public static void deleteMyCollection(RekognitionClient rekClient, String
collectionId) {
        try {
            DeleteCollectionRequest deleteCollectionRequest =
DeleteCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            DeleteCollectionResponse deleteCollectionResponse =
rekClient.deleteCollection(deleteCollectionRequest);
            System.out.println(collectionId + ": " +
deleteCollectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
```

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

- Per i dettagli sull'API, DeleteCollectionconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- Per i dettagli sull'API, DeleteCollectionconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def delete_collection(self):
        """
        Deletes the collection.
        """
        try:

self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
            logger.exception("Couldn't delete collection %s.",
self.collection_id)
            raise
```

- Per i dettagli sull'API, consulta DeleteCollection AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DeleteFaces`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DeleteFaces`.

Per ulteriori informazioni, consulta Eliminazione dei volti da una raccolta.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to delete one or more faces from
    /// a Rekognition collection.
    /// </summary>
    public class DeleteFaces
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection";
            var faces = new List<string> { "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx" };

            var rekognitionClient = new AmazonRekognitionClient();

            var deleteFacesRequest = new DeleteFacesRequest()
            {
                CollectionId = collectionId,
                FaceIds = faces,
            };

            DeleteFacesResponse deleteFacesResponse = await
 rekognitionClient.DeleteFacesAsync(deleteFacesRequest);
            deleteFacesResponse.DeletedFaces.ForEach(face =>
            {
                Console.WriteLine($"FaceID: {face}");
            });
```

```
        }
    }
```

- Per i dettagli sull'API, DeleteFacesconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per eliminare volti da una raccolta

Il `delete-faces` comando seguente elimina la faccia specificata da una raccolta.

```
aws rekognition delete-faces \
    --collection-id MyCollection
    --face-ids '["0040279c-0178-436e-b70a-e61b074e96b0"]'
```

Output:

```
{
    "DeletedFaces": [
        "0040279c-0178-436e-b70a-e61b074e96b0"
    ]
}
```

Per ulteriori informazioni, consulta Eliminazione di volti da una raccolta nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta DeleteFacesCommand Reference.AWS CLI

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

            Where:
                collectionId - The id of the collection from which faces are
 deleted.\s
                faceId - The id of the face to delete.\s
           """;

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

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }

    /**
```

```
     * Deletes a face from the specified Amazon Rekognition collection.
     *
     * @param rekClient     an instance of the Amazon Rekognition client
     * @param collectionId  the ID of the collection from which the face should
 be deleted
     * @param faceId        the ID of the face to be deleted
     * @throws RekognitionException if an error occurs while deleting the face
     */
    public static void deleteFacesCollection(RekognitionClient rekClient,
            String collectionId,
            String faceId) {

        try {
            DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceIds(faceId)
                    .build();

            rekClient.deleteFaces(deleteFacesRequest);
            System.out.println("The face was deleted from the collection.");

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

- Per i dettagli sull'API, DeleteFacesconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun deleteFacesCollection(
```

```
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
        println("$faceIdVal was deleted from the collection")
    }
}
```

- Per i dettagli sull'API, DeleteFacesconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
```

```
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def delete_faces(self, face_ids):
        """
        Deletes faces from the collection.

        :param face_ids: The list of IDs of faces to delete.
        :return: The list of IDs of faces that were deleted.
        """
        try:
            response = self.rekognition_client.delete_faces(
                CollectionId=self.collection_id, FaceIds=face_ids
            )
            deleted_ids = response["DeletedFaces"]
            logger.info(
                "Deleted %s faces from %s.", len(deleted_ids), self.collection_id
            )
        except ClientError:
            logger.exception("Couldn't delete faces from %s.",
self.collection_id)
            raise
        else:
            return deleted_ids
```

- Per i dettagli sull'API, consulta DeleteFaces AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeCollection`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DescribeCollection`.

Per ulteriori informazioni, consulta Descrizione di una raccolta.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to describe the contents of a
/// collection.
/// </summary>
public class DescribeCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();
```

```
            string collectionId = "MyCollection";
            Console.WriteLine($"Describing collection: {collectionId}");

            var describeCollectionRequest = new DescribeCollectionRequest()
            {
                CollectionId = collectionId,
            };

            var describeCollectionResponse = await
 rekognitionClient.DescribeCollectionAsync(describeCollectionRequest);
            Console.WriteLine($"Collection ARN:
 {describeCollectionResponse.CollectionARN}");
            Console.WriteLine($"Face count:
 {describeCollectionResponse.FaceCount}");
            Console.WriteLine($"Face model version:
 {describeCollectionResponse.FaceModelVersion}");
            Console.WriteLine($"Created:
 {describeCollectionResponse.CreationTimestamp}");
        }
    }
```

- Per i dettagli sull'API, DescribeCollectionconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per descrivere una collezione

L'`describe-collection`esempio seguente visualizza i dettagli sulla raccolta specificata.

```
aws rekognition describe-collection \
    --collection-id MyCollection
```

Output:

```
{
    "FaceCount": 200,
    "CreationTimestamp": 1569444828.274,
    "CollectionARN": "arn:aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
```

```
    "FaceModelVersion": "4.0"
}
```

Per ulteriori informazioni, consulta Descrivere una collezione nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta DescribeCollectionCommand Reference.AWS CLI

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
```

```
            """;

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

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

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    /**
     * Describes an Amazon Rekognition collection.
     *
     * @param rekClient         The Amazon Rekognition client used to make the
request.
     * @param collectionName    The name of the collection to describe.
     *
     * @throws RekognitionException If an error occurs while describing the
collection.
     */
    public static void describeColl(RekognitionClient rekClient, String
collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
DescribeCollectionRequest.builder()
                    .collectionId(collectionName)
                    .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
                    .describeCollection(describeCollectionRequest);
            System.out.println("Collection Arn : " +
describeCollectionResponse.collectionARN());
            System.out.println("Created : " +
describeCollectionResponse.creationTimestamp().toString());

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

```
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, DescribeCollectionconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

- Per i dettagli sull'API, DescribeCollectionconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        """
        try:
            response = self.rekognition_client.describe_collection(
                CollectionId=self.collection_id
            )
            # Work around capitalization of Arn vs. ARN
            response["CollectionArn"] = response.get("CollectionARN")
            (
                self.collection_arn,
                self.face_count,
                self.created,
            ) = self._unpack_collection(response)
            logger.info("Got data for collection %s.", self.collection_id)
        except ClientError:
            logger.exception("Couldn't get data for collection %s.",
self.collection_id)
            raise
        else:
            return self.to_dict()
```

- Per i dettagli sull'API, consulta DescribeCollection AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DetectFaces`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DetectFaces`.

Per ulteriori informazioni, consulta Rilevamento dei volti in un'immagine.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DetectFaces
{
    public static async Task Main()
    {
        string photo = "input.jpg";
        string bucket = "amzn-s3-demo-bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectFacesRequest = new DetectFacesRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },
```

```
                // Attributes can be "ALL" or "DEFAULT".
                // "DEFAULT": BoundingBox, Confidence, Landmarks, Pose, and
 Quality.
                // "ALL": See https://docs.aws.amazon.com/sdkfornet/v3/apidocs/
items/Rekognition/TFaceDetail.html
                Attributes = new List<string>() { "ALL" },
            };

            try
            {
                DetectFacesResponse detectFacesResponse = await
 rekognitionClient.DetectFacesAsync(detectFacesRequest);
                bool hasAll = detectFacesRequest.Attributes.Contains("ALL");
                foreach (FaceDetail face in detectFacesResponse.FaceDetails)
                {
                    Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
 left={face.BoundingBox.Top} width={face.BoundingBox.Width}
 height={face.BoundingBox.Height}");
                    Console.WriteLine($"Confidence: {face.Confidence}");
                    Console.WriteLine($"Landmarks: {face.Landmarks.Count}");
                    Console.WriteLine($"Pose: pitch={face.Pose.Pitch}
 roll={face.Pose.Roll} yaw={face.Pose.Yaw}");
                    Console.WriteLine($"Brightness:
 {face.Quality.Brightness}\tSharpness: {face.Quality.Sharpness}");

                    if (hasAll)
                    {
                        Console.WriteLine($"Estimated age is between
 {face.AgeRange.Low} and {face.AgeRange.High} years old.");
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Visualizza le informazioni del riquadro di delimitazione per tutti i volti di un'immagine.

```
    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}");
        }
    }
```

- Per i dettagli sull'API, DetectFacesconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per rilevare i volti in un'immagine

Il `detect-faces` comando seguente rileva i volti nell'immagine specificata archiviata in un bucket Amazon S3.

```
aws rekognition detect-faces \
    --image '{"S3Object":{"Bucket":"MyImageS3Bucket","Name":"MyFriend.jpg"}}' \
    --attributes "ALL"
```

Output:

```
{
    "FaceDetails": [
        {
            "Confidence": 100.0,
            "Eyeglasses": {
                "Confidence": 98.91107940673828,
                "Value": false
            },
            "Sunglasses": {
                "Confidence": 99.7966537475586,
                "Value": false
            },
            "Gender": {
                "Confidence": 99.56611633300781,
                "Value": "Male"
            },
```

```
            "Landmarks": [
                {
                    "Y": 0.26721030473709106,
                    "X": 0.6204193830490112,
                    "Type": "eyeLeft"
                },
                {
                    "Y": 0.26831310987472534,
                    "X": 0.6776827573776245,
                    "Type": "eyeRight"
                },
                {
                    "Y": 0.3514654338359833,
                    "X": 0.6241428852081299,
                    "Type": "mouthLeft"
                },
                {
                    "Y": 0.35258132219314575,
                    "X": 0.6713621020317078,
                    "Type": "mouthRight"
                },
                {
                    "Y": 0.3140771687030792,
                    "X": 0.6428444981575012,
                    "Type": "nose"
                },
                {
                    "Y": 0.24662546813488007,
                    "X": 0.6001564860343933,
                    "Type": "leftEyeBrowLeft"
                },
                {
                    "Y": 0.24326619505882263,
                    "X": 0.6303644776344299,
                    "Type": "leftEyeBrowRight"
                },
                {
                    "Y": 0.23818562924861908,
                    "X": 0.6146903038024902,
                    "Type": "leftEyeBrowUp"
                },
                {
                    "Y": 0.24373626708984375,
                    "X": 0.6640064716339111,
```

```
                    "Type": "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
            }
        }
    ]
}
```

Per ulteriori informazioni, consulta Detecting Faces in an Image nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta DetectFacesCommand Reference.AWS CLI

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

import java.util.List;

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

```
        final String usage = """

            Usage:   <bucketName> <sourceImage>

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

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

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

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

    /**
     * Detects faces in an image stored in an Amazon S3 bucket using the Amazon
Rekognition service.
     *
     * @param rekClient    The Amazon Rekognition client used to interact with
the Rekognition service.
     * @param bucketName   The name of the Amazon S3 bucket where the source
image is stored.
     * @param sourceImage  The name of the source image file in the Amazon S3
bucket.
     */
    public static void detectFacesinImage(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
```

```
                .build();

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

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

            DetectFacesResponse facesResponse =
 rekClient.detectFaces(facesRequest);
            List<FaceDetail> faceDetails = facesResponse.faceDetails();
            for (FaceDetail face : faceDetails) {
                AgeRange ageRange = face.ageRange();
                System.out.println("The detected face is estimated to be between
 "
                        + ageRange.low().toString() + " and " +
 ageRange.high().toString()
                        + " years old.");

                System.out.println("There is a smile : " +
 face.smile().value().toString());
            }

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

- Per i dettagli sull'API, DetectFacesconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between
 ${ageRange?.low} and ${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}
```

- Per i dettagli sull'API, DetectFacesconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
```

```
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
            raise
        else:
            return faces
```

- Per i dettagli sull'API, consulta DetectFaces AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DetectLabels`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DetectLabels`.

Per ulteriori informazioni, consulta Rilevamento delle etichette in un'immagine.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect labels within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DetectLabels
{
```

```
        public static async Task Main()
        {
            string photo = "del_river_02092020_01.jpg"; // "input.jpg";
            string bucket = "amzn-s3-demo-bucket"; // "bucket";

            var rekognitionClient = new AmazonRekognitionClient();

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

            try
            {
                DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (Label label in detectLabelsResponse.Labels)
                {
                    Console.WriteLine($"Name: {label.Name} Confidence:
{label.Confidence}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Rileva le etichette in un file di immagine archiviato sul tuo computer.

```
   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);
            }
        }
    }
```

- Per i dettagli sull'API, DetectLabelsconsulta AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
//! Detect instances of real-world entities within an image by using Amazon
 Rekognition
/*!
  \param imageBucket: The Amazon Simple Storage Service (Amazon S3) bucket
 containing an image.
  \param imageKey: The Amazon S3 key of an image object.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::Rekognition::detectLabels(const Aws::String &imageBucket,
                                       const Aws::String &imageKey,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::Rekognition::RekognitionClient rekognitionClient(clientConfiguration);

    Aws::Rekognition::Model::DetectLabelsRequest request;
    Aws::Rekognition::Model::S3Object s3Object;
    s3Object.SetBucket(imageBucket);
    s3Object.SetName(imageKey);

    Aws::Rekognition::Model::Image image;
    image.SetS3Object(s3Object);

    request.SetImage(image);

    const Aws::Rekognition::Model::DetectLabelsOutcome outcome =
 rekognitionClient.DetectLabels(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::Rekognition::Model::Label> &labels =
 outcome.GetResult().GetLabels();
        if (labels.empty()) {
            std::cout << "No labels detected" << std::endl;
        } else {
            for (const Aws::Rekognition::Model::Label &label: labels) {
                std::cout << label.GetName() << ": " << label.GetConfidence() <<
 std::endl;
            }
        }
    } else {
        std::cerr << "Error while detecting labels: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, DetectLabelsconsulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Per rilevare un'etichetta in un'immagine

L'`detect-labels`esempio seguente rileva scene e oggetti in un'immagine archiviata in un bucket Amazon S3.

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

Output:

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

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


Per ulteriori informazioni, consulta Detecting Labels in an Image nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta DetectLabelsCommand Reference.AWS CLI

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

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

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

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

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

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

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

    /**
     * Detects the labels in an image stored in an Amazon S3 bucket using the
Amazon Rekognition service.
     *
     * @param rekClient     the Amazon Rekognition client used to make the
detection request
     * @param bucketName    the name of the Amazon S3 bucket where the image is
stored
     * @param sourceImage   the name of the image file to be analyzed
     */
    public static void detectImageLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

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

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

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

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

- Per i dettagli sull'API, DetectLabelsconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }
    val request =
        DetectLabelsRequest {
            image = souImage
            maxLabels = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
```

```
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- Per i dettagli sull'API, DetectLabelsconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_labels(self, max_labels):
        """
```

```
        Detects labels in the image. Labels are objects and people.

        :param max_labels: The maximum number of labels to return.
        :return: The list of labels detected in the image.
        """
        try:
            response = self.rekognition_client.detect_labels(
                Image=self.image, MaxLabels=max_labels
            )
            labels = [RekognitionLabel(label) for label in response["Labels"]]
            logger.info("Found %s labels in %s.", len(labels), self.image_name)
        except ClientError:
            logger.info("Couldn't detect labels in %s.", self.image_name)
            raise
        else:
            return labels
```

- Per i dettagli sull'API, consulta DetectLabels AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DetectModerationLabels`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DetectModerationLabels`.

Per ulteriori informazioni, consulta Rilevamento di immagini non appropriate.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   using System;
   using System.Threading.Tasks;
   using Amazon.Rekognition;
   using Amazon.Rekognition.Model;

   /// <summary>
   /// Uses the Amazon Rekognition Service to detect unsafe content in a
   /// JPEG or PNG format image.
   /// </summary>
   public class DetectModerationLabels
   {
       public static async Task Main(string[] args)
       {
           string photo = "input.jpg";
           string bucket = "amzn-s3-demo-bucket";

           var rekognitionClient = new AmazonRekognitionClient();

           var detectModerationLabelsRequest = new
DetectModerationLabelsRequest()
           {
               Image = new Image()
               {
                   S3Object = new S3Object()
                   {
                       Name = photo,
                       Bucket = bucket,
                   },
               },
               MinConfidence = 60F,
           };

           try
           {
               var detectModerationLabelsResponse = await
rekognitionClient.DetectModerationLabelsAsync(detectModerationLabelsRequest);
               Console.WriteLine("Detected labels for " + photo);
               foreach (ModerationLabel label in
detectModerationLabelsResponse.ModerationLabels)
               {
                   Console.WriteLine($"Label: {label.Name}");
                   Console.WriteLine($"Confidence: {label.Confidence}");
                   Console.WriteLine($"Parent: {label.ParentName}");
```

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

- Per i dettagli sull'API, DetectModerationLabelsconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per rilevare contenuti non sicuri in un'immagine

Il `detect-moderation-labels` comando seguente rileva contenuti non sicuri nell'immagine specificata archiviata in un bucket Amazon S3.

```
aws rekognition detect-moderation-labels \
    --image "S3Object={Bucket=MyImageS3Bucket,Name=gun.jpg}"
```

Output:

```
{
    "ModerationModelVersion": "3.0",
    "ModerationLabels": [
        {
            "Confidence": 97.29618072509766,
            "ParentName": "Violence",
            "Name": "Weapon Violence"
        },
        {
            "Confidence": 97.29618072509766,
            "ParentName": "",
            "Name": "Violence"
        }
    ]
}
```

Per ulteriori informazioni, consulta Rilevamento di immagini non sicure nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta Command Reference. DetectModerationLabelsAWS CLI

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

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

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

    public static void main(String[] args) {
        final String usage = """
            Usage:  <bucketName>  <sourceImage>

            Where:
                bucketName - The name of the S3 bucket where the images are
 stored.
```

```
                sourceImage - The name of the image (for example, pic1.png).\s
            """;

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

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

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

    /**
     * Detects moderation labels in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient      the Amazon Rekognition client to use for the
detection
     * @param bucketName     the name of the Amazon S3 bucket where the image is
stored
     * @param sourceImage    the name of the image file to be analyzed
     *
     * @throws RekognitionException if there is an error during the image
detection process
     */
    public static void detectModLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();
```

```
            DetectModerationLabelsRequest moderationLabelsRequest =
 DetectModerationLabelsRequest.builder()
                    .image(targetImage)
                    .minConfidence(60F)
                    .build();

            DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                    .detectModerationLabels(moderationLabelsRequest);
            List<ModerationLabel> labels =
 moderationLabelsResponse.moderationLabels();
            System.out.println("Detected labels for image");
            for (ModerationLabel label : labels) {
                System.out.println("Label: " + label.name()
                        + "\n Confidence: " + label.confidence().toString() + "%"
                        + "\n Parent:" + label.parentName());
            }

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

- Per i dettagli sull'API, DetectModerationLabelsconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
```

```
            this.bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} %
 Parent: ${label.parentName}")
        }
    }
}
```

- Per i dettagli sull'API, DetectModerationLabelsconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
```

```
                    an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify
content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image
            )
            labels = [
                RekognitionModerationLabel(label)
                for label in response["ModerationLabels"]
            ]
            logger.info(
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels
```

- Per i dettagli sull'API, consulta DetectModerationLabels AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DetectText`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DetectText`.

Per ulteriori informazioni, consulta Rilevamento del testo in un'immagine.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect text in an image. The
/// example was created using the AWS SDK for .NET version 3.7 and .NET
/// Core 5.0.
/// </summary>
public class DetectText
{
    public static async Task Main()
    {
        string photo = "Dad_photographer.jpg"; // "input.jpg";
        string bucket = "amzn-s3-demo-bucket"; // "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectTextRequest = new DetectTextRequest()
        {
            Image = new Image()
```

```
                {
                    S3Object = new S3Object()
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },
            };

            try
            {
                DetectTextResponse detectTextResponse = await
rekognitionClient.DetectTextAsync(detectTextRequest);
                Console.WriteLine($"Detected lines and words for {photo}");
                detectTextResponse.TextDetections.ForEach(text =>
                {
                    Console.WriteLine($"Detected: {text.DetectedText}");
                    Console.WriteLine($"Confidence: {text.Confidence}");
                    Console.WriteLine($"Id : {text.Id}");
                    Console.WriteLine($"Parent Id: {text.ParentId}");
                    Console.WriteLine($"Type: {text.Type}");
                });
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
```

- Per i dettagli sull'API, DetectTextconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per rilevare il testo in un'immagine

Il `detect-text` comando seguente rileva il testo nell'immagine specificata.

```
aws rekognition detect-text \
```

```
    --image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"ExamplePicture.jpg"}}'
```

Output:

```
{
    "TextDetections": [
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.24624845385551453,
                    "Top": 0.28288066387176514,
                    "Left": 0.391388863325119,
                    "Height": 0.022687450051307678
                },
                "Polygon": [
                    {
                        "Y": 0.28288066387176514,
                        "X": 0.391388863325119
                    },
                    {
                        "Y": 0.2826388478279114,
                        "X": 0.6376373171806335
                    },
                    {
                        "Y": 0.30532628297805786,
                        "X": 0.637677013874054
                    },
                    {
                        "Y": 0.305568128824234,
                        "X": 0.39142853021621704
                    }
                ]
            },
            "Confidence": 94.35709381103516,
            "DetectedText": "ESTD 1882",
            "Type": "LINE",
            "Id": 0
        },
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33933889865875244,
```

```
                    "Top": 0.32603850960731506,
                    "Left": 0.34534579515457153,
                    "Height": 0.07126858830451965
                },
                "Polygon": [
                    {
                        "Y": 0.32603850960731506,
                        "X": 0.34534579515457153
                    },
                    {
                        "Y": 0.32633158564567566,
                        "X": 0.684684693813324
                    },
                    {
                        "Y": 0.3976001739501953,
                        "X": 0.684575080871582
                    },
                    {
                        "Y": 0.3973070979118347,
                        "X": 0.345236212015152
                    }
                ]
            },
            "Confidence": 99.95779418945312,
            "DetectedText": "BRAINS",
            "Type": "LINE",
            "Id": 1
        },
        {
            "Confidence": 97.22098541259766,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.061079490929841995,
                    "Top": 0.2843210697174072,
                    "Left": 0.391391396522522,
                    "Height": 0.021029088646173477
                },
                "Polygon": [
                    {
                        "Y": 0.2843210697174072,
                        "X": 0.391391396522522
                    },
                    {
                        "Y": 0.2828207015991211,
```

```
                    "X": 0.4524524509906769
                },
                {
                    "Y": 0.3038259446620941,
                    "X": 0.4534534513950348
                },
                {
                    "Y": 0.30532634258270264,
                    "X": 0.3923923969268799
                }
            ]
        },
        "DetectedText": "ESTD",
        "ParentId": 0,
        "Type": "WORD",
        "Id": 2
    },
    {
        "Confidence": 91.49320983886719,
        "Geometry": {
            "BoundingBox": {
                "Width": 0.07007007300853729,
                "Top": 0.2828207015991211,
                "Left": 0.5675675868988037,
                "Height": 0.02250562608242035
            },
            "Polygon": [
                {
                    "Y": 0.2828207015991211,
                    "X": 0.5675675868988037
                },
                {
                    "Y": 0.2828207015991211,
                    "X": 0.6376376152038574
                },
                {
                    "Y": 0.30532634258270264,
                    "X": 0.6376376152038574
                },
                {
                    "Y": 0.30532634258270264,
                    "X": 0.5675675868988037
                }
            ]
```

```
            },
            "DetectedText": "1882",
            "ParentId": 0,
            "Type": "WORD",
            "Id": 3
        },
        {
            "Confidence": 99.95779418945312,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33933934569358826,
                    "Top": 0.32633158564567566,
                    "Left": 0.3453453481197357,
                    "Height": 0.07127484679222107
                },
                "Polygon": [
                    {
                        "Y": 0.32633158564567566,
                        "X": 0.3453453481197357
                    },
                    {
                        "Y": 0.32633158564567566,
                        "X": 0.684684693813324
                    },
                    {
                        "Y": 0.39759939908981323,
                        "X": 0.6836836934089661
                    },
                    {
                        "Y": 0.39684921503067017,
                        "X": 0.3453453481197357
                    }
                ]
            },
            "DetectedText": "BRAINS",
            "ParentId": 1,
            "Type": "WORD",
            "Id": 4
        }
    ]
}
```


- Per i dettagli sull'API, vedere DetectTextin AWS CLI Command Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:   <bucketName> <sourceImage>\n" +
            "\n" +
            "Where:\n" +
            "   bucketName - The name of the S3 bucket where the image is stored
\n" +
            "   sourceImage - The path to the image that contains text (for
 example, pic1.png). \n";

        if (args.length != 2) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

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

    /**
     * Detects text labels in an image stored in an S3 bucket using Amazon
Rekognition.
     *
     * @param rekClient    an instance of the Amazon Rekognition client
     * @param bucketName   the name of the S3 bucket where the image is stored
     * @param sourceImage  the name of the image file in the S3 bucket
     * @throws RekognitionException if an error occurs while calling the Amazon
Rekognition API
     */
    public static void detectTextLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

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

            DetectTextRequest textRequest = DetectTextRequest.builder()
                    .image(souImage)
                    .build();

            DetectTextResponse textResponse = rekClient.detectText(textRequest);
            List<TextDetection> textCollection = textResponse.textDetections();
            System.out.println("Detected lines and words");
            for (TextDetection text : textCollection) {
```

```
                System.out.println("Detected: " + text.detectedText());
                System.out.println("Confidence: " +
 text.confidence().toString());
                System.out.println("Id : " + text.id());
                System.out.println("Parent Id: " + text.parentId());
                System.out.println("Type: " + text.type());
                System.out.println();
            }

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

- Per i dettagli sull'API, DetectTextconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
```

```
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id:  ${text.parentId}")
            println("Type: ${text.type}")
        }
    }
}
```

- Per i dettagli sull'API, DetectTextconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client
```

```
    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in
 response["TextDetections"]]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

- Per i dettagli sull'API, consulta DetectText AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DisassociateFaces`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DisassociateFaces`.

CLI

AWS CLI

```
aws rekognition disassociate-faces --face-ids list-of-face-ids
  --user-id user-id --collection-id collection-name --region region-name
```

- Per i dettagli sull'API, consulta DisassociateFaces AWS CLICommand Reference.

Python

SDK per 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()
```

- Per i dettagli sull'API, consulta DisassociateFaces AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`GetCelebrityInfo`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `GetCelebrityInfo`.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to retrieve information about the
/// celebrity identified by the supplied celebrity Id.
/// </summary>
public class CelebrityInfo
{
    public static async Task Main()
    {
        string celebId = "nnnnnnnn";

        var rekognitionClient = new AmazonRekognitionClient();
```

```
            var celebrityInfoRequest = new GetCelebrityInfoRequest
            {
                Id = celebId,
            };

            Console.WriteLine($"Getting information for celebrity: {celebId}");

            var celebrityInfoResponse = await
 rekognitionClient.GetCelebrityInfoAsync(celebrityInfoRequest);

            // Display celebrity information.
            Console.WriteLine($"celebrity name: {celebrityInfoResponse.Name}");
            Console.WriteLine("Further information (if available):");
            celebrityInfoResponse.Urls.ForEach(url =>
            {
                Console.WriteLine(url);
            });
        }
    }
```

- Per i dettagli sull'API, GetCelebrityInfoconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per ottenere informazioni su una celebrità

Il `get-celebrity-info` comando seguente visualizza informazioni sulla celebrità specificata. Il `id` parametro proviene da una precedente chiamata `arecognize-celebrities`.

```
aws rekognition get-celebrity-info --id nnnnnnn
```

Output:

```
{
    "Name": "Celeb A",
```

```
    "Urls": [
        "www.imdb.com/name/aaaaaaaaa"
    ]
}
```

Per ulteriori informazioni, consulta Ottenere informazioni su una celebrità nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta GetCelebrityInfoCommand Reference.AWS CLI

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUsare Rekognition con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`IndexFaces`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `IndexFaces`.

Per ulteriori informazioni, consulta Indicizzazione dei volti in una raccolta.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces in an image
/// that has been uploaded to an Amazon Simple Storage Service (Amazon S3)
/// bucket and then adds the information to a collection.
/// </summary>
```

```
    public class AddFaces
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection2";
            string bucket = "amzn-s3-demo-bucket";
            string photo = "input.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            var image = new Image
            {
                S3Object = new S3Object
                {
                    Bucket = bucket,
                    Name = photo,
                },
            };

            var indexFacesRequest = new IndexFacesRequest
            {
                Image = image,
                CollectionId = collectionId,
                ExternalImageId = photo,
                DetectionAttributes = new List<string>() { "ALL" },
            };

            IndexFacesResponse indexFacesResponse = await
rekognitionClient.IndexFacesAsync(indexFacesRequest);

            Console.WriteLine($"{photo} added");
            foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
            {
                Console.WriteLine($"Face detected: Faceid is
{faceRecord.Face.FaceId}");
            }
        }
    }
```

- Per i dettagli sull'API, IndexFacesconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per aggiungere volti a una collezione

Il `index-faces` comando seguente aggiunge le facce trovate in un'immagine alla raccolta specificata.

```
aws rekognition index-faces \
    --image '{"S3Object":{"Bucket":"MyVideoS3Bucket","Name":"MyPicture.jpg"}}' \
    --collection-id MyCollection \
    --max-faces 1 \
    --quality-filter "AUTO" \
    --detection-attributes "ALL" \
    --external-image-id "MyPicture.jpg"
```

Output:

```
{
    "FaceRecords": [
        {
            "FaceDetail": {
                "Confidence": 99.993408203125,
                "Eyeglasses": {
                    "Confidence": 99.11750030517578,
                    "Value": false
                },
                "Sunglasses": {
                    "Confidence": 99.98249053955078,
                    "Value": false
                },
                "Gender": {
                    "Confidence": 99.92769622802734,
                    "Value": "Male"
                },
                "Landmarks": [
                    {
                        "Y": 0.26750367879867554,
                        "X": 0.6202793717384338,
                        "Type": "eyeLeft"
                    },
                    {
```

```
                        "Y": 0.26642778515815735,
                        "X": 0.6787431836128235,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.31361380219459534,
                        "X": 0.6421601176261902,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.3495299220085144,
                        "X": 0.6216195225715637,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.35194727778434753,
                        "X": 0.669899046421051,
                        "Type": "mouthRight"
                    },
                    {
                        "Y": 0.26844894886016846,
                        "X": 0.6210268139839172,
                        "Type": "leftPupil"
                    },
                    {
                        "Y": 0.26707562804222107,
                        "X": 0.6817160844802856,
                        "Type": "rightPupil"
                    },
                    {
                        "Y": 0.24834522604942322,
                        "X": 0.6018546223640442,
                        "Type": "leftEyeBrowLeft"
                    },
                    {
                        "Y": 0.24397172033786774,
                        "X": 0.6172008514404297,
                        "Type": "leftEyeBrowUp"
                    },
                    {
                        "Y": 0.24677404761314392,
                        "X": 0.6339119076728821,
                        "Type": "leftEyeBrowRight"
                    },
```

```
                    {
                        "Y": 0.24582654237747192,
                        "X": 0.6619398593902588,
                        "Type": "rightEyeBrowLeft"
                    },
                    {
                        "Y": 0.23973053693771362,
                        "X": 0.6804757118225098,
                        "Type": "rightEyeBrowUp"
                    },
                    {
                        "Y": 0.24441994726657867,
                        "X": 0.6978968977928162,
                        "Type": "rightEyeBrowRight"
                    },
                    {
                        "Y": 0.2695908546447754,
                        "X": 0.6085202693939209,
                        "Type": "leftEyeLeft"
                    },
                    {
                        "Y": 0.26716896891593933,
                        "X": 0.6315826177597046,
                        "Type": "leftEyeRight"
                    },
                    {
                        "Y": 0.26289820671081543,
                        "X": 0.6202316880226135,
                        "Type": "leftEyeUp"
                    },
                    {
                        "Y": 0.27123287320137024,
                        "X": 0.6205548048019409,
                        "Type": "leftEyeDown"
                    },
                    {
                        "Y": 0.2668408751487732,
                        "X": 0.6663622260093689,
                        "Type": "rightEyeLeft"
                    },
                    {
                        "Y": 0.26741549372673035,
                        "X": 0.6910083889961243,
                        "Type": "rightEyeRight"
```

```
                    },
                    {
                        "Y": 0.2614026665687561,
                        "X": 0.6785826086997986,
                        "Type": "rightEyeUp"
                    },
                    {
                        "Y": 0.27075251936912537,
                        "X": 0.6789616942405701,
                        "Type": "rightEyeDown"
                    },
                    {
                        "Y": 0.3211299479007721,
                        "X": 0.6324167847633362,
                        "Type": "noseLeft"
                    },
                    {
                        "Y": 0.32276326417922974,
                        "X": 0.6558475494384766,
                        "Type": "noseRight"
                    },
                    {
                        "Y": 0.34385165572166443,
                        "X": 0.6444970965385437,
                        "Type": "mouthUp"
                    },
                    {
                        "Y": 0.3671635091304779,
                        "X": 0.6459195017814636,
                        "Type": "mouthDown"
                    }
                ],
                "Pose": {
                    "Yaw": -9.54541015625,
                    "Roll": -0.5709401965141296,
                    "Pitch": 0.6045494675636292
                },
                "Emotions": [
                    {
                        "Confidence": 39.90074157714844,
                        "Type": "HAPPY"
                    },
                    {
                        "Confidence": 23.38753890991211,
```

```
                        "Type": "CALM"
                    },
                    {
                        "Confidence": 5.840933322906494,
                        "Type": "CONFUSED"
                    }
                ],
                "AgeRange": {
                    "High": 63,
                    "Low": 45
                },
                "EyesOpen": {
                    "Confidence": 99.80887603759766,
                    "Value": true
                },
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618015021085739,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770642817020416
                },
                "Smile": {
                    "Confidence": 99.69740295410156,
                    "Value": false
                },
                "MouthOpen": {
                    "Confidence": 99.97393798828125,
                    "Value": false
                },
                "Quality": {
                    "Sharpness": 95.54405975341797,
                    "Brightness": 63.867706298828125
                },
                "Mustache": {
                    "Confidence": 97.05007934570312,
                    "Value": false
                },
                "Beard": {
                    "Confidence": 87.34505462646484,
                    "Value": false
                }
            },
            "Face": {
                "BoundingBox": {
```

```
                    "Width": 0.18562500178813934,
                    "Top": 0.1618015021085739,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770642817020416
                },
                "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
                "ExternalImageId": "example-image.jpg",
                "Confidence": 99.993408203125,
                "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
            }
        }
    ],
    "UnindexedFaces": [],
    "FaceModelVersion": "3.0",
    "OrientationCorrection": "ROTATE_0"
}
```

Per ulteriori informazioni, consulta Adding Faces to a Collection nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta Command IndexFacesReference AWS CLI .

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import java.util.List;

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

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

            Where:
                collectionName - The name of the collection.
                sourceImage - The name of the image (for example, pic1.png).
                bucketName - The name of the S3 bucket.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        String bucketName = args[2];;
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        addToCollection(rekClient, collectionId, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Adds a face from an image to an Amazon Rekognition collection.
     *
     * @param rekClient     the Amazon Rekognition client
     * @param collectionId  the ID of the collection to add the face to
     * @param bucketName    the name of the Amazon S3 bucket containing the image
     * @param sourceImage   the name of the image file to add to the collection
     * @throws RekognitionException if there is an error while interacting with
 the Amazon Rekognition service
     */
    public static void addToCollection(RekognitionClient rekClient, String
 collectionId, String bucketName, String sourceImage) {
```

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

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

            IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                    .collectionId(collectionId)
                    .image(targetImage)
                    .maxFaces(1)
                    .qualityFilter(QualityFilter.AUTO)
                    .detectionAttributes(Attribute.DEFAULT)
                    .build();

            IndexFacesResponse facesResponse =
rekClient.indexFaces(facesRequest);
            System.out.println("Results for the image");
            System.out.println("\n Faces indexed:");
            List<FaceRecord> faceRecords = facesResponse.faceRecords();
            for (FaceRecord faceRecord : faceRecords) {
                System.out.println("  Face ID: " + faceRecord.face().faceId());
                System.out.println("  Location:" +
faceRecord.faceDetail().boundingBox().toString());
            }

            List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
            System.out.println("Faces not indexed:");
            for (UnindexedFace unindexedFace : unindexedFaces) {
                System.out.println("  Location:" +
unindexedFace.faceDetail().boundingBox().toString());
                System.out.println("  Reasons:");
                for (Reason reason : unindexedFace.reasons()) {
                    System.out.println("Reason:  " + reason);
                }
            }

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

```
    }
}
```

- Per i dettagli sull'API, IndexFacesconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun addToCollection(
    collectionIdVal: String?,
    sourceImage: String,
) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        IndexFacesRequest {
            collectionId = collectionIdVal
            image = souImage
            maxFaces = 1
            qualityFilter = QualityFilter.Auto
            detectionAttributes = listOf(Attribute.Default)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)

        // Display the results.
        println("Results for the image")
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
            println("Face ID: ${faceRecord.face?.faceId}")
```

```
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
            println("Reasons:")

            unindexedFace.reasons?.forEach { reason ->
                println("Reason:  $reason")
            }
        }
    }
}
```

- Per i dettagli sull'API, IndexFacesconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
```

```
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def index_faces(self, image, max_faces):
        """
        Finds faces in the specified image, indexes them, and stores them in the
        collection.

        :param image: The image to index.
        :param max_faces: The maximum number of faces to index.
        :return: A tuple. The first element is a list of indexed faces.
                 The second element is a list of faces that couldn't be indexed.
        """
        try:
            response = self.rekognition_client.index_faces(
                CollectionId=self.collection_id,
                Image=image.image,
                ExternalImageId=image.image_name,
                MaxFaces=max_faces,
                DetectionAttributes=["ALL"],
            )
            indexed_faces = [
                RekognitionFace({**face["Face"], **face["FaceDetail"]})
                for face in response["FaceRecords"]
```

```
            ]
            unindexed_faces = [
                RekognitionFace(face["FaceDetail"])
                for face in response["UnindexedFaces"]
            ]
            logger.info(
                "Indexed %s faces in %s. Could not index %s faces.",
                len(indexed_faces),
                image.image_name,
                len(unindexed_faces),
            )
        except ClientError:
            logger.exception("Couldn't index faces in image %s.",
image.image_name)
            raise
        else:
            return indexed_faces, unindexed_faces
```

- Per i dettagli sull'API, consulta IndexFaces AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ListCollections`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `ListCollections`.

Per ulteriori informazioni, consulta Creazione dell'elenco delle raccolte.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   using System;
   using System.Threading.Tasks;
   using Amazon.Rekognition;
   using Amazon.Rekognition.Model;

   /// <summary>
   /// Uses Amazon Rekognition to list the collection IDs in the
   /// current account.
   /// </summary>
   public class ListCollections
   {
       public static async Task Main()
       {
           var rekognitionClient = new AmazonRekognitionClient();

           Console.WriteLine("Listing collections");
           int limit = 10;

           var listCollectionsRequest = new ListCollectionsRequest
           {
               MaxResults = limit,
           };

           var listCollectionsResponse = new ListCollectionsResponse();

           do
           {
               if (listCollectionsResponse is not null)
               {
                   listCollectionsRequest.NextToken =
listCollectionsResponse.NextToken;
               }

               listCollectionsResponse = await
rekognitionClient.ListCollectionsAsync(listCollectionsRequest);

               listCollectionsResponse.CollectionIds.ForEach(id =>
               {
                   Console.WriteLine(id);
               });
           }
           while (listCollectionsResponse.NextToken is not null);
       }
```

```
    }
```

- Per i dettagli sull'API, ListCollectionsconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per elencare le raccolte disponibili

Il `list-collections` comando seguente elenca le raccolte disponibili nell' AWS account.

```
aws rekognition list-collections
```

Output:

```
{
    "FaceModelVersions": [
        "2.0",
        "3.0",
        "3.0",
        "3.0",
        "4.0",
        "1.0",
        "3.0",
        "4.0",
        "4.0",
        "4.0"
    ],
    "CollectionIds": [
        "MyCollection1",
        "MyCollection2",
        "MyCollection3",
        "MyCollection4",
        "MyCollection5",
        "MyCollection6",
        "MyCollection7",
        "MyCollection8",
        "MyCollection9",
        "MyCollection10"
```

```
    ]
}
```

Per ulteriori informazioni, consulta Listing Collections nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta Command ListCollectionsReference AWS CLI .

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListCollections {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Listing collections");
```

```
        listAllCollections(rekClient);
        rekClient.close();
    }

    public static void listAllCollections(RekognitionClient rekClient) {
        try {
            ListCollectionsRequest listCollectionsRequest =
 ListCollectionsRequest.builder()
                    .maxResults(10)
                    .build();

            ListCollectionsResponse response =
 rekClient.listCollections(listCollectionsRequest);
            List<String> collectionIds = response.collectionIds();
            for (String resultId : collectionIds) {
                System.out.println(resultId);
            }

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

- Per i dettagli sull'API, ListCollectionsconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
            maxResults = 10
```

```
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- Per i dettagli sull'API, ListCollectionsconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
 API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def list_collections(self, max_results):
        """
        Lists collections for the current account.
```

```
        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
            response =
self.rekognition_client.list_collections(MaxResults=max_results)
            collections = [
                RekognitionCollection({"CollectionId": col_id},
self.rekognition_client)
                for col_id in response["CollectionIds"]
            ]
        except ClientError:
            logger.exception("Couldn't list collections.")
            raise
        else:
            return collections
```

- Per i dettagli sull'API, consulta ListCollections AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ListFaces`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `ListFaces`.

Per ulteriori informazioni, consulta Creazione dell'elenco dei volti in una raccolta.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to retrieve the list of faces
    /// stored in a collection.
    /// </summary>
    public class ListFaces
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection2";

            var rekognitionClient = new AmazonRekognitionClient();

            var listFacesResponse = new ListFacesResponse();
            Console.WriteLine($"Faces in collection {collectionId}");

            var listFacesRequest = new ListFacesRequest
            {
                CollectionId = collectionId,
                MaxResults = 1,
            };

            do
            {
                listFacesResponse = await
rekognitionClient.ListFacesAsync(listFacesRequest);
                listFacesResponse.Faces.ForEach(face =>
                {
                    Console.WriteLine(face.FaceId);
                });

                listFacesRequest.NextToken = listFacesResponse.NextToken;
            }
            while (!string.IsNullOrEmpty(listFacesResponse.NextToken));
        }
    }
```

- Per i dettagli sull'API, ListFacesconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per elencare i volti di una raccolta

Il `list-faces` comando seguente elenca le facce della raccolta specificata.

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

Output:

```
{
    "FaceModelVersion": "3.0",
    "Faces": [
        {
            "BoundingBox": {
                "Width": 0.5216310024261475,
                "Top": 0.3256250023841858,
                "Left": 0.13394300639629364,
                "Height": 0.3918749988079071
            },
            "FaceId": "0040279c-0178-436e-b70a-e61b074e96b0",
            "ExternalImageId": "image1.jpg",
            "Confidence": 100.0,
            "ImageId": "f976e487-3719-5e2d-be8b-ea2724c26991"
        },
        {
            "BoundingBox": {
                "Width": 0.5074880123138428,
                "Top": 0.3774999976158142,
                "Left": 0.18302799761295319,
                "Height": 0.3812499940395355
            },
            "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
            "ExternalImageId": "image2.jpg",
            "Confidence": 99.99930572509766,
            "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
        },
        {
```

```
            "BoundingBox": {
                "Width": 0.5574039816856384,
                "Top": 0.37187498807907104,
                "Left": 0.14559100568294525,
                "Height": 0.4181250035762787
            },
            "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
            "ExternalImageId": "image3.jpg",
            "Confidence": 99.99960327148438,
            "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
        },
        {
            "BoundingBox": {
                "Width": 0.18562500178813934,
                "Top": 0.1618019938468933,
                "Left": 0.5575000047683716,
                "Height": 0.24770599603652954
            },
            "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
            "ExternalImageId": "image4.jpg",
            "Confidence": 99.99340057373047,
            "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
        },
        {
            "BoundingBox": {
                "Width": 0.5307819843292236,
                "Top": 0.2862499952316284,
                "Left": 0.1564060002565384,
                "Height": 0.3987500071525574
            },
            "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
            "ExternalImageId": "image5.jpg",
            "Confidence": 99.99970245361328,
            "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
        },
        {
            "BoundingBox": {
                "Width": 0.5773710012435913,
                "Top": 0.34437501430511475,
                "Left": 0.12396000325679779,
                "Height": 0.4337500035762787
            },
            "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
            "ExternalImageId": "image6.jpg",
```

```
            "Confidence": 100.0,
            "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
        },
        {
            "BoundingBox": {
                "Width": 0.5349419713020325,
                "Top": 0.29124999046325684,
                "Left": 0.16389399766921997,
                "Height": 0.40187498927116394
            },
            "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
            "ExternalImageId": "image7.jpg",
            "Confidence": 99.99979400634766,
            "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
        },
        {
            "BoundingBox": {
                "Width": 0.41499999165534973,
                "Top": 0.09187500178813934,
                "Left": 0.28083300590515137,
                "Height": 0.3112500011920929
            },
            "FaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
            "ExternalImageId": "image8.jpg",
            "Confidence": 99.99769592285156,
            "ImageId": "a294da46-2cb1-5cc4-9045-61d7ca567662"
        },
        {
            "BoundingBox": {
                "Width": 0.48166701197624207,
                "Top": 0.20999999344348907,
                "Left": 0.21250000596046448,
                "Height": 0.36125001311302185
            },
            "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
            "ExternalImageId": "image9.jpg",
            "Confidence": 99.99949645996094,
            "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
        },
        {
            "BoundingBox": {
                "Width": 0.18562500178813934,
                "Top": 0.1618019938468933,
                "Left": 0.5575000047683716,
```

```
                "Height": 0.24770599603652954
            },
            "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
            "ExternalImageId": "image10.jpg",
            "Confidence": 99.99340057373047,
            "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
        }
    ]
}
```

Per ulteriori informazioni, consulta Listing Faces in a Collection nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta Command ListFacesReference AWS CLI .

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

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

```
public class ListFacesInCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId>

                Where:
                   collectionId - The name of the collection.\s
                """;

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

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

        System.out.println("Faces in collection " + collectionId);
        listFacesCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void listFacesCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            ListFacesRequest facesRequest = ListFacesRequest.builder()
                    .collectionId(collectionId)
                    .maxResults(10)
                    .build();

            ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
            List<Face> faces = facesResponse.faces();
            for (Face face : faces) {
                System.out.println("Confidence level there is a face: " +
 face.confidence());
                System.out.println("The face Id value is " + face.faceId());
            }

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

```
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, ListFacesconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listFaces(request)
        response.faces?.forEach { face ->
            println("Confidence level there is a face: ${face.confidence}")
            println("The face Id value is ${face.faceId}")
        }
    }
}
```

- Per i dettagli sull'API, ListFacesconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def list_faces(self, max_results):
        """
        Lists the faces currently indexed in the collection.

        :param max_results: The maximum number of faces to return.
        :return: The list of faces in the collection.
        """
        try:
            response = self.rekognition_client.list_faces(
                CollectionId=self.collection_id, MaxResults=max_results
            )
            faces = [RekognitionFace(face) for face in response["Faces"]]
            logger.info(
                "Found %s faces in collection %s.", len(faces),
self.collection_id
            )
        except ClientError:
            logger.exception(
                "Couldn't list faces in collection %s.", self.collection_id
            )
            raise
        else:
            return faces
```

- Per i dettagli sull'API, consulta ListFaces AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`RecognizeCelebrities`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `RecognizeCelebrities`.

Per ulteriori informazioni, consulta Riconoscimento delle celebrità in un'immagine.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

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

    /// <summary>
    /// Shows how to use Amazon Rekognition to identify celebrities in a photo.
    /// </summary>
    public class CelebritiesInImage
    {
        public static async Task Main(string[] args)
        {
            string photo = "moviestars.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            var recognizeCelebritiesRequest = new RecognizeCelebritiesRequest();

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

```
            }

            img.Bytes = new MemoryStream(data);
            recognizeCelebritiesRequest.Image = img;

            Console.WriteLine($"Looking for celebrities in image {photo}\n");

            var recognizeCelebritiesResponse = await
rekognitionClient.RecognizeCelebritiesAsync(recognizeCelebritiesRequest);


Console.WriteLine($"{recognizeCelebritiesResponse.CelebrityFaces.Count}
celebrity(s) were recognized.\n");
            recognizeCelebritiesResponse.CelebrityFaces.ForEach(celeb =>
            {
                Console.WriteLine($"Celebrity recognized: {celeb.Name}");
                Console.WriteLine($"Celebrity ID: {celeb.Id}");
                BoundingBox boundingBox = celeb.Face.BoundingBox;
                Console.WriteLine($"position: {boundingBox.Left}
{boundingBox.Top}");
                Console.WriteLine("Further information (if available):");
                celeb.Urls.ForEach(url =>
                {
                    Console.WriteLine(url);
                });
            });


Console.WriteLine($"{recognizeCelebritiesResponse.UnrecognizedFaces.Count}
face(s) were unrecognized.");
        }
    }
```

- Per i dettagli sull'API, RecognizeCelebritiesconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per riconoscere le celebrità in un'immagine

Il `recognize-celebrities` comando seguente riconosce le celebrità nell'immagine specificata archiviata in un bucket Amazon S3. :

```
aws rekognition recognize-celebrities \
    --image "S3Object={Bucket=MyImageS3Bucket,Name=moviestars.jpg}"
```

Output:

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

Per ulteriori informazioni, consulta Riconoscere le celebrità in un'immagine nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta Command Reference. RecognizeCelebritiesAWS CLI

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

import software.amazon.awssdk.services.rekognition.model.*;

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

```
 */
public class RecognizeCelebrities {
    public static void main(String[] args) {
        final String usage = """
                Usage:   <bucketName> <sourceImage>

                Where:
                   bucketName - The name of the S3 bucket where the images are
 stored.
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

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

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

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

    /**
     * Recognizes all celebrities in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient    the Amazon Rekognition client used to perform the
 celebrity recognition operation
     * @param bucketName   the name of the Amazon S3 bucket where the source
 image is stored
     * @param sourceImage  the name of the source image file stored in the Amazon
 S3 bucket
     */
    public static void recognizeAllCelebrities(RekognitionClient rekClient,
 String bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
```

```
                .bucket(bucketName)
                .name(sourceImage)
                .build();

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

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

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

                System.out.println("Further information (if available):");
                for (String url : celebrity.urls()) {
                    System.out.println(url);
                }
                System.out.println();
            }
            System.out.println(result.unrecognizedFaces().size() + " face(s) were
 unrecognized.");

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

- Per i dettagli sull'API, RecognizeCelebritiesconsulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
        println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
    }
}
```

- Per i dettagli sull'API, RecognizeCelebritiesconsulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def recognize_celebrities(self):
        """
        Detects celebrities in the image.

        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
        try:
            response =
 self.rekognition_client.recognize_celebrities(Image=self.image)
            celebrities = [
```

```
                RekognitionCelebrity(celeb) for celeb in
response["CelebrityFaces"]
            ]
            other_faces = [
                RekognitionFace(face) for face in response["UnrecognizedFaces"]
            ]
            logger.info(
                "Found %s celebrities and %s other faces in %s.",
                len(celebrities),
                len(other_faces),
                self.image_name,
            )
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.",
self.image_name)
            raise
        else:
            return celebrities, other_faces
```

- Per i dettagli sull'API, consulta RecognizeCelebrities AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`SearchFaces`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `SearchFaces`.

Per ulteriori informazioni, consulta Ricerca di un volto (ID volto).

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to find faces in an image that
/// match the face Id provided in the method request.
/// </summary>
public class SearchFacesMatchingId
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        string faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";

        var rekognitionClient = new AmazonRekognitionClient();

        // Search collection for faces matching the face id.
        var searchFacesRequest = new SearchFacesRequest
        {
            CollectionId = collectionId,
            FaceId = faceId,
            FaceMatchThreshold = 70F,
            MaxFaces = 2,
        };

        SearchFacesResponse searchFacesResponse = await
rekognitionClient.SearchFacesAsync(searchFacesRequest);

        Console.WriteLine("Face matching faceId " + faceId);
```

```
            Console.WriteLine("Matche(s): ");
            searchFacesResponse.FaceMatches.ForEach(face =>
            {
                Console.WriteLine($"FaceId: {face.Face.FaceId} Similarity:
 {face.Similarity}");
            });
        }
    }
```

- Per i dettagli sull'API, SearchFacesconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per cercare volti in una raccolta che corrispondano a un Face ID.

Il `search-faces` comando seguente cerca i volti in una raccolta che corrispondono al face ID specificato.

```
aws rekognition search-faces \
    --face-id 8d3cfc70-4ba8-4b36-9644-90fba29c2dac \
    --collection-id MyCollection
```

Output:

```
{
    "SearchedFaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
    "FaceModelVersion": "3.0",
    "FaceMatches": [
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.48166701197624207,
                    "Top": 0.20999999344348907,
                    "Left": 0.21250000596046448,
                    "Height": 0.36125001311302185
                },
                "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
                "ExternalImageId": "image1.jpg",
```

```
                "Confidence": 99.99949645996094,
                "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
            },
            "Similarity": 99.30997467041016
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618019938468933,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770599603652954
                },
                "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
                "ExternalImageId": "example-image.jpg",
                "Confidence": 99.99340057373047,
                "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
            },
            "Similarity": 99.24862670898438
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618019938468933,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770599603652954
                },
                "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
                "ExternalImageId": "image3.jpg",
                "Confidence": 99.99340057373047,
                "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
            },
            "Similarity": 99.24862670898438
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5349419713020325,
                    "Top": 0.29124999046325684,
                    "Left": 0.16389399766921997,
                    "Height": 0.40187498927116394
                },
                "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
```

```
                "ExternalImageId": "image9.jpg",
                "Confidence": 99.99979400634766,
                "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
            },
            "Similarity": 96.73158264160156
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5307819843292236,
                    "Top": 0.2862499952316284,
                    "Left": 0.1564060002565384,
                    "Height": 0.3987500071525574
                },
                "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
                "ExternalImageId": "image10.jpg",
                "Confidence": 99.99970245361328,
                "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
            },
            "Similarity": 96.48291015625
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5074880123138428,
                    "Top": 0.3774999976158142,
                    "Left": 0.18302799761295319,
                    "Height": 0.3812499940395355
                },
                "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
                "ExternalImageId": "image6.jpg",
                "Confidence": 99.99930572509766,
                "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
            },
            "Similarity": 96.43287658691406
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5574039816856384,
                    "Top": 0.37187498807907104,
                    "Left": 0.14559100568294525,
                    "Height": 0.4181250035762787
                },
```

```
                "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
                "ExternalImageId": "image5.jpg",
                "Confidence": 99.99960327148438,
                "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
            },
            "Similarity": 95.25305938720703
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5773710012435913,
                    "Top": 0.34437501430511475,
                    "Left": 0.12396000325679779,
                    "Height": 0.4337500035762787
                },
                "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
                "ExternalImageId": "image8.jpg",
                "Confidence": 100.0,
                "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
            },
            "Similarity": 95.22837829589844
        }
    ]
}
```

Per ulteriori informazioni, consulta la sezione Ricerca di un volto utilizzando il relativo Face ID nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta Command SearchFacesReference AWS CLI .

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

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

                Usage:    <collectionId> <sourceImage>

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

                """;

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

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
```

```
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }

    public static void searchFaceInCollection(RekognitionClient rekClient, String
 collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(new
 File(sourceImage));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            SearchFacesByImageRequest facesByImageRequest =
 SearchFacesByImageRequest.builder()
                    .image(souImage)
                    .maxFaces(10)
                    .faceMatchThreshold(70F)
                    .collectionId(collectionId)
                    .build();

            SearchFacesByImageResponse imageResponse =
 rekClient.searchFacesByImage(facesByImageRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " +
 face.similarity());
                System.out.println();
            }

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

- Per i dettagli sull'API, SearchFacesconsulta AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
```

```
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def search_faces(self, face_id, threshold, max_faces):
        """
        Searches for faces in the collection that match another face from the
        collection.

        :param face_id: The ID of the face in the collection to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: The list of matching faces found in the collection. This list
does
                 not contain the face specified by `face_id`.
        """
        try:
            response = self.rekognition_client.search_faces(
                CollectionId=self.collection_id,
                FaceId=face_id,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
            logger.info(
                "Found %s faces in %s that match %s.",
                len(faces),
                self.collection_id,
                face_id,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                face_id,
            )
            raise
```

```
        else:
            return faces
```

- Per i dettagli sull'API, consulta SearchFaces AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`SearchFacesByImage`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `SearchFacesByImage`.

Per ulteriori informazioni, consulta Ricerca di un volto (immagine).

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to search for images matching those
/// in a collection.
/// </summary>
public class SearchFacesMatchingImage
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
```

```
            string bucket = "amzn-s3-demo-bucket";
            string photo = "input.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            // Get an image object from S3 bucket.
            var image = new Image()
            {
                S3Object = new S3Object()
                {
                    Bucket = bucket,
                    Name = photo,
                },
            };

            var searchFacesByImageRequest = new SearchFacesByImageRequest()
            {
                CollectionId = collectionId,
                Image = image,
                FaceMatchThreshold = 70F,
                MaxFaces = 2,
            };

            SearchFacesByImageResponse searchFacesByImageResponse = await
rekognitionClient.SearchFacesByImageAsync(searchFacesByImageRequest);

            Console.WriteLine("Faces matching largest face in image from " +
photo);
            searchFacesByImageResponse.FaceMatches.ForEach(face =>
            {
                Console.WriteLine($"FaceId: {face.Face.FaceId}, Similarity:
{face.Similarity}");
            });
        }
    }
```

- Per i dettagli sull'API, SearchFacesByImageconsulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Per cercare volti in una raccolta che corrispondano al volto più grande di un'immagine.

Il `search-faces-by-image` comando seguente cerca i volti in una raccolta che corrispondono al volto più grande dell'immagine specificata. :

```
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"
}
```


Per ulteriori informazioni, consulta Searching for a face using an image nella Amazon Rekognition Developer Guide.

- Per i dettagli sull'API, consulta Command SearchFacesByImageReference AWS CLI .

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

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

                Usage:    <collectionId> <sourceImage>

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

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

```
        }

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

        System.out.println("Searching for a face in a collections");
        searchFacebyId(rekClient, collectionId, faceId);
        rekClient.close();
    }

    public static void searchFacebyId(RekognitionClient rekClient, String
 collectionId, String faceId) {
        try {
            SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceId(faceId)
                    .faceMatchThreshold(70F)
                    .maxFaces(2)
                    .build();

            SearchFacesResponse imageResponse =
 rekClient.searchFaces(searchFacesRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " +
 face.similarity());
                System.out.println();
            }

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

- Per i dettagli sull'API, SearchFacesByImageconsulta AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.

        :param image: The image that contains the reference face to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: A tuple. The first element is the face found in the reference
image.
                 The second element is the list of matching faces found in the
                 collection.
        """
        try:
            response = self.rekognition_client.search_faces_by_image(
                CollectionId=self.collection_id,
                Image=image.image,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            image_face = RekognitionFace(
                {
                    "BoundingBox": response["SearchedFaceBoundingBox"],
                    "Confidence": response["SearchedFaceConfidence"],
                }
            )
            collection_faces = [
                RekognitionFace(face["Face"]) for face in response["FaceMatches"]
            ]
            logger.info(
                "Found %s faces in the collection that match the largest "
                "face in %s.",
                len(collection_faces),
                image.image_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
```

```
                self.collection_id,
                image.image_name,
            )
            raise
        else:
            return image_face, collection_faces
```

- Per i dettagli sull'API, consulta SearchFacesByImage AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Usare Rekognition con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scenari per l'utilizzo di Amazon Rekognition AWS SDKs

I seguenti esempi di codice mostrano come implementare scenari comuni in Amazon Rekognition con. AWS SDKs Questi scenari mostrano come eseguire attività specifiche richiamando più funzioni all'interno di Amazon Rekognition o combinandole con altre. Servizi AWS Ogni scenario include un collegamento al codice sorgente completo, dove è possibile trovare le istruzioni su come configurare ed eseguire il codice.

Gli scenari sono relativi a un livello intermedio di esperienza per aiutarti a comprendere le azioni di servizio nel contesto.

Esempi

- Crea una raccolta Amazon Rekognition e trova i volti al suo interno utilizzando un SDK AWS
- Creazione di un'applicazione di gestione delle risorse fotografiche che consente agli utenti di gestire le foto utilizzando etichette
- Rileva PPE nelle immagini con Amazon Rekognition utilizzando un SDK AWS
- Rileva e visualizza elementi nelle immagini con Amazon Rekognition utilizzando un SDK AWS
- Rileva i volti in un'immagine utilizzando un SDK AWS
- Rileva le informazioni nei video utilizzando Amazon Rekognition e l'SDK AWS
- Rileva oggetti nelle immagini con Amazon Rekognition utilizzando un SDK AWS
- Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un SDK AWS

- Salva EXIF e altre informazioni sull'immagine utilizzando un SDK AWS

Crea una raccolta Amazon Rekognition e trova i volti al suo interno utilizzando un SDK AWS

L'esempio di codice seguente mostra come:

- Crea una raccolta Amazon Rekognition.
- Aggiungi immagini alla raccolta e rileva i volti al suo interno.
- Cerca nella raccolta i volti che corrispondono a un'immagine di riferimento.
- Eliminare una raccolta.

Per ulteriori informazioni, consulta Ricerca dei volti in una raccolta.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Crea classi che eseguono il wrap delle funzioni di Amazon Rekognition.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
from rekognition_objects import RekognitionFace
from rekognition_image_detection import RekognitionImage

logger = logging.getLogger(__name__)


class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
```

```
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    @classmethod
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened
 and its
                                bytes are read.
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
                           file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from
 the
                 file.
        """
        with open(image_file_name, "rb") as img_file:
            image = {"Bytes": img_file.read()}
        name = image_file_name if image_name is None else image_name
        return cls(image, name, rekognition_client)


class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
 API.
    """
```

```
    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def create_collection(self, collection_id):
        """
        Creates an empty collection.

        :param collection_id: Text that identifies the collection.
        :return: The newly created collection.
        """
        try:
            response = self.rekognition_client.create_collection(
                CollectionId=collection_id
            )
            response["CollectionId"] = collection_id
            collection = RekognitionCollection(response, self.rekognition_client)
            logger.info("Created collection %s.", collection_id)
        except ClientError:
            logger.exception("Couldn't create collection %s.", collection_id)
            raise
        else:
            return collection


    def list_collections(self, max_results):
        """
        Lists collections for the current account.

        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
            response =
self.rekognition_client.list_collections(MaxResults=max_results)
            collections = [
                RekognitionCollection({"CollectionId": col_id},
self.rekognition_client)
```

```
                for col_id in response["CollectionIds"]
            ]
        except ClientError:
            logger.exception("Couldn't list collections.")
            raise
        else:
            return collections



class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
```

```
        )


    def to_dict(self):
        """
        Renders parts of the collection data to a dict.

        :return: The collection data as a dict.
        """
        rendering = {
            "collection_id": self.collection_id,
            "collection_arn": self.collection_arn,
            "face_count": self.face_count,
            "created": self.created,
        }
        return rendering


    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        """
        try:
            response = self.rekognition_client.describe_collection(
                CollectionId=self.collection_id
            )
            # Work around capitalization of Arn vs. ARN
            response["CollectionArn"] = response.get("CollectionARN")
            (
                self.collection_arn,
                self.face_count,
                self.created,
            ) = self._unpack_collection(response)
            logger.info("Got data for collection %s.", self.collection_id)
        except ClientError:
            logger.exception("Couldn't get data for collection %s.",
self.collection_id)
            raise
        else:
            return self.to_dict()
```

```
    def delete_collection(self):
        """
        Deletes the collection.
        """
        try:

self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
            logger.exception("Couldn't delete collection %s.",
self.collection_id)
            raise


    def index_faces(self, image, max_faces):
        """
        Finds faces in the specified image, indexes them, and stores them in the
        collection.

        :param image: The image to index.
        :param max_faces: The maximum number of faces to index.
        :return: A tuple. The first element is a list of indexed faces.
                 The second element is a list of faces that couldn't be indexed.
        """
        try:
            response = self.rekognition_client.index_faces(
                CollectionId=self.collection_id,
                Image=image.image,
                ExternalImageId=image.image_name,
                MaxFaces=max_faces,
                DetectionAttributes=["ALL"],
            )
            indexed_faces = [
                RekognitionFace({**face["Face"], **face["FaceDetail"]})
                for face in response["FaceRecords"]
            ]
            unindexed_faces = [
                RekognitionFace(face["FaceDetail"])
                for face in response["UnindexedFaces"]
            ]
            logger.info(
                "Indexed %s faces in %s. Could not index %s faces.",
                len(indexed_faces),
```

```
                image.image_name,
                len(unindexed_faces),
            )
        except ClientError:
            logger.exception("Couldn't index faces in image %s.",
image.image_name)
            raise
        else:
            return indexed_faces, unindexed_faces


    def list_faces(self, max_results):
        """
        Lists the faces currently indexed in the collection.

        :param max_results: The maximum number of faces to return.
        :return: The list of faces in the collection.
        """
        try:
            response = self.rekognition_client.list_faces(
                CollectionId=self.collection_id, MaxResults=max_results
            )
            faces = [RekognitionFace(face) for face in response["Faces"]]
            logger.info(
                "Found %s faces in collection %s.", len(faces),
self.collection_id
            )
        except ClientError:
            logger.exception(
                "Couldn't list faces in collection %s.", self.collection_id
            )
            raise
        else:
            return faces


    def search_faces(self, face_id, threshold, max_faces):
        """
        Searches for faces in the collection that match another face from the
        collection.

        :param face_id: The ID of the face in the collection to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
```

```
        :param max_faces: The maximum number of faces to return.
        :return: The list of matching faces found in the collection. This list
does
                 not contain the face specified by `face_id`.
        """
        try:
            response = self.rekognition_client.search_faces(
                CollectionId=self.collection_id,
                FaceId=face_id,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
            logger.info(
                "Found %s faces in %s that match %s.",
                len(faces),
                self.collection_id,
                face_id,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                face_id,
            )
            raise
        else:
            return faces


    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.

        :param image: The image that contains the reference face to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: A tuple. The first element is the face found in the reference
image.
                 The second element is the list of matching faces found in the
                 collection.
```

```
        """
        try:
            response = self.rekognition_client.search_faces_by_image(
                CollectionId=self.collection_id,
                Image=image.image,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            image_face = RekognitionFace(
                {
                    "BoundingBox": response["SearchedFaceBoundingBox"],
                    "Confidence": response["SearchedFaceConfidence"],
                }
            )
            collection_faces = [
                RekognitionFace(face["Face"]) for face in response["FaceMatches"]
            ]
            logger.info(
                "Found %s faces in the collection that match the largest "
                "face in %s.",
                len(collection_faces),
                image.image_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                image.image_name,
            )
            raise
        else:
            return image_face, collection_faces


class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""

    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
                     functions.
        :param timestamp: The time when the face was detected, if the face was
```

```
                        detected in a video.
        """
        self.bounding_box = face.get("BoundingBox")
        self.confidence = face.get("Confidence")
        self.landmarks = face.get("Landmarks")
        self.pose = face.get("Pose")
        self.quality = face.get("Quality")
        age_range = face.get("AgeRange")
        if age_range is not None:
            self.age_range = (age_range.get("Low"), age_range.get("High"))
        else:
            self.age_range = None
        self.smile = face.get("Smile", {}).get("Value")
        self.eyeglasses = face.get("Eyeglasses", {}).get("Value")
        self.sunglasses = face.get("Sunglasses", {}).get("Value")
        self.gender = face.get("Gender", {}).get("Value", None)
        self.beard = face.get("Beard", {}).get("Value")
        self.mustache = face.get("Mustache", {}).get("Value")
        self.eyes_open = face.get("EyesOpen", {}).get("Value")
        self.mouth_open = face.get("MouthOpen", {}).get("Value")
        self.emotions = [
            emo.get("Type")
            for emo in face.get("Emotions", [])
            if emo.get("Confidence", 0) > 50
        ]
        self.face_id = face.get("FaceId")
        self.image_id = face.get("ImageId")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
        rendering = {}
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.age_range is not None:
            rendering["age"] = f"{self.age_range[0]} - {self.age_range[1]}"
        if self.gender is not None:
            rendering["gender"] = self.gender
        if self.emotions:
            rendering["emotions"] = self.emotions
```

```
        if self.face_id is not None:
            rendering["face_id"] = self.face_id
        if self.image_id is not None:
            rendering["image_id"] = self.image_id
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        has = []
        if self.smile:
            has.append("smile")
        if self.eyeglasses:
            has.append("eyeglasses")
        if self.sunglasses:
            has.append("sunglasses")
        if self.beard:
            has.append("beard")
        if self.mustache:
            has.append("mustache")
        if self.eyes_open:
            has.append("open eyes")
        if self.mouth_open:
            has.append("open mouth")
        if has:
            rendering["has"] = has
        return rendering
```

Usa le classi wrapper per creare una raccolta di volti a partire da un set di immagini e poi cercare i volti nella raccolta.

```
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)
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUsare Rekognition con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Creazione di un'applicazione di gestione delle risorse fotografiche che consente agli utenti di gestire le foto utilizzando etichette

Nell'esempio di codice seguente viene illustrato come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante etichette.

.NET

SDK per .NET

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK per C++

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK per Java 2.x

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK per JavaScript (v3)

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. GitHub

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK per Kotlin

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK per PHP

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK per Rust

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUsare Rekognition con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Rileva PPE nelle immagini con Amazon Rekognition utilizzando un SDK AWS

Il seguente esempio di codice mostra come creare un'app che utilizza Amazon Rekognition per rilevare i dispositivi di protezione individuale (DPI) nelle immagini.

Java

SDK per Java 2.x

Mostra come creare una AWS Lambda funzione che rileva le immagini con dispositivi di protezione individuale.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUsare Rekognition con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Rileva e visualizza elementi nelle immagini con Amazon Rekognition utilizzando un SDK AWS

L'esempio di codice seguente mostra come:

- Rileva gli elementi nelle immagini utilizzando Amazon Rekognition.
- Visualizza immagini e disegna riquadri di delimitazione attorno agli elementi rilevati.

Per ulteriori informazioni, consulta Visualizzazione dei riquadri di delimitazione.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Crea classi per eseguire il wrap delle funzioni di Amazon Rekognition.

```
import logging
from pprint import pprint
import boto3
```

```
from botocore.exceptions import ClientError
import requests

from rekognition_objects import (
    RekognitionFace,
    RekognitionCelebrity,
    RekognitionLabel,
    RekognitionModerationLabel,
    RekognitionText,
    show_bounding_boxes,
    show_polygons,
)

logger = logging.getLogger(__name__)


class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    @classmethod
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened
 and its
                                bytes are read.
```

```
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
                           file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from
the
                 file.
        """
        with open(image_file_name, "rb") as img_file:
            image = {"Bytes": img_file.read()}
        name = image_file_name if image_name is None else image_name
        return cls(image, name, rekognition_client)


    @classmethod
    def from_bucket(cls, s3_object, rekognition_client):
        """
        Creates a RekognitionImage object from an Amazon S3 object.

        :param s3_object: An Amazon S3 object that identifies the image. The
image
                          is not retrieved until needed for a later call.
        :param rekognition_client: A Boto3 Rekognition client.
        :return: The RekognitionImage object, initialized with Amazon S3 object
data.
        """
        image = {"S3Object": {"Bucket": s3_object.bucket_name, "Name":
s3_object.key}}
        return cls(image, s3_object.key, rekognition_client)


    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
```

```
            raise
        else:
            return faces


    def detect_labels(self, max_labels):
        """
        Detects labels in the image. Labels are objects and people.

        :param max_labels: The maximum number of labels to return.
        :return: The list of labels detected in the image.
        """
        try:
            response = self.rekognition_client.detect_labels(
                Image=self.image, MaxLabels=max_labels
            )
            labels = [RekognitionLabel(label) for label in response["Labels"]]
            logger.info("Found %s labels in %s.", len(labels), self.image_name)
        except ClientError:
            logger.info("Couldn't detect labels in %s.", self.image_name)
            raise
        else:
            return labels


    def recognize_celebrities(self):
        """
        Detects celebrities in the image.

        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
        try:
            response =
self.rekognition_client.recognize_celebrities(Image=self.image)
            celebrities = [
                RekognitionCelebrity(celeb) for celeb in
response["CelebrityFaces"]
            ]
            other_faces = [
                RekognitionFace(face) for face in response["UnrecognizedFaces"]
            ]
            logger.info(
```

```
                "Found %s celebrities and %s other faces in %s.",
                len(celebrities),
                len(other_faces),
                self.image_name,
            )
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.",
self.image_name)
            raise
        else:
            return celebrities, other_faces



    def compare_faces(self, target_image, similarity):
        """
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value
greater
                           than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                 reference image. The second element is the list of faces that
have
                 a similarity value below the specified threshold.
        """
        try:
            response = self.rekognition_client.compare_faces(
                SourceImage=self.image,
                TargetImage=target_image.image,
                SimilarityThreshold=similarity,
            )
            matches = [
                RekognitionFace(match["Face"]) for match in
response["FaceMatches"]
            ]
            unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
            logger.info(
                "Found %s matched faces and %s unmatched faces.",
                len(matches),
                len(unmatches),
            )
```

```
        except ClientError:
            logger.exception(
                "Couldn't match faces from %s to %s.",
                self.image_name,
                target_image.image_name,
            )
            raise
        else:
            return matches, unmatches


    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify
content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image
            )
            labels = [
                RekognitionModerationLabel(label)
                for label in response["ModerationLabels"]
            ]
            logger.info(
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels


    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
```

```
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in
 response["TextDetections"]]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

Crea funzioni helper per disegnare riquadri di delimitazione e poligoni.

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

logger = logging.getLogger(__name__)


def show_bounding_boxes(image_bytes, box_sets, colors):
    """
    Draws bounding boxes on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param box_sets: A list of lists of bounding boxes to draw on the image.
    :param colors: A list of colors to use to draw the bounding boxes.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for boxes, color in zip(box_sets, colors):
        for box in boxes:
            left = image.width * box["Left"]
            top = image.height * box["Top"]
            right = (image.width * box["Width"]) + left
            bottom = (image.height * box["Height"]) + top
            draw.rectangle([left, top, right, bottom], outline=color, width=3)
    image.show()
```

```
def show_polygons(image_bytes, polygons, color):
    """
    Draws polygons on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param polygons: The list of polygons to draw on the image.
    :param color: The color to use to draw the polygons.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for polygon in polygons:
        draw.polygon(
            [
                (image.width * point["X"], image.height * point["Y"])
                for point in polygon
            ],
            outline=color,
        )
    image.show()
```

Crea classi per analizzare gli oggetti restituiti da Amazon Rekognition.

```
class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""

    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
                     functions.
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
        """
        self.bounding_box = face.get("BoundingBox")
        self.confidence = face.get("Confidence")
        self.landmarks = face.get("Landmarks")
        self.pose = face.get("Pose")
        self.quality = face.get("Quality")
```

```
        age_range = face.get("AgeRange")
        if age_range is not None:
            self.age_range = (age_range.get("Low"), age_range.get("High"))
        else:
            self.age_range = None
        self.smile = face.get("Smile", {}).get("Value")
        self.eyeglasses = face.get("Eyeglasses", {}).get("Value")
        self.sunglasses = face.get("Sunglasses", {}).get("Value")
        self.gender = face.get("Gender", {}).get("Value", None)
        self.beard = face.get("Beard", {}).get("Value")
        self.mustache = face.get("Mustache", {}).get("Value")
        self.eyes_open = face.get("EyesOpen", {}).get("Value")
        self.mouth_open = face.get("MouthOpen", {}).get("Value")
        self.emotions = [
            emo.get("Type")
            for emo in face.get("Emotions", [])
            if emo.get("Confidence", 0) > 50
        ]
        self.face_id = face.get("FaceId")
        self.image_id = face.get("ImageId")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
        rendering = {}
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.age_range is not None:
            rendering["age"] = f"{self.age_range[0]} - {self.age_range[1]}"
        if self.gender is not None:
            rendering["gender"] = self.gender
        if self.emotions:
            rendering["emotions"] = self.emotions
        if self.face_id is not None:
            rendering["face_id"] = self.face_id
        if self.image_id is not None:
            rendering["image_id"] = self.image_id
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        has = []
```

```
        if self.smile:
            has.append("smile")
        if self.eyeglasses:
            has.append("eyeglasses")
        if self.sunglasses:
            has.append("sunglasses")
        if self.beard:
            has.append("beard")
        if self.mustache:
            has.append("mustache")
        if self.eyes_open:
            has.append("open eyes")
        if self.mouth_open:
            has.append("open mouth")
        if has:
            rendering["has"] = has
        return rendering



class RekognitionCelebrity:
    """Encapsulates an Amazon Rekognition celebrity."""

    def __init__(self, celebrity, timestamp=None):
        """
        Initializes the celebrity object.

        :param celebrity: Celebrity data, in the format returned by Amazon
 Rekognition
                          functions.
        :param timestamp: The time when the celebrity was detected, if the
 celebrity
                          was detected in a video.
        """
        self.info_urls = celebrity.get("Urls")
        self.name = celebrity.get("Name")
        self.id = celebrity.get("Id")
        self.face = RekognitionFace(celebrity.get("Face"))
        self.confidence = celebrity.get("MatchConfidence")
        self.bounding_box = celebrity.get("BoundingBox")
        self.timestamp = timestamp

    def to_dict(self):
        """
```

```
        Renders some of the celebrity data to a dict.

        :return: A dict that contains the celebrity data.
        """
        rendering = self.face.to_dict()
        if self.name is not None:
            rendering["name"] = self.name
        if self.info_urls:
            rendering["info URLs"] = self.info_urls
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionPerson:
    """Encapsulates an Amazon Rekognition person."""

    def __init__(self, person, timestamp=None):
        """
        Initializes the person object.

        :param person: Person data, in the format returned by Amazon Rekognition
                       functions.
        :param timestamp: The time when the person was detected, if the person
                          was detected in a video.
        """
        self.index = person.get("Index")
        self.bounding_box = person.get("BoundingBox")
        face = person.get("Face")
        self.face = RekognitionFace(face) if face is not None else None
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the person data to a dict.

        :return: A dict that contains the person data.
        """
        rendering = self.face.to_dict() if self.face is not None else {}
        if self.index is not None:
            rendering["index"] = self.index
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
```

```
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionLabel:
    """Encapsulates an Amazon Rekognition label."""

    def __init__(self, label, timestamp=None):
        """
        Initializes the label object.

        :param label: Label data, in the format returned by Amazon Rekognition
                      functions.
        :param timestamp: The time when the label was detected, if the label
                          was detected in a video.
        """
        self.name = label.get("Name")
        self.confidence = label.get("Confidence")
        self.instances = label.get("Instances")
        self.parents = label.get("Parents")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the label data to a dict.

        :return: A dict that contains the label data.
        """
        rendering = {}
        if self.name is not None:
            rendering["name"] = self.name
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionModerationLabel:
    """Encapsulates an Amazon Rekognition moderation label."""

    def __init__(self, label, timestamp=None):
        """
```

```
        Initializes the moderation label object.

        :param label: Label data, in the format returned by Amazon Rekognition
                      functions.
        :param timestamp: The time when the moderation label was detected, if the
                          label was detected in a video.
        """
        self.name = label.get("Name")
        self.confidence = label.get("Confidence")
        self.parent_name = label.get("ParentName")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the moderation label data to a dict.

        :return: A dict that contains the moderation label data.
        """
        rendering = {}
        if self.name is not None:
            rendering["name"] = self.name
        if self.parent_name is not None:
            rendering["parent_name"] = self.parent_name
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionText:
    """Encapsulates an Amazon Rekognition text element."""

    def __init__(self, text_data):
        """
        Initializes the text object.

        :param text_data: Text data, in the format returned by Amazon Rekognition
                          functions.
        """
        self.text = text_data.get("DetectedText")
        self.kind = text_data.get("Type")
        self.id = text_data.get("Id")
        self.parent_id = text_data.get("ParentId")
        self.confidence = text_data.get("Confidence")
```

```
        self.geometry = text_data.get("Geometry")

    def to_dict(self):
        """
        Renders some of the text data to a dict.

        :return: A dict that contains the text data.
        """
        rendering = {}
        if self.text is not None:
            rendering["text"] = self.text
        if self.kind is not None:
            rendering["kind"] = self.kind
        if self.geometry is not None:
            rendering["polygon"] = self.geometry.get("Polygon")
        return rendering
```

Usa le classi wrapper per rilevare gli elementi nelle immagini e visualizzarne i riquadri di delimitazione. Le immagini utilizzate in questo esempio sono disponibili su GitHub insieme alle istruzioni e ad altro codice.

```
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)
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUsare Rekognition con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Rileva i volti in un'immagine utilizzando un SDK AWS

L'esempio di codice seguente mostra come:

- Salva un'immagine in un bucket Amazon S3.
- Utilizza Amazon Rekognition per rilevare i dettagli del viso, come fascia di età, sesso ed emozione (ad esempio sorridente).
- Visualizzare questi dettagli.

Rust

SDK per Rust

Salva l'immagine in un bucket Amazon S3 con il prefisso uploads, utilizza Amazon Rekognition per rilevare i dettagli del viso, come fascia di età, sesso ed emozione (sorridente, ecc.) e visualizza tali dettagli.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Amazon S3

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUsare Rekognition con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Rileva le informazioni nei video utilizzando Amazon Rekognition e l'SDK AWS

Gli esempi di codice seguenti mostrano come:

- Avvia i processi di Amazon Rekognition per rilevare elementi come persone, oggetti e testo nei video.
- Controlla lo stato del processo fino al suo termine.
- Crea un output con l'elenco degli elementi rilevati da ciascun processo.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Ottieni risultati relativi alle celebrità da un video che si trova in un bucket Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.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);
        }
    }
}
```

Rileva le etichette in un video tramite un'operazione di rilevamento delle etichette.

```
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);
        }
    }
}
```

Rileva i volti in un video archiviato in un bucket Amazon S3.

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

```
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

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

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

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

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

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
```

```
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

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

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

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video 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);
        }
    }
}
```

Rileva contenuti non appropriati o offensivi in un video archiviato in un bucket Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
```

```
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import
 software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

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

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;
```

```
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

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

        startModerationDetection(rekClient, channel, bucket, video);
        getModResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startModerationDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {

        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video 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);
        }
    }
}
```

Rileva segmenti di segnali d'azione tecnici e segmenti di rilevamento delle riprese in un video archiviato in un bucket Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
```

```
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
import java.util.List;

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

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
```

```
                    video - The name of video (for example, people.mp4).\s
                    topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                    roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """";

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

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

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

    public static void startSegmentDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
```

```
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video 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);
        }
    }
}
```

Rileva il testo in un video archiviato in un bucket Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

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

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
```

```
                    topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                    roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """";

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

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

        startTextLabels(rekClient, channel, bucket, video);
        getTextResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startTextLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video 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);
```

```
        }
    }
}
```

Rileva le persone in un video archiviato in un bucket Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

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

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>
```

```
                Where:
                   bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

       if (args.length != 4) {
           System.out.println(usage);
           System.exit(1);
       }

       String bucket = args[0];
       String video = args[1];
       String topicArn = args[2];
       String roleArn = args[3];
       Region region = Region.US_EAST_1;
       RekognitionClient rekClient = RekognitionClient.builder()
               .region(region)
               .build();

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

       startPersonLabels(rekClient, channel, bucket, video);
       getPersonDetectionResults(rekClient);
       System.out.println("This example is done!");
       rekClient.close();
   }

   public static void startPersonLabels(RekognitionClient rekClient,
           NotificationChannel channel,
           String bucket,
           String video) {
       try {
           S3Object s3Obj = S3Object.builder()
                   .bucket(bucket)
                   .name(video)
                   .build();
```

```
            Video 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);
        }
```

```
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection
  - StartTextDetection

Kotlin

SDK per Kotlin

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Rileva i volti in un video archiviato in un bucket Amazon S3.

```
suspend fun startFaceDetection(
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
```

```
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vidOb
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val startLabelDetectionResult = rekClient.startFaceDetection(request)
        startJobId = startLabelDetectionResult.jobId.toString()
    }
}

suspend fun getFaceResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        var response: GetFaceDetectionResponse? = null

        val recognitionRequest =
            GetFaceDetectionRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            response = rekClient.getFaceDetection(recognitionRequest)
            status = response.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
```

```
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = response?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        // Show face information.
        response?.faces?.forEach { face ->
            println("Age: ${face.face?.ageRange}")
            println("Face: ${face.face?.beard}")
            println("Eye glasses: ${face?.face?.eyeglasses}")
            println("Mustache: ${face.face?.mustache}")
            println("Smile: ${face.face?.smile}")
        }
    }
}
```

Rileva contenuti non appropriati o offensivi in un video archiviato in un bucket Amazon S3.

```
suspend fun startModerationDetection(
    channel: NotificationChannel?,
    bucketVal: String?,
    videoVal: String?,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }
    val request =
        StartContentModerationRequest {
```

```
            jobTag = "Moderation"
            notificationChannel = channel
            video = vidOb
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val startModDetectionResult = rekClient.startContentModeration(request)
        startJobId = startModDetectionResult.jobId.toString()
    }
}

suspend fun getModResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var modDetectionResponse: GetContentModerationResponse? = null

        val modRequest =
            GetContentModerationRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            modDetectionResponse = rekClient.getContentModeration(modRequest)
            status = modDetectionResponse.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = modDetectionResponse?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")
```

```
        modDetectionResponse?.moderationLabels?.forEach { mod ->
            val seconds: Long = mod.timestamp / 1000
            print("Mod label: $seconds ")
            println(mod.moderationLabel)
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API SDK AWS per Kotlin.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection
  - StartTextDetection

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUsare Rekognition con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Rileva oggetti nelle immagini con Amazon Rekognition utilizzando un SDK AWS

Gli esempi di codice seguenti mostrano come creare un'applicazione che utilizza Amazon Rekognition per rilevare oggetti in base a una categoria nelle immagini.

.NET

SDK per .NET

Mostra come utilizzare l'API .NET di Amazon Rekognition per creare un'applicazione che utilizza Amazon Rekognition per identificare gli oggetti in base a una categoria nelle immagini situate in un bucket Amazon Simple Storage Service (Amazon S3). L'applicazione invia all'amministratore una notifica e-mail sui risultati tramite Amazon Simple Email Service (Amazon SES).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK per Java 2.x

Mostra come utilizzare l'API Java di Amazon Rekognition per creare un'applicazione che utilizza Amazon Rekognition per identificare gli oggetti in base a una categoria nelle immagini situate in un bucket Amazon Simple Storage Service (Amazon S3). L'applicazione invia all'amministratore una notifica e-mail sui risultati tramite Amazon Simple Email Service (Amazon SES).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK per JavaScript (v3)

Mostra come usare Amazon Rekognition AWS SDK per JavaScript con la per creare un'app che utilizzi Amazon Rekognition per identificare gli oggetti per categoria nelle immagini che si trovano in un bucket Amazon Simple Storage Service (Amazon S3). L'applicazione invia all'amministratore una notifica e-mail sui risultati tramite Amazon Simple Email Service (Amazon SES).

Scopri come:

- Creare un utente non autenticato tramite Amazon Cognito.
- Analizza le immagini per rilevare gli oggetti tramite Amazon Rekognition.
- Verificare un indirizzo e-mail per Amazon SES.
- Inviare una notifica e-mail tramite Amazon SES.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK per Kotlin

Mostra come utilizzare l'API Kotlin di Amazon Rekognition per creare un'applicazione che utilizza Amazon Rekognition per identificare gli oggetti in base a una categoria nelle immagini situate in un bucket Amazon Simple Storage Service (Amazon S3). L'applicazione invia all'amministratore una notifica e-mail sui risultati tramite Amazon Simple Email Service (Amazon SES).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK per Python (Boto3)

Illustra come utilizzare il AWS SDK per Python (Boto3) per creare un'applicazione Web che consenta di eseguire le seguenti operazioni:

- Caricamento di foto in un bucket Amazon Simple Storage Service (Amazon S3).
- Utilizzo di Amazon Rekognition per analizzare ed etichettare le foto.
- Utilizzo di Amazon Simple Email Service (Amazon SES) per inviare report dell'analisi delle immagini tramite e-mail.

Questo esempio contiene due componenti principali: una pagina web scritta in JavaScript che è costruita con React e un servizio REST scritto in Python creato con Flask-. RESTful

È possibile utilizzare la pagina Web React per:

- Visualizzare un elenco di immagini archiviate nel bucket S3.
- Caricare le immagini dal computer nel bucket S3.
- Visualizzare immagini ed etichette che identificano gli elementi rilevati nell'immagine.
- Ottenere un report relativo a tutte le immagini nel bucket S3 e inviarlo tramite email.

La pagina Web richiama il servizio REST. Il servizio invia richieste a AWS per eseguire le seguenti operazioni:

- Ottenere e filtrare l'elenco delle immagini nel bucket S3.
- Caricare le foto nel bucket S3.
- Utilizzare Amazon Rekognition per analizzare le singole foto e ottenere un elenco di etichette che identificano gli articoli rilevati al loro interno.
- Analizzare tutte le foto presenti nel bucket S3 e usare Amazon SES per inviare un report tramite e-mail.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Amazon S3
- Amazon SES

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUsare Rekognition con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un SDK AWS

Gli esempi di codice seguenti mostrano come rilevare persone e oggetti in un video con Amazon Rekognition.

Java

SDK per Java 2.x

Mostra come utilizzare l'API Java di Amazon Rekognition per creare un'applicazione che rileva volti e oggetti nei video situati in un bucket Amazon Simple Storage Service (Amazon S3). L'applicazione invia all'amministratore una notifica e-mail sui risultati tramite Amazon Simple Email Service (Amazon SES).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK per Python (Boto3)

Usa Amazon Rekognition per rilevare volti, oggetti e persone nei video avviando processi di rilevamento asincrono. Questo esempio, inoltre, configura Amazon Rekognition per notificare un argomento Amazon Simple Notification Service (Amazon SNS) al completamento dei processi e sottoscrive una coda Amazon Simple Queue Service (Amazon SQS) all'argomento. Quando la coda riceve un messaggio su un processo, questo viene recuperato e vengono restituiti i risultati.

Questo esempio è visualizzato al meglio su GitHub. Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUsare Rekognition con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Salva EXIF e altre informazioni sull'immagine utilizzando un SDK AWS

L'esempio di codice seguente mostra come:

- Recuperare informazioni EXIF da un file JPG, JPEG o PNG.
- Carica il file immagine in un bucket Amazon S3.
- Utilizza Amazon Rekognition per identificare i tre attributi principali (etichette) nel file.
- Aggiungi le informazioni su EXIF ed etichette a una tabella Amazon DynamoDB nella regione.

Rust

SDK per Rust

Recupera le informazioni EXIF da un file JPG, JPEG o PNG, carica il file di immagine in un bucket Amazon S3, utilizza Amazon Rekognition per identificare i tre attributi principali (etichette in Amazon Rekognition) nel file e aggiungi le informazioni su EXIF ed etichette a una tabella Amazon DynamoDB nella regione.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- DynamoDB
- Amazon Rekognition
- Amazon S3

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUsare Rekognition con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Documentazione di riferimento API

La documentazione di riferimento delle API Amazon Rekognition si trova ora in Amazon Rekognition API Reference.

Amazon Rekognition Security

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

Utilizza i seguenti argomenti per scoprire come proteggere le risorse di Amazon Rekognition.

Argomenti

- Identity and Access Management per Amazon Rekognition
- Protezione dei dati in Amazon Rekognition
- Utilizzo di Amazon Rekognition con endpoint VPC Amazon
- Convalida della conformità per Amazon Rekognition
- Resilienza in Amazon Rekognition
- Analisi della configurazione e delle vulnerabilità in Amazon Rekognition
- Prevenzione del problema "confused deputy" tra servizi
- Sicurezza dell'infrastruttura in Amazon Rekognition

Identity and Access Management per Amazon Rekognition

AWS Identity and Access Management (IAM) è un programma Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi può essere autenticato (accesso effettuato) e autorizzato (dotato di autorizzazioni) per utilizzare le risorse Amazon Rekognition. IAM è un software Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Argomenti

- Destinatari
- Autenticazione con identità
- Gestione dell'accesso con policy
- Come funziona Amazon Rekognition con IAM
- AWS politiche gestite per Amazon Rekognition
- Best practice per le policy

- Utilizzo della console Amazon Rekognition
- Esempi di utilizzo di policy basate sull'identità di Amazon Rekognition
- Esempi di policy basate sulle risorse di Amazon Rekognition
- Risoluzione dei problemi relativi all'identità e all'accesso di Amazon Rekognition

Destinatari

Il modo in cui usi AWS Identity and Access Management (IAM) varia a seconda del lavoro svolto in Amazon Rekognition.

Utente del servizio: se utilizzi il servizio Amazon Rekognition per svolgere il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. All'aumentare del numero di funzionalità Amazon Rekognition utilizzate per il lavoro, potrebbero essere necessarie ulteriori autorizzazioni. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità in Amazon Rekognition, consulta Risoluzione dei problemi relativi all'identità e all'accesso di Amazon Rekognition.

Amministratore del servizio: se sei il responsabile delle risorse Amazon Rekognition presso la tua azienda, probabilmente disponi dell'accesso completo a Amazon Rekognition. Il compito dell'utente è determinare le funzionalità e le risorse di Amazon Rekognition a cui gli utenti del servizio devono accedere. Devi inviare le richieste all'amministratore IAM per cambiare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per ulteriori informazioni su come la tua azienda può utilizzare IAM con Amazon Rekognition, consulta Come funziona Amazon Rekognition con IAM.

Amministratore IAM: se sei un amministratore IAM, potresti essere interessato a ottenere informazioni su come scrivere policy per gestire l'accesso ad Amazon Rekognition. Per visualizzare policy basate su identità Amazon Rekognition di esempio che puoi utilizzare in IAM, consulta Esempi di utilizzo di policy basate sull'identità di Amazon Rekognition.

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi essere autenticato (aver effettuato l' Utente root dell'account AWS accesso AWS) come utente IAM o assumendo un ruolo IAM.

Puoi accedere AWS come identità federata utilizzando le credenziali fornite tramite una fonte di identità. AWS IAM Identity Center Gli utenti (IAM Identity Center), l'autenticazione Single Sign-On

della tua azienda e le tue credenziali di Google o Facebook sono esempi di identità federate. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Quando accedi AWS utilizzando la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere al AWS Management Console o al portale di AWS accesso. Per ulteriori informazioni sull'accesso a AWS, vedi Come accedere al tuo Account AWS nella Guida per l'Accedi ad AWS utente.

Se accedi a AWS livello di codice, AWS fornisce un kit di sviluppo software (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le tue richieste utilizzando le tue credenziali. Se non utilizzi AWS strumenti, devi firmare tu stesso le richieste. Per ulteriori informazioni sul metodo consigliato per la firma delle richieste, consulta Signature Version 4 AWS per le richieste API nella Guida per l'utente IAM.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, ti AWS consiglia di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza del tuo account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'utente di AWS IAM Identity Center e Utilizzo dell'autenticazione a più fattori (MFA)AWS in IAM nella Guida per l'utente IAM.

Utenti IAM e gruppi

Un utente IAM è un'identità interna Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, se si hanno casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente IAM.

Un gruppo IAM è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, potresti avere un gruppo denominato IAMAdminse concedere a quel gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli

utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Casi d'uso per utenti IAM nella Guida per l'utente IAM.

Ruoli IAM

Un ruolo IAM è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. Per assumere temporaneamente un ruolo IAM in AWS Management Console, puoi passare da un ruolo utente a un ruolo IAM (console). Puoi assumere un ruolo chiamando un'operazione AWS CLI o AWS API o utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Accesso utente federato: per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta Create a role for a third-party identity provider (federation) nella Guida per l'utente IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .
- Autorizzazioni utente IAM temporanee: un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, con alcuni Servizi AWS, è possibile allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.
- Accesso a più servizi: alcuni Servizi AWS utilizzano le funzionalità di altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è normale che quel servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.

- Sessioni di accesso inoltrato (FAS): quando utilizzi un utente o un ruolo IAM per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FAS utilizza le autorizzazioni del principale che chiama an Servizio AWS, combinate con la richiesta Servizio AWS per effettuare richieste ai servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le operazioni. Per i dettagli delle policy relative alle richieste FAS, consulta Forward access sessions.
- Ruolo di servizio: un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Create a role to delegate permissions to an Servizio AWS nella Guida per l'utente IAM.
- Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un ruolo IAM per gestire le credenziali temporanee per le applicazioni in esecuzione su un' EC2 istanza e che AWS CLI effettuano richieste AWS API. Questa soluzione è preferibile alla memorizzazione delle chiavi di accesso all'interno dell' EC2 istanza. Per assegnare un AWS ruolo a un' EC2 istanza e renderlo disponibile per tutte le sue applicazioni, create un profilo di istanza collegato all'istanza. Un profilo di istanza contiene il ruolo e consente ai programmi in esecuzione sull' EC2 istanza di ottenere credenziali temporanee. Per ulteriori informazioni, consulta Utilizzare un ruolo IAM per concedere le autorizzazioni alle applicazioni in esecuzione su EC2 istanze Amazon nella IAM User Guide.

Gestione dell'accesso con policy

Puoi controllare l'accesso AWS creando policy e collegandole a AWS identità o risorse. Una policy è un oggetto AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un principale (utente, utente root o sessione di ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per ulteriori informazioni

sulla struttura e sui contenuti dei documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente IAM.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire operazioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire operazioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. L'amministratore può quindi aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'operazione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dall' AWS Management Console AWS CLI, dall'o dall' AWS API.

Utilizzo di policy basate su identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono integrate direttamente in un singolo utente, gruppo o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più utenti, gruppi e ruoli nel tuo Account AWS. Le politiche gestite includono politiche AWS gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scelta fra policy gestite e policy inline nella Guida per l'utente IAM.

Utilizzo delle policy basate su risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Esempi di policy basate sulle risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non puoi utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Elenchi di controllo degli accessi () ACLs

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLs sono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3 e Amazon VPC sono esempi di servizi che supportano. AWS WAF ACLs Per ulteriori informazioni ACLs, consulta la panoramica della lista di controllo degli accessi (ACL) nella Amazon Simple Storage Service Developer Guide.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite delle autorizzazioni è una funzionalità avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal`sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente IAM.
- Politiche di controllo del servizio (SCPs): SCPs sono politiche JSON che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in. AWS Organizations AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più di proprietà dell' Account AWS azienda. Se abiliti tutte le funzionalità di un'organizzazione, puoi applicare le politiche di controllo del servizio (SCPs) a uno o tutti i tuoi account. L'SCP limita le autorizzazioni per le entità presenti negli account dei membri, inclusa ciascuna di esse. Utente root dell'account AWS Per ulteriori informazioni su Organizations and SCPs, consulta le politiche di controllo dei servizi nella Guida AWS Organizations per l'utente.
- Politiche di controllo delle risorse (RCPs): RCPs sono politiche JSON che puoi utilizzare per impostare le autorizzazioni massime disponibili per le risorse nei tuoi account senza aggiornare le politiche IAM allegate a ciascuna risorsa di tua proprietà. L'RCP limita le autorizzazioni per le risorse negli account dei membri e può influire sulle autorizzazioni effettive per le identità,

incluse le Utente root dell'account AWS, indipendentemente dal fatto che appartengano o meno all'organizzazione. Per ulteriori informazioni su Organizations e RCPs, incluso un elenco di Servizi AWS tale supporto RCPs, vedere Resource control policies (RCPs) nella Guida per l'AWS Organizations utente.

- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta Policy di sessione nella Guida per l'utente IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta la logica di valutazione delle policy nella IAM User Guide.

Come funziona Amazon Rekognition con IAM

Prima di utilizzare IAM per gestire l'accesso ad Amazon Rekognition, è necessario comprendere quali funzioni IAM sono disponibili per l'uso con Amazon Rekognition. Per avere una visione di alto livello di come Amazon Rekognition AWS e altri servizi funzionano con IAM AWS , consulta Services That Work with IAM nella IAM User Guide.

Argomenti

- Policy basate su identità di Amazon Rekognition
- Policy basate sulle risorse di Amazon Rekognition
- Ruoli IAM Amazon Rekognition

Policy basate su identità di Amazon Rekognition

Con le policy basate su identità di IAM, è possibile specificare quali azioni e risorse sono consentite o rifiutate, nonché le condizioni in base alle quali le azioni sono consentite o rifiutate. Amazon Rekognition supporta operazioni, risorse e chiavi di condizione specifiche. Per informazioni su tutti gli elementi utilizzati in una policy JSON, consulta Documentazione di riferimento degli elementi delle policy JSON IAM nella Guida per l'utente IAM.

Operazioni

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'elemento `Action`di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso a un criterio. Le azioni politiche in genere hanno lo stesso nome dell'operazione AWS API associata. Ci sono alcune eccezioni, ad esempio le operazioni di sola autorizzazione che non hanno un'operazione API corrispondente. Esistono anche alcune operazioni che richiedono più operazioni in una policy. Queste operazioni aggiuntive sono denominate operazioni dipendenti.

Includi le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Le operazioni delle policy in Amazon Rekognition utilizzano il seguente prefisso prima dell'operazione: `rekognition:`. Ad esempio, per concedere a qualcuno l'autorizzazione a rilevare oggetti, scene o concetti in un'immagine con l'operazione `DetectLabels` API Amazon Rekognition, è possibile includere l'azione `rekognition:DetectLabels` nei criteri. Le istruzioni della policy devono includere un elemento `Action` o `NotAction`. Amazon Rekognition definisce un proprio set di operazioni che descrivono le attività eseguibili con quel servizio.

Per specificare più azioni in una sola istruzione, separa ciascuna di esse con una virgola come mostrato di seguito:

```
"Action": [
      "rekognition:action1",
      "rekognition:action2"
```

È possibile specificare più azioni tramite caratteri jolly (*). Ad esempio, per specificare tutte le azioni che iniziano con la parola `Describe`, includi la seguente azione:

```
"Action": "rekognition:Describe*"
```

Per visualizzare un elenco delle operazioni Amazon Rekognition, consulta Operazioni definite da Amazon Rekognition nella Guida per l'utente IAM.

Risorse

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'elemento JSON `Resource`della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Le istruzioni devono includere un elemento `Resource`o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). È possibile eseguire questa operazione per operazioni che supportano un tipo di risorsa specifico, note come autorizzazioni a livello di risorsa.

Per le operazioni che non supportano le autorizzazioni a livello di risorsa, ad esempio le operazioni di elenco, utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

Per ulteriori informazioni sul formato di ARNs, consulta Amazon Resource Names (ARNs) e AWS Service Namespaces.

Ad esempio, per specificare l'organizzazione `MyCollection` nell'istruzione, utilizza il seguente ARN:

```
"Resource": "arn:aws:rekognition:us-east-1:123456789012:collection/MyCollection"
```

Per specificare tutti le istanze che appartengono ad un account specifico, utilizza il carattere jolly (*):

```
"Resource": "arn:aws:rekognition:us-east-1:123456789012:collection/*"
```

Alcune operazioni di Amazon Rekognition, ad esempio quelle per la creazione delle risorse, non possono essere eseguite su una risorsa specifica. In questi casi, è necessario utilizzare il carattere jolly (*).

```
"Resource": "*"
```

Per visualizzare un elenco dei tipi di risorse di Amazon Rekognition ARNs e relativi, consulta Resources Defined by Amazon Rekognition nella IAM User Guide. Per informazioni sulle operazioni con cui è possibile specificare l'ARN di ogni risorsa, consultare Operazioni definite da Amazon Rekognition.

Chiavi di condizione

Amazon Rekognition non fornisce chiavi di condizione specifiche del servizio, ma supporta l'utilizzo di alcune chiavi di condizione globali. Per visualizzare tutte le chiavi di condizione AWS globali, consulta AWS Global Condition Context Keys nella IAM User Guide.

Policy basate sulle risorse di Amazon Rekognition

Amazon Rekognition supporta le policy basate sulle risorse per la copia dei modelli delle etichette personalizzate. Per ulteriori informazioni, consulta Esempi di policy basate su risorse di Amazon Rekognition.

Anche altri servizi, ad esempio Amazon S3, supportano policy di autorizzazioni basate su risorse. Ad esempio, è possibile associare una policy a un bucket S3 per gestire le autorizzazioni di accesso a quel bucket.

Per accedere alle immagini archiviate in un bucket Amazon S3, devi disporre delle autorizzazioni di accesso all'oggetto nel bucket S3. Grazie a questa autorizzazione, Amazon Rekognition può scaricare le immagini dal bucket S3. La seguente policy di esempio consente all'utente di eseguire l'`s3:GetObject`azione sul bucket S3 denominato amzn-s3-demo-bucket3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket3/*"
            ]
        }
    ]
}
```

Per usare un bucket S3 con la funzione Versioni multiple attivata, aggiungi l'operazione `s3:GetObjectVersion`, come illustrato nell'esempio seguente.

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

```
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket3/*"
            ]
        }
    ]
```

Ruoli IAM Amazon Rekognition

Un ruolo IAM è un'entità all'interno del tuo account che dispone di autorizzazioni specifiche. AWS

Utilizzo delle credenziali temporanee con Amazon Rekognition

È possibile utilizzare credenziali temporanee per effettuare l'accesso con la federazione, assumere un ruolo IAM o un ruolo multi-account. Ottieni credenziali di sicurezza temporanee chiamando operazioni AWS STS API come AssumeRoleo. GetFederationToken

Amazon Rekognition supporta l'uso delle credenziali temporanee.

Ruoli collegati ai servizi

I ruoli collegati ai AWS servizi consentono ai servizi di accedere alle risorse di altri servizi per completare un'azione per conto dell'utente. I ruoli collegati ai servizi sono visualizzati nell'account IAM e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non può modificarle.

Amazon Rekognition non supporta i ruoli collegati ai servizi.

Ruoli dei servizi

Questa caratteristica consente a un servizio di assumere un ruolo di servizio per conto dell'utente. Questo ruolo consente al servizio di accedere alle risorse in altri servizi per completare un'azione per conto dell'utente. I ruoli dei servizi sono visualizzati nell'account IAM e sono di proprietà dell'account. Ciò significa che un amministratore IAM può modificare le autorizzazioni per questo ruolo. Tuttavia, questo potrebbe pregiudicare la funzionalità del servizio.

Amazon Rekognition supporta i ruoli di servizio.

L'utilizzo di un ruolo di servizio può creare un problema di sicurezza in quanto Amazon Rekognition viene utilizzato per chiamare un altro servizio e agire su risorse a cui non dovrebbe avere accesso. Per tenere il tuo account al sicuro, dovrai limitare l'ambito di accesso di Amazon Rekognition alle sole risorse che stai utilizzando. Questo può essere fatto associando una policy di attendibilità al ruolo di

servizio IAM. Per informazioni su come fare, consulta Prevenzione del problema "confused deputy" tra servizi.

Scelta di un ruolo IAM in Amazon Rekognition

Quando si configura Amazon Rekognition per l'analisi dei video archiviati, è necessario scegliere un ruolo per consentire a Amazon Rekognition di accedere a Amazon Rekognition per conto dell'utente. Se hai creato in precedenza un ruolo di servizio o un ruolo collegato ai servizi, Amazon Rekognition fornisce un elenco di ruoli tra cui scegliere. Per ulteriori informazioni, consulta the section called "Configurazione di Video Amazon Rekognition".

Esempio: configurazione di Amazon Rekognition per accedere alle immagini in un bucket Amazon S3

Di seguito è riportato un esempio di come è possibile configurare Amazon Rekognition per l'analisi delle immagini in un bucket Amazon S3. Se desideri utilizzare Amazon Rekognition per analizzare le immagini in un bucket Amazon S3, devi fare quanto segue:

1. Assicurati che il tuo utente/ruolo IAM (il cliente) sia autorizzato a chiamare le operazioni API Amazon Rekognition pertinenti (ad esempio, ecc.) DetectLabels DetectFaces

   Allega una policy basata sull'identità che conceda le autorizzazioni appropriate per richiamare le operazioni API desiderate. Ad esempio, per concedere al proprio ruolo le autorizzazioni di chiamata `DetectLabels` e`DetectFaces`, è necessario allegare al proprio ruolo una policy simile alla seguente:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:DetectLabels",
                "rekognition:DetectFaces"
                // other Rekognition permissions as needed
            ],
            "Resource": "*"
        }
    ]
}
```

2. Il servizio Amazon Rekognition richiede l'autorizzazione per accedere al tuo bucket Amazon S3. Crea un ruolo di servizio IAM, che dovrai passare ad Amazon Rekognition quando effettui chiamate API. Assicurati che il ruolo del servizio: si fida del responsabile `s3:GetObject` del servizio Amazon Rekognition, disponga delle autorizzazioni per il tuo bucket.

   La politica di fiducia potrebbe essere simile alla seguente:

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

   La politica basata sull'identità associata al ruolo di servizio potrebbe essere simile alla seguente:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

AWS politiche gestite per Amazon Rekognition

Per aggiungere autorizzazioni a utenti, gruppi e ruoli, è più facile utilizzare le politiche AWS gestite che scrivere le politiche da soli. Creare policy gestite dal cliente IAM per fornire al tuo team solo le autorizzazioni di cui ha bisogno richiede tempo e competenza. Per iniziare rapidamente, puoi utilizzare le nostre politiche AWS gestite. Queste politiche coprono casi d'uso comuni e sono disponibili nel tuo AWS account. Per ulteriori informazioni sulle policy AWS gestite, consulta le policy AWS gestite nella IAM User Guide.

AWS i servizi mantengono e aggiornano le politiche AWS gestite. Non è possibile modificare le autorizzazioni nelle politiche AWS gestite. I servizi occasionalmente aggiungono altre autorizzazioni a una policy gestita da AWS per supportare nuove funzionalità. Questo tipo di aggiornamento interessa tutte le identità (utenti, gruppi e ruoli) a cui è collegata la policy. È più probabile che i servizi aggiornino una policy gestita da AWS quando viene avviata una nuova funzionalità o quando diventano disponibili nuove operazioni. I servizi non rimuovono le autorizzazioni da una policy AWS gestita, quindi gli aggiornamenti delle policy non comprometteranno le autorizzazioni esistenti.

Inoltre, AWS supporta politiche gestite per le funzioni lavorative che si estendono su più servizi. Ad esempio, la policy ReadOnlyAccess AWS gestita fornisce l'accesso in sola lettura a tutti i AWS servizi e le risorse. Quando un servizio lancia una nuova funzionalità, AWS aggiunge autorizzazioni di sola lettura per nuove operazioni e risorse. Per l'elenco e la descrizione delle policy di funzione dei processi, consulta la sezione Policy gestite da AWS per funzioni di processi nella Guida per l'utente di IAM.

Policy gestita da AWS: AmazonRekognitionFullAccess

`AmazonRekognitionFullAccess` assegna l'accesso completo alle risorse Amazon Rekognition, incluse la creazione e l'eliminazione di raccolte.

È possibile allegare la policy `AmazonRekognitionFullAccess` alle identità IAM.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "rekognition:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Policy gestita da AWS: AmazonRekognitionReadOnlyAccess

`AmazonRekognitionReadOnlyAccess` assegna l'accesso in sola lettura alle risorse Amazon Rekognition.

È possibile allegare la policy `AmazonRekognitionReadOnlyAccess` alle identità IAM.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

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

Policy gestita da AWS: AmazonRekognitionServiceRole

`AmazonRekognitionServiceRole` consente ad Amazon Rekognition di chiamare Flusso di dati Amazon Kinesis e i servizi Amazon SNS per tuo conto.

È possibile allegare la policy `AmazonRekognitionServiceRole` alle identità IAM.

Se utilizzi questo ruolo di servizio, dovresti proteggere il tuo account limitando l'ambito di accesso di Amazon Rekognition alle sole risorse che stai utilizzando. Questo può essere fatto associando una policy di attendibilità al ruolo di servizio IAM. Per informazioni su come fare, consulta Prevenzione del problema "confused deputy" tra servizi.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

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

Policy gestita da AWS: AmazonRekognitionCustomLabelsFullAccess

Questa politica è per gli utenti di Etichette personalizzate Amazon Rekognition. Utilizza la AmazonRekognitionCustomLabelsFullAccess policy per consentire agli utenti l'accesso completo all'API Amazon Rekognition Custom Labels e l'accesso completo ai bucket di console creati dalla console Amazon Rekognition Custom Labels.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

```
{
    "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 aggiorna le policy AWS gestite

Visualizza i dettagli sugli aggiornamenti delle politiche AWS gestite per Amazon Rekognition da quando questo servizio ha iniziato a tracciare queste modifiche. Per gli avvisi automatici sulle modifiche apportate a questa pagina, sottoscrivere il feed RSS nella pagina di Cronologia dei documenti di Amazon Rekognition.

Modifica	Descrizione	Data
Le azioni che coinvolgono lavori di analisi dei media sono state aggiunte alla seguente politica gestita: • Policy gestita da AWS: AmazonRekognitionReadOnlyAccess	Amazon Rekognition ha aggiunto le seguenti operazioni alla policy gestita `AmazonRekognitionReadOnlyAccess` : • `GetMediaAnalysisJob` • `ListMediaAnalysisJob`	31 ottobre 2023
Le azioni che coinvolgono la gestione degli utenti sono state aggiunte alla seguente politica gestita: • Policy gestita da AWS: AmazonRekognitionReadOnlyAccess	Amazon Rekognition ha aggiunto le seguenti operazioni alla policy gestita `AmazonRekognitionReadOnlyAccess` : • `ListUsers` • `SearchUsers` • `SearchUsersByImage`	12 giugno 2023
Actions for ProjectPolicy e Custom Labels Model Copy sono state aggiunte alle seguenti politiche gestite: • Policy gestita da AWS: AmazonRekognitionFullAccess	Amazon Rekognition ha aggiunto le seguenti operazioni alla policy gestita `AmazonRekognitionCustomLabelsFullAccess` e `AmazonRekognitionFullAccess` :	21 luglio 2022

Modifica	Descrizione	Data
• Policy gestita da AWS: AmazonRekognitionCustomLabelsFullAccess	• `CopyProjectVersion` • `PutProjectPolicy` • `ListProjectPolicies` • `DeleteProjectPolicy`	
Actions for ProjectPolicy e Custom Labels Model Copy sono state aggiunte alle seguenti politiche gestite: • Policy gestita da AWS: AmazonRekognitionReadOnlyAccess	Amazon Rekognition ha aggiunto le seguenti operazioni alla policy gestita AmazonRekognitionReadOnlyAccess : • `ListProjectPolicies`	21 luglio 2022
Aggiornamento della gestione dei set di dati per le seguenti policy gestite: • Policy gestita da AWS: AmazonRekognitionReadOnlyAccess • Policy gestita da AWS: AmazonRekognitionFullAccess • Policy gestita da AWS: AmazonRekognitionCustomLabelsFullAccess	Amazon Rekognition ha aggiunto le seguenti azioni AmazonRekognitionReadOnlyAccess alle AmazonRekognitionFullOnlyAccess politiche e gestite AmazonRekognitionCustomLabelsFullAccess • `CreateDataset` • `ListDatasetEntries` • `ListDatasetLabels` • `DescribeDataset` • `UpdateDatasetEntries` • `DistributeDatasetEntries` • `DeleteDataset`	1° novembre 2021

Modifica	Descrizione	Data
Aggiornamento dei tag per Policy gestita da AWS: AmazonRekognitionReadOnlyAccess e Policy gestita da AWS: AmazonRekognitionFullAccess	Amazon Rekognition ha aggiunto nuove azioni di tagging alle policy e. AmazonRekognitionFullAccess AmazonRekognitionReadOnlyAccess	2 aprile 2021
Amazon Rekognition ha iniziato a monitorare le modifiche	Amazon Rekognition ha iniziato a tracciare le modifiche alle sue politiche gestite. AWS	2 aprile 2021

Best practice per le policy

Le policy basate sull'identità determinano se qualcuno può creare, accedere o eliminare risorse Amazon Rekognition all'interno dell'account. Queste operazioni possono comportare costi aggiuntivi per l' Account AWS. Quando crei o modifichi policy basate su identità, segui queste linee guida e raccomandazioni:

- Inizia con le politiche AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche gestite che concedono le autorizzazioni per molti casi d'uso comuni.AWS Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per ulteriori informazioni, consulta Policy gestite da AWSo Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente IAM.
- Applica le autorizzazioni con privilegio minimo: quando imposti le autorizzazioni con le policy IAM, concedi solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegi minimi. Per ulteriori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso: per limitare l'accesso a operazioni e risorse è possibile aggiungere una condizione alle tue policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio AWS CloudFormation. Per

ulteriori informazioni, consulta la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente IAM.

- Utilizzo di IAM Access Analyzer per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali: IAM Access Analyzer convalida le policy nuove ed esistenti in modo che aderiscano alla sintassi della policy IAM (JSON) e alle best practice di IAM. IAM Access Analyzer offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per ulteriori informazioni, consulta Convalida delle policy per il Sistema di analisi degli accessi IAM nella Guida per l'utente IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungi le condizioni MFA alle policy. Per ulteriori informazioni, consulta Protezione dell'accesso API con MFA nella Guida per l'utente IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Utilizzo della console Amazon Rekognition

Ad eccezione della funzionalità Etichette personalizzate Amazon Rekognition, Amazon Rekognition non richiede alcuna autorizzazione di aggiunta quando si utilizza la console Amazon Rekognition. Per informazioni su Etichette personalizzate Amazon Rekognition, consulta il passaggio 5: Impostare le autorizzazioni della console Etichette personalizzate Amazon Rekognition.

Non è necessario consentire autorizzazioni minime per la console per gli utenti che effettuano chiamate solo verso o l' AWS CLI API. AWS Al contrario, è possibile accedere solo alle operazioni che soddisfano l'operazione API che stai cercando di eseguire.

Esempi di utilizzo di policy basate sull'identità di Amazon Rekognition

Per impostazione predefinita, gli utenti e i ruoli IAM non dispongono dell'autorizzazione per creare o modificare risorse Amazon Rekognition. Inoltre, non possono eseguire attività utilizzando l'API AWS Management Console AWS CLI, o. AWS Un amministratore IAM deve creare policy IAM che concedono a utenti e ruoli l'autorizzazione per eseguire operazioni API specifiche sulle risorse specificate di cui hanno bisogno. L'amministratore devi quindi collegare queste policy a utenti o gruppi che richiedono tali autorizzazioni.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy nella scheda JSON nella Guida per l'utente IAM.

Argomenti

- Esempi di policy Etichette personalizzate Amazon Rekognition
- Esempio 1: assegnazione a un utente dell'accesso in sola lettura alle risorse
- Esempio 2: assegnazione a un utente dell'accesso completo alle risorse
- Consentire agli utenti di visualizzare le loro autorizzazioni

Esempi di policy Etichette personalizzate Amazon Rekognition

È possibile creare criteri basati sull'identità per Etichette personalizzate Amazon Rekognition. Per ulteriori informazioni, consulta Sicurezza.

Esempio 1: assegnazione a un utente dell'accesso in sola lettura alle risorse

L'esempio seguente consente l'accesso in modalità di sola lettura alle risorse Amazon Rekognition.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:CompareFaces",
                "rekognition:DetectFaces",
                "rekognition:DetectLabels",
                "rekognition:ListCollections",
                "rekognition:ListFaces",
                "rekognition:SearchFaces",
                "rekognition:SearchFacesByImage",
                "rekognition:DetectText",
                "rekognition:GetCelebrityInfo",
                "rekognition:RecognizeCelebrities",
                "rekognition:DetectModerationLabels",
                "rekognition:GetLabelDetection",
                "rekognition:GetFaceDetection",
                "rekognition:GetContentModeration",
                "rekognition:GetPersonTracking",
```

```
                "rekognition:GetCelebrityRecognition",
                "rekognition:GetFaceSearch",
                "rekognition:GetTextDetection",
                "rekognition:GetSegmentDetection",
                "rekognition:DescribeStreamProcessor",
                "rekognition:ListStreamProcessors",
                "rekognition:DescribeProjects",
                "rekognition:DescribeProjectVersions",
                "rekognition:DetectCustomLabels",
                "rekognition:DetectProtectiveEquipment",
                "rekognition:ListTagsForResource",
               "rekognition:ListDatasetEntries",
                "rekognition:ListDatasetLabels",
                "rekognition:DescribeDataset"

            ],
            "Resource": "*"
        }
    ]
}
```

Esempio 2: assegnazione a un utente dell'accesso completo alle risorse

L'esempio seguente consente l'accesso completo alle risorse Amazon Rekognition.

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

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono cpllegate alla relativa identità utente. Questa politica

include le autorizzazioni per completare questa azione sulla console o utilizzando l'API o a livello di codice. AWS CLI 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": "*"
        }
    ]
}
```

Esempi di policy basate sulle risorse di Amazon Rekognition

Etichette personalizzate Amazon Rekognition utilizza una policy basata su risorse, nota come policy di progetto, per gestire le autorizzazioni di copia per una versione del modello.

Una policy di progetto consente o nega un'autorizzazione a copiare una versione del modello da un progetto di origine a un progetto di destinazione. Hai bisogno di una policy di progetto se il progetto di

destinazione si trova in un AWS account diverso o se desideri limitare l'accesso all'interno di un AWS account. Ad esempio, potresti voler negare le autorizzazioni di copia a un ruolo IAM specifico. Per ulteriori informazioni, consulta Copia di un modello.

Concessione dell'autorizzazione per copiare la versione di un modello

L'esempio seguente consente al principale `arn:aws:iam::123456789012:role/Admin` di copiare la versione del modello `arn:aws:rekognition:us-east-1:123456789012:project/my_project/version/test_1/1627045542080`.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Principal":{
        "AWS":"arn:aws:iam::123456789012:role/Admin"
      },
      "Action":"rekognition:CopyProjectVersion",
      "Resource":"arn:aws:rekognition:us-east-1:123456789012:project/my_project/
version/test_1/1627045542080"
    }
  ]
}
```

Risoluzione dei problemi relativi all'identità e all'accesso di Amazon Rekognition

Utilizza le informazioni seguenti per diagnosticare e risolvere i problemi comuni che possono verificarsi durante l'utilizzo di Amazon Rekognition e IAM.

Argomenti

- Non dispongo dell'autorizzazione per eseguire un'operazione in Amazon Rekognition
- Non sono autorizzato a eseguire iam: PassRole
- Sono un amministratore e desidero consentire ad altri utenti di accedere a Amazon Rekognition
- Voglio consentire a persone esterne al mio AWS account di accedere alle mie risorse Amazon Rekognition

Non dispongo dell'autorizzazione per eseguire un'operazione in Amazon Rekognition

Se ti AWS Management Console dice che non sei autorizzato a eseguire un'azione, devi contattare l'amministratore per ricevere assistenza. L'amministratore è colui che ti ha fornito le credenziali di accesso.

L'errore di esempio seguente si verifica quando l'utente IAM `mateojackson` tenta di utilizzare la console per visualizzare i dettagli relativi a *`widget`*, ma non dispone delle autorizzazioni `rekognition:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 rekognition:GetWidget on resource: my-example-widget
```

In questo caso, Mateo richiede al suo amministratore di aggiornare le policy per poter accedere alla risorsa *`my-example-widget`* utilizzando l'azione `rekognition:`*`GetWidget`*.

Non sono autorizzato a eseguire iam: PassRole

Se si riceve un errore che indica che non si è autorizzati a eseguire l'operazione `iam:PassRole`, è necessario aggiornare le policy per poter passare un ruolo ad Amazon Rekognition.

Alcuni Servizi AWS consentono di passare un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

Il seguente esempio di errore si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in Amazon Rekognition. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per passare il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Sono un amministratore e desidero consentire ad altri utenti di accedere a Amazon Rekognition

Per consentire ad altri di accedere ad Amazon Rekognition, devi concedere l'autorizzazione alle persone o alle applicazioni che devono accedervi. Se lo utilizzi AWS IAM Identity Center per gestire persone e applicazioni, assegni set di autorizzazioni a utenti o gruppi per definirne il livello di accesso. I set di autorizzazioni creano e assegnano automaticamente le policy IAM ai ruoli IAM associati alla persona o all'applicazione. Per ulteriori informazioni, consulta Set di autorizzazioni nella Guida per l'AWS IAM Identity Center utente.

Se non utilizzi IAM Identity Center, devi creare entità IAM (utenti o ruoli) per le persone o le applicazioni che necessitano di accesso. Dovrai quindi collegare all'entità una policy che conceda le autorizzazioni corrette in Amazon Rekognition. Dopo aver concesso le autorizzazioni, fornisci le credenziali all'utente o allo sviluppatore dell'applicazione. Utilizzeranno tali credenziali per accedere. AWSPer ulteriori informazioni sulla creazione di utenti, gruppi, policy e autorizzazioni IAM, consulta IAM Identities and Policies and permissions in IAM nella IAM User Guide.

Voglio consentire a persone esterne al mio AWS account di accedere alle mie risorse Amazon Rekognition

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche per consentire alle persone di accedere alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per sapere se Amazon Rekognition supporta queste funzionalità, consultare Come funziona Amazon Rekognition con IAM.
- Per scoprire come fornire l'accesso alle risorse di tua proprietà, consulta Fornire l'accesso a un utente IAM in Account AWS un altro Account AWS di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a soggetti Account AWS di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (Federazione delle identità) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Protezione dei dati in Amazon Rekognition

Il modello di responsabilità AWS condivisa Modello si applica alla protezione dei dati in Amazon Rekognition. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i. Cloud AWS L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, vedi le Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al Modello di responsabilità condivisa AWS e GDPR nel Blog sulla sicurezza AWS .

Ai fini della protezione dei dati, consigliamo di proteggere Account AWS le credenziali e configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Usa SSL/TLS per comunicare con le risorse. AWS È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con. AWS CloudTrail Per informazioni sull'utilizzo dei CloudTrail percorsi per acquisire AWS le attività, consulta Lavorare con i CloudTrail percorsi nella Guida per l'AWS CloudTrail utente.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-3 per accedere AWS tramite un'interfaccia a riga di comando o un'API, usa un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-3.

Ti consigliamo di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori con Rekognition o Servizi AWS altri utenti utilizzando la console, l'API o. AWS CLI AWS SDKs I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando fornisci un URL a un server esterno, ti suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta al server.

Crittografia dei dati

Le seguenti informazioni spiegano dove Amazon Rekognition utilizza la crittografia dei dati per proteggere i dati.

Crittografia a riposo

Immagini Amazon Rekognition

Immagini

Le immagini trasferite alle operazioni dell'API Amazon Rekognition possono essere archiviate e utilizzate per migliorare il servizio, a meno che tu non abbia effettuato l'opt-out visitando la pagina della policy di rifiuto dei servizi IA e seguendo la procedura ivi spiegata. Le immagini archiviate vengono crittografate a riposo (Amazon S3) utilizzando AWS Key Management Service (SSE-KMS).

Raccolte

Per le operazioni di confronto di volti che memorizzano le informazioni in una raccolta, l'algoritmo di rilevamento sottostante rileva prima i volti nell'immagine di input, estrae un vettore per ogni volto e quindi archivia i vettori di volti nella raccolta. Amazon Rekognition utilizza questi vettori di volti quando esegue i confronti tra volti. I vettori di volti sono archiviati come una matrice di float e crittografati a riposo.

Video Amazon Rekognition

Video

Per analizzare un video, Amazon Rekognition copia i tuoi video nel servizio per l'elaborazione. Il video può essere archiviato e utilizzato per migliorare il servizio, a meno che tu non abbia effettuato l'opt-out visitando la pagina della policy di rifiuto dei servizi IA e seguendo la procedura ivi spiegata. I video vengono crittografati a riposo (Amazon S3) utilizzando AWS Key Management Service (SSE-KMS).

Etichette personalizzate Amazon Rekognition

Etichette personalizzate Amazon Rekognition esegue la crittografia dei dati a riposo.

Immagini

Per addestrare il tuo modello, Etichette personalizzate Amazon Rekognition crea una copia delle immagini di addestramento e di test di origine. Le immagini copiate vengono crittografate a riposo in Amazon Simple Storage Service (S3) utilizzando la crittografia lato server con una chiave KMS

fornita dall'utente o una chiave KMS di AWS KMS key proprietà. AWS Etichette personalizzate Amazon Rekognition supporta solo chiavi KMS simmetriche. Le tue immagini di origine non vengono modificate. Per ulteriori informazioni, consulta Addestramento di un modello Etichette personalizzate Amazon Rekognition.

Modelli

Per impostazione predefinita, Etichette personalizzate Amazon Rekognition esegue la crittografia dei modelli addestrati e dei file manifesto archiviati nei bucket Amazon S3 utilizzando la crittografia lato server con una Chiave di proprietà di AWS. Per ulteriori informazioni, consulta Protezione dei dati con la crittografia lato server. I risultati dell'allenamento vengono scritti nel bucket specificato nel parametro di input to. `OutputConfig` CreateProjectVersion I risultati dell'addestramento vengono crittografati utilizzando le impostazioni di crittografia configurate per il bucket (`OutputConfig`).

Bucket della console

La console Etichette personalizzate Amazon Rekognition crea un bucket Amazon S3 (bucket della console) che puoi utilizzare per gestire i tuoi progetti. Il bucket della console è crittografato utilizzando la crittografia Amazon S3 predefinita. Per ulteriori informazioni, consulta Crittografia predefinita di Amazon Simple Storage Service per i bucket S3. Se utilizzi la tua chiave KMS, configura il bucket della console dopo averlo creato. Per ulteriori informazioni, consulta Protezione dei dati con la crittografia lato server. Etichette personalizzate Amazon Rekognition blocca l'accesso pubblico al bucket della console.

Rekognition Face Liveness

Tutti i dati relativi alla sessione archiviati nell'account del servizio Rekognition Face Liveness sono completamente crittografati quando sono a riposo. Per impostazione predefinita, le immagini di riferimento e di controllo sono crittografate utilizzando una chiave di proprietà di AWS nell'account del servizio. Tuttavia, puoi scegliere di fornire AWS KMS le tue chiavi per crittografare queste immagini.

Crittografia in transito

Gli endpoint API di Amazon Rekognition supportano solo connessioni protette tramite HTTPS. Tutte le comunicazioni sono crittografate con Transport Layer Security (TLS).

Gestione delle chiavi

È possibile utilizzare AWS Key Management Service (KMS) per gestire le chiavi per le immagini e i video di input archiviati nei bucket Amazon S3. Per ulteriori informazioni, consulta Concetti di AWS Key Management Service.

Crittografia a chiave gestita dal cliente per Face Liveness

L'CreateFaceLivenessSessionAPI accetta un `KmsKeyId` parametro opzionale. Puoi fornire l'`id` della chiave KMS creata nel tuo account. Questa chiave verrà utilizzata per crittografare le immagini di riferimento e di controllo ottenute durante l'StartFaceLivenessSessionAPI e durante l'GetFaceLivenessSessionResultsAPI le immagini verranno decrittografate utilizzando questa chiave prima di restituire i risultati. Se la CreateFaceLivenessSession richiesta include un OutputConfig, le immagini di riferimento e di controllo verranno caricate nei percorsi Amazon S3 specificati. Ti consigliamo di abilitare la crittografia lato server (SSE-S3) nei bucket Amazon S3 in modo che i dati continuino a rimanere crittografati anche quando sono a riposo.

Quando fornisci il tuo ID AWS KMS chiave, il servizio Rekognition Face Liveness ottiene l'autorizzazione a utilizzare la chiave gestita dal cliente per conto del principale che richiama il. APIs I principali (utenti o ruoli) utilizzati per richiamare il backend del cliente (APIs`CreateFaceLivenessSession`e`GetFaceLivenessSessionResults`) APIs devono avere accesso per eseguire le seguenti operazioni:

- kms:DescribeKey
- kms:GenerateDataKey
- kms:Decrypt

Riservatezza del traffico Internet

Un endpoint Amazon Virtual Private Cloud (Amazon VPC) per Amazon Rekognition è un'entità logica all'interno di un VPC che consente la connettività solo ad Amazon Rekognition. Amazon VPC instrada le richieste ad Amazon Rekognition e reindirizza le risposte al VPC. Per ulteriori informazioni, consulta Endpoint VPC nella Guida per l'utente di Amazon VPC. Per informazioni sull'utilizzo degli endpoint VPC Amazon con Amazon Rekognition, consulta Utilizzo di Amazon Rekognition con endpoint VPC Amazon .

Utilizzo di Amazon Rekognition con endpoint VPC Amazon

Se usi Amazon Virtual Private Cloud (Amazon VPC) per l'hosting delle risorse AWS, puoi stabilire una connessione privata tra il VPC e Amazon Rekognition. Puoi utilizzare questa connessione per permettere ad Amazon Rekognition di comunicare con le risorse nel VPC senza accedere alla rete Internet pubblica.

Amazon VPC è un servizio AWS che puoi utilizzare per avviare risorse AWS in una rete virtuale da te definita. Con un VPC;, detieni il controllo delle impostazioni della rete, come l'intervallo di indirizzi IP, le sottoreti, le tabelle di routing e i gateway di rete. Con gli endpoint VPC, la rete AWS gestisce il routing tra il VPC e i servizi AWS.

Per collegare il tuo VPC a Amazon Rekognition, definisci un endpoint VPC dell'interfaccia per Amazon Rekognition. Un endpoint di interfaccia è un'interfaccia di rete elastica con un indirizzo IP privato che funge da punto di ingresso per il traffico destinato a un servizio supportato AWS . L'endpoint offre una connettività scalabile e affidabile a Amazon Rekognition e non richiede un gateway Internet, un'istanza Network Address Translation (NAT) o una connessione VPN. Per ulteriori informazioni, consulta Che cos'è Amazon VPC? nella Guida per l'utente Amazon VPC.

Gli endpoint VPC dell'interfaccia sono abilitati da AWS. PrivateLink Questa tecnologia AWS consente la comunicazione privata tra i servizi AWS utilizzando un'interfaccia di rete elastica con indirizzi IP privati.

> **Note**
>
> Tutti gli endpoint FIPS (Federal Information Processing Standard) di Amazon Rekognition sono supportati da AWS. PrivateLink

Creazione endpoint VPC Amazon per Amazon Rekognition

Puoi creare due tipi di endpoint VPC Amazon da utilizzare con Amazon Rekognition.

- Un endpoint VPC da utilizzare con le operazioni Amazon Rekognition. Per la maggior parte degli utenti, questo è il tipo di endpoint VPC più adatto.
- Un endpoint VPC per le operazioni Amazon Rekognition con endpoint conformi allo standard federale Federal Information Processing Standard (FIPS) Publication 140-2 degli Stati Uniti.

Per iniziare a usare Amazon Rekognition con il VPC, utilizza la console Amazon VPC per creare un endpoint VPC di interfaccia per Amazon Rekognition. Per istruzioni, consulta la procedura "Per creare un endpoint di interfaccia a un servizio AWS tramite la console" in Creazione di un endpoint di interfaccia. Tieni presente le seguenti fasi della procedura:

- Fase 3 – Per Categoria servizio, scegli Servizi AWS.
- Fase 4 – Per Nome servizio, scegli una delle seguenti opzioni:

- com.amazonaws.region.rekognition – Crea un endpoint VPC per le operazioni Amazon Rekognition.
- com.amazonaws.region.rekognition-fips – Crea un endpoint VPC per le operazioni Amazon Rekognition con endpoint conformi allo standard federale Federal Information Processing Standard (FIPS) Publication 140-2 Stati Uniti.

Per ulteriori informazioni, consulta l'argomento relativo alle nozioni di base nella Guida per l'utente di Amazon VPC.

Creazione di una policy di endpoint VPC per Amazon Rekognition

Puoi creare una policy per gli endpoint VPC Amazon per Amazon Rekognition per specificare quanto segue:

- Il principale che può eseguire azioni.
- Le azioni che possono essere eseguite.
- Le risorse sui cui si possono eseguire operazioni.

Per ulteriori informazioni, consultare Controllo degli accessi ai servizi con endpoint VPC nella Guida per l'utente di Amazon VPC.

Il criterio di esempio seguente consente agli utenti che si connettono a Amazon Rekognition tramite l'endpoint VPC di chiamare l'operazione `DetectFaces` API. Il criterio impedisce agli utenti di eseguire altre operazioni Amazon Rekognition API tramite l'endpoint VPC.

Gli utenti possono ancora chiamare altre operazioni Amazon Rekognition API dall'esterno del VPC. Per informazioni su come negare l'accesso alle operazioni Amazon Rekognition API esterne al VPC, vedere Policy basate su identità di Amazon Rekognition.

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

```
            "Principal": "*"
        }
    ]
}
```

Per modificare la policy endpoint VPC per Amazon Rekognition

1. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Se non hai già creato l'endpoint per Amazon Rekognition, scegli Crea endpoint. Quindi seleziona com.amazonaws. *Region*.rekognition e scegli Crea endpoint.
3. Nel pannello di navigazione, seleziona Endpoint.
4. Seleziona com.amazonaws. ***Region***.rekognition endpoint e scegli la scheda Policy nella metà inferiore dello schermo.
5. Scegli Edit Policy (Modifica policy) e apporta le modifiche alla policy.

Convalida della conformità per Amazon Rekognition

I revisori di terze parti valutano la sicurezza e la conformità di Amazon Rekognition nell'ambito di diversi programmi di conformità. AWS Questi includono SOC, PCI, FedRAMP, HIPAA e altri.

Per un elenco di AWS servizi nell'ambito di programmi di conformità specifici, consulta AWS Services in Scope by Compliance Program. Per informazioni generali, consulta Programmi di conformità di AWS.

Puoi scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Download dei rapporti in AWS Artifact.

La tua responsabilità di conformità quando usi Amazon Rekognition è determinata dalla sensibilità dei tuoi dati, dagli obiettivi di conformità della tua azienda e dalle leggi e dai regolamenti applicabili. AWS fornisce le seguenti risorse per contribuire alla conformità:

- Guide introduttive su sicurezza e conformità: queste guide all'implementazione illustrano considerazioni sull'architettura e forniscono passaggi per implementare ambienti di base incentrati sulla sicurezza e la conformità. AWS
- Whitepaper sull'architettura per la sicurezza e la conformità HIPAA: questo white paper descrive in che modo le aziende possono utilizzare per creare applicazioni conformi allo standard HIPAA. AWS

- AWS Risorse per la conformità: questa raccolta di cartelle di lavoro e guide potrebbe riguardare il settore e la località in cui operi.
- AWS Config— Questo AWS servizio valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida del settore e alle normative.
- AWS Security Hub— Questo AWS servizio offre una visione completa dello stato di sicurezza dell'utente e consente AWS di verificare la conformità agli standard e alle best practice del settore della sicurezza.

Resilienza in Amazon Rekognition

L'infrastruttura AWS globale è costruita attorno a regioni e zone di disponibilità. AWS AWS Le regioni forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti. Con le zone di disponibilità, puoi progettare e gestire applicazioni e database che eseguono automaticamente il failover tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture a data center singolo o multiplo tradizionali.

Per ulteriori informazioni su AWS regioni e zone di disponibilità, consulta Global Infrastructure.AWS

Oltre all'infrastruttura AWS globale, Amazon Rekognition offre diverse funzionalità per supportare le tue esigenze di resilienza e backup dei dati.

Analisi della configurazione e delle vulnerabilità in Amazon Rekognition

Configurazione e controllo IT sono una responsabilità condivisa tra AWS e te, il nostro cliente. Per ulteriori informazioni, consulta il modello di responsabilità AWS condivisa.

Prevenzione del problema "confused deputy" tra servizi

In AWS, la rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante) chiama un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per agire sulle risorse di un altro cliente, anche se non dovrebbe avere le autorizzazioni corrette, e ciò porta al problema del "confused deputy".

Per evitare che ciò accada, AWS mette a disposizione strumenti che consentono di proteggere i dati relativi a tutti i servizi con responsabili del servizio a cui è stato concesso l'accesso alle risorse dell'account.

È consigliabile utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` nelle policy delle risorse per limitare le autorizzazioni con cui Amazon Rekognition fornisce un altro servizio alla risorsa.

Se il valore di `aws:SourceArn` non contiene l'ID account, ad esempio un ARN di un bucket Amazon S3, è necessario utilizzare entrambe le chiavi per limitare le autorizzazioni. Se si utilizzano entrambe le chiavi e il valore `aws:SourceArn` contiene l'ID account, il valore `aws:SourceAccount` e l'account nel valore `aws:SourceArn` deve utilizzare lo stesso ID account nella stessa dichiarazione di policy.

Utilizza `aws:SourceArn` se desideri consentire l'associazione di una sola risorsa all'accesso tra servizi. Utilizza `aws:SourceAccount` se desideri consentire l'associazione di qualsiasi risorsa in tale account all'uso tra servizi.

Il valore di `aws:SourceArn` deve essere l'ARN della risorsa utilizzata da Rekognition, che è specificata con il seguente formato: `arn:aws:rekognition:region:account:resource`.

Il valore di `arn:User ARN` dovrebbe essere l'ARN dell'utente che chiamerà l'operazione di analisi video (l'utente che assume un ruolo).

L'approccio consigliato per il problema "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa.

Se non conosci l'ARN completo della risorsa o scegli più risorse, utilizza la chiave `aws:SourceArn` con caratteri jolly (`*`) per le parti sconosciute dell'ARN. Ad esempio `arn:aws:rekognition:*:111122223333:*`.

Per proteggersi dal problema "confused deputy", completa le seguenti fasi:

1. Nel pannello di navigazione della console IAM seleziona l'opzione Ruoli. La console visualizzerà i ruoli del tuo account attuale.
2. Seleziona il nome del ruolo che desideri modificare. Il ruolo che modifichi dovrebbe avere la politica delle AmazonRekognitionServiceRoleautorizzazioni. Seleziona la scheda Relazioni di attendibilità.
3. Seleziona Modifica policy di attendibilità.

4. Nella pagina Modifica policy di attendibilità, sostituisci la policy JSON predefinita con una policy che utilizza una o entrambe le chiavi di contesto della condizione globale `aws:SourceArn` e `aws:SourceAccount`. Vedi le policy di esempio riportate di seguito.
5. Scegli Aggiorna policy.

Gli esempi seguenti contengono policy di attendibilità che mostrano il modo in cui puoi utilizzare le chiavi di contesto della condizione globale `aws:SourceArn` e `aws:SourceAccount` in Amazon Rekognition per prevenire il problema "confused deputy".

Se lavori con video archiviati e trasmessi in streaming, puoi utilizzare una policy come la seguente nel tuo ruolo IAM:

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

Se lavori esclusivamente con video archiviati, puoi utilizzare una policy come la seguente nel tuo ruolo IAM (tieni presente che non devi includere l'argomento `StringLike` che specifica il `streamprocessor`):

```
{
```

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

Sicurezza dell'infrastruttura in Amazon Rekognition

In quanto servizio gestito, Amazon Rekognition è AWS protetto dalla sicurezza di rete globale. Per informazioni sui servizi di AWS sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utilizza chiamate API AWS pubblicate per accedere ad Amazon Rekognition attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. In alternativa, è possibile utilizzare AWS Security Token Service (AWS STS) per generare le credenziali di sicurezza temporanee per sottoscrivere le richieste.

Monitoraggio di Amazon Rekognition

Il monitoraggio è importante per garantire l'affidabilità, la disponibilità e le prestazioni di Amazon Rekognition e delle altre soluzioni AWS. AWS fornisce gli strumenti di monitoraggio seguenti per tenere sotto controllo Rekognition, segnalare un problema e intervenire automaticamente quando necessario:

- Amazon CloudWatch monitora AWS le tue risorse e le applicazioni su cui esegui AWS in tempo reale. Puoi raccogliere i parametri e tenerne traccia, creare pannelli di controllo personalizzati e impostare allarmi per inviare una notifica o intraprendere azioni quando un parametro specificato raggiunge una determinata soglia. Ad esempio, puoi tenere CloudWatch traccia dell'utilizzo della CPU o di altri parametri delle tue EC2 istanze Amazon e avviare automaticamente nuove istanze quando necessario. Per ulteriori informazioni, consulta la Amazon CloudWatch User Guide.
- Amazon CloudWatch Logs ti consente di monitorare, archiviare e accedere ai tuoi file di registro da EC2 istanze Amazon e altre fonti. CloudTrail CloudWatch I log possono monitorare le informazioni contenute nei file di registro e avvisarti quando vengono raggiunte determinate soglie. Puoi inoltre archiviare i dati del log in storage estremamente durevole. Per ulteriori informazioni, consulta la Amazon CloudWatch Logs User Guide.
- Amazon EventBridge può essere utilizzato per automatizzare AWS i tuoi servizi e rispondere automaticamente agli eventi di sistema, come problemi di disponibilità delle applicazioni o modifiche delle risorse. Gli eventi AWS relativi ai servizi vengono forniti quasi EventBridge in tempo reale. Puoi compilare regole semplici che indichino quali eventi sono considerati di interesse per te e quali operazioni automatizzate intraprendere quando un evento corrisponde a una regola. Per ulteriori informazioni, consulta Amazon EventBridge User Guide.
- AWS CloudTrailacquisisce le chiamate API e gli eventi correlati effettuati da o per conto del tuo AWS account e invia i file di log a un bucket Amazon S3 da te specificato. Puoi identificare quali utenti e account hanno chiamato AWS, l'indirizzo IP di origine da cui sono state effettuate le chiamate e quando sono avvenute le chiamate. Per ulteriori informazioni, consulta la Guida per l'utente AWS CloudTrail.

Monitoraggio di Rekognition con Amazon CloudWatch

Con CloudWatch, puoi ottenere metriche per le singole operazioni di Rekognition o metriche globali di Rekognition per il tuo account, puoi utilizzare le metriche per monitorare lo stato della tua soluzione basata su Rekognition e impostare allarmi per avvisarti quando una o più metriche

superano una soglia definita. Ad esempio, puoi impostare parametri per una serie di errori del server che si sono verificati oppure per il numero di volti che sono stati rilevati. Puoi anche visualizzare parametri per il numero di volte che si è verificata una specifica operazione di Rekognition. Per visualizzare le metriche, puoi utilizzare Amazon CloudWatch AWS Command Line Interface, Amazon o l'CloudWatch API.

Puoi anche visualizzare parametri aggregati per un determinato periodo di tempo utilizzando la console Rekognition. Per ulteriori informazioni, consulta Esercizio 4: visualizzare i parametri aggregati (console).

Utilizzo delle CloudWatch metriche per Rekognition

Per utilizzare i parametri, devi specificare le seguenti informazioni:

- La dimensione del parametro o nessuna dimensione. Una dimensione è una coppia nome-valore che consente di identificare un parametro in modo univoco. In Rekognition è disponibile una dimensione denominata Operation, che fornisce parametri per un'operazione specifica. Se non specifichi una dimensione, l'ambito del parametro è rappresentato da tutte le operazioni di Rekognition nel tuo account.
- Il nome del parametro, ad esempio `UserErrorCount`.

Puoi ottenere i dati di monitoraggio per Rekognition utilizzando l', AWS Management Console AWS CLI la o l'API. CloudWatch Puoi anche utilizzare l' CloudWatch API tramite uno degli Amazon AWS Software Development Kit (SDKs) o gli strumenti CloudWatch API. La console mostra una serie di grafici basati sui dati grezzi dell' CloudWatch API. In base alle tue esigenze, potresti decidere di utilizzare i grafici visualizzati nella console o quelli recuperati dall'API.

L'elenco seguente mostra alcuni usi comuni dei parametri. Questi suggerimenti sono solo introduttivi e non costituiscono un elenco completo.

Come?	Parametri rilevanti
Come è possibile monitorare il numero di volti riconosciuti?	Monitora la statistica `Sum` del parametro `DetectedFaceCount` .

Come?	Parametri rilevanti
Come è possibile sapere se l'applicazione ha raggiunto il numero massimo di richieste al secondo?	Monitora la statistica `Sum` del parametro `ThrottledCount` .
Come è possibile monitorare gli errori di richiesta?	Utilizza la statistica `Sum` del parametro `UserErrorCount` .
Come è possibile trovare il numero totale di richieste?	Utilizza le statistiche `ResponseTime` e `Data Samples` del parametro `ResponseTime` . Questo parametro include tutte le richieste che generano un errore. Se desideri visualizzare solo le chiamate alle operazioni riuscite, utilizza il parametro `SuccessfulRequestCount` .
Come è possibile monitorare la latenza delle chiamate alle operazioni di `Rekognition` ?	Utilizza il parametro `ResponseTime` .
Come è possibile monitorare il numero di volte che `IndexFaces` aggiunge volti alle raccolte di Rekognition?	Monitora la statistica `Sum` con il parametro `SuccessfulRequestCount` e l'operazi one `IndexFaces` . Utilizza la dimensione `Operation` per selezionare l'operazione e il parametro.

È necessario disporre delle CloudWatch autorizzazioni appropriate per monitorare Rekognition. CloudWatch Per ulteriori informazioni, vedi Autenticazione e controllo degli accessi per Amazon CloudWatch).

Accesso ai parametri di Rekognition

Gli esempi seguenti mostrano come accedere alle metriche di Rekognition utilizzando la console, l'e CloudWatch l'API. AWS CLI CloudWatch

Come visualizzare i parametri (console)

1. Apri la console all'indirizzo. CloudWatch https://console.aws.amazon.com/cloudwatch/
2. Scegli Metrics (Parametri), scegli la scheda All Metrics (Tutti i parametri), quindi Rekognition.

3. Scegli Metrics with no dimensions (Parametri senza dimensioni), quindi scegli un parametro.

   Ad esempio, scegli la DetectedFacemetrica per misurare quanti volti sono stati rilevati.

4. Seleziona un valore per l'intervallo di date. Il conteggio dei parametri viene visualizzato nel grafico.

Per visualizzare i parametri delle chiamate all'operazione **`DetectFaces`** eseguite correttamente in un periodo di tempo (CLI)

- Aprite AWS CLI e immettete il seguente comando:

  ```
  aws cloudwatch get-metric-statistics --metric-name
  SuccessfulRequestCount --start-time 2017-1-1T19:46:20 --end-time
  2017-1-6T19:46:57 --period 3600 --namespace AWS/Rekognition --
  statistics Sum --dimensions Name=Operation,Value=DetectFaces --region
  us-west-2
  ```

  Questo esempio mostra le chiamate all'operazione `DetectFaces` eseguite correttamente in un periodo di tempo. Per ulteriori informazioni, consulta get-metric-statistics.

Per accedere alle metriche (CloudWatch API)

- Chiama `GetMetricStatistics`. Per ulteriori informazioni, consulta Amazon CloudWatch API Reference.

Creazione di un allarme

Puoi creare un CloudWatch allarme che invia un messaggio Amazon Simple Notification Service (Amazon SNS) quando l'allarme cambia stato. Un allarme controlla un singolo parametro in un periodo di tempo specificato ed esegue una o più operazioni in base al valore del parametro relativo a una determinata soglia in una serie di periodi di tempo. L'operazione corrisponde all'invio di una notifica a un argomento di Amazon SNS o a una policy di Auto Scaling.

Gli allarmi richiamano azioni solo per cambiamenti di stato sostenuti. CloudWatch gli allarmi non richiamano azioni semplicemente perché si trovano in uno stato particolare. È necessario che lo stato cambi e rimanga costante per un periodo specificato

Per impostare un allarme (console)

1. Accedi a AWS Management Console e apri la CloudWatch console all'indirizzo. https://console.aws.amazon.com/cloudwatch/
2. Scegli Crea allarme. Viene avviata la procedura guidata per la creazione di allarmi.
3. Nell'elenco Metrics with no dimensions (Parametri senza dimensioni), scegli Rekognition Metrics (Parametri Rekognition), quindi scegli un parametro.

   Ad esempio, scegli DetectedFaceCountdi impostare un allarme per un numero massimo di volti rilevati.
4. Nell'area Time Range (Intervallo di tempo), seleziona un valore per l'intervallo di tempo che includa le operazioni di rilevamento facciale che hai richiamato. Seleziona Next (Successivo).
5. Compila i campi Name (Nome) e Description (Descrizione). Per Whenever (Ogni volta), scegli **>=** e inserisci un valore massimo che preferisci.
6. Se desideri CloudWatch inviarti un'e-mail quando viene raggiunto lo stato di allarme, per Ogni volta che questo allarme:, scegli State is ALARM. Per inviare allarmi a un argomento Amazon SNS esistente, in Invia notifica a:, scegli un argomento SNS esistente. Per impostare il nome e gli indirizzi e-mail per un nuovo elenco di sottoscrizioni e-mail, scegli Crea argomento, CloudWatch salva l'elenco e lo visualizza nel campo in modo da poterlo utilizzare per impostare allarmi futuri.

   > **Note**
   >
   > Se utilizzi Crea argomento per creare un nuovo argomento Amazon SNS, gli indirizzi e-mail devono essere verificati prima che i destinatari previsti ricevano le notifiche. Le e-mail vengono inviate da Amazon SNS solo quando l'allarme passa allo stato definito. Se lo stato cambia prima della verifica degli indirizzi e-mail, i destinatari previsti non riceveranno una notifica.
7. Visualizza un'anteprima dell'allarme nella sezione Alarm Preview (Anteprima allarme). Scegli Crea allarme.

Per impostare un allarme (AWS CLI)

- Apri AWS CLI e inserisci il seguente comando. Cambia il valore del parametro `alarm-actions` per fare riferimento a un argomento Amazon SNS creato in precedenza.

```
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
```

Questo esempio mostra come creare un allarme quando si verificano più di 10 errori utente in 5 minuti. Per ulteriori informazioni, consulta put-metric-alarm.

Per impostare un allarme (CloudWatch API)

- Chiama `PutMetricAlarm`. Per ulteriori informazioni, consulta Amazon CloudWatch API Reference.

CloudWatchmetriche per Rekognition

Questa sezione contiene informazioni sui CloudWatch parametri Amazon e sulla dimensione Operation disponibili per Amazon Rekognition.

Puoi anche visualizzare una vista aggregata dei parametri di Rekognition dalla console di Rekognition. Per ulteriori informazioni, consulta Esercizio 4: visualizzare i parametri aggregati (console).

CloudWatch metriche per Rekognition

La tabella seguente riepiloga i parametri di Rekognition.

Parametro	Descrizione
SuccessfulRequestCount	Il numero di richieste eseguite correttamente. L'intervallo di codici di risposta per una richiesta eseguita correttamente è compreso tra 200 a 299. Unità: numero Statistiche valide: `Sum,Average`

Parametro	Descrizione
ThrottledCount	Il numero di richieste sottoposte a throttling. Rekognition sottopone a throttling una richiesta quando riceve più richieste del limite di transazioni per secondo impostato per il tuo account. Se il limite impostato per il tuo account viene superato frequentemente, puoi richiedere un aumento del limite. Per richiedere un aumento, consulta Service Limits per AWS. Unità: numero Statistiche valide: `Sum,Average`
ResponseTime	Il periodo di tempo, in millisecondi, impiegato da Rekognition per calcolare la risposta. Unità: 1. Conteggio delle statistiche `Data Samples` 2. Millisecondi per le statistiche `Average` Statistiche valide: `Data Samples,Average` **Note** Il parametro `ResponseTime` non è incluso nel riquadro del parametro Rekognition.
DetectedFaceCount	Il numero di volti rilevati con l'operazione `IndexFaces` o `DetectFaces` . Unità: numero Statistiche valide: `Sum,Average`
DetectedLabelCount	Il numero di etichette rilevate con l'operazione `DetectLabels` . Unità: numero Statistiche valide: `Sum,Average`

Parametro	Descrizione
ServerErrorCount	Il numero di errori del server. L'intervallo di codici di risposta per un errore del server è compreso tra 500 a 599. Unità: numero Statistiche valide: `Sum, Average`
UserErrorCount	Il numero di errori utente (parametri non validi, immagine non valida, nessuna autorizzazione e così via). L'intervallo di codici di risposta per un errore utente è compreso tra 400 e 499. Unità: numero Statistiche valide: `Sum,Average`
MinInferenceUnits	Il numero minimo di unità di inferenza specificato durante la richiesta `StartProjectVersion` . Unità: numero Statistiche valide: `Average`
MaxInferenceUnits	Il numero massimo di unità di inferenza specificato durante la richiesta `StartProjectVersion` . Unità: numero Statistiche valide: `Average`
DesiredInferenceUnits	Il numero di unità di inferenza a cui Rekognition sta aumentando o diminuendo. Unità: numero Statistiche valide: `Average`

Parametro	Descrizione
InServiceInference Units	Il numero di unità di inferenza utilizzate dal modello. Unità: numero Statistiche valide: `Average` Si consiglia di utilizzare la statistica Average per ottenere la media di 1 minuto del numero di istanze utilizzate.

CloudWatch metriche per Rekognition Streaming

Rekognition ha anche un secondo spazio dei nomi utilizzato per le operazioni di streaming, “Rekognition Streaming”. La tabella seguente riepiloga i parametri di Rekognition Streaming.

Parametro	Descrizione
SuccessfulRequestCount	Il numero di richieste eseguite correttamente. L'intervallo di codici di risposta per una richiesta eseguita correttamente è compreso tra 200 a 299. Unità: numero Statistiche valide: `Sum,Average`
CallCount	Il numero di operazioni specificate eseguite nel tuo account. Statistiche valide: `Sum,Average`
ThrottledCount	Il numero di richieste sottoposte a throttling. Rekognition sottopone a throttling una richiesta quando riceve più richieste del limite di transazioni per secondo impostato per il tuo account. Se il limite impostato per il tuo account viene superato frequentemente, puoi richiedere un aumento del limite. Per richiedere un aumento, consulta Service Limits per AWS. Unità: numero Statistiche valide: `Sum,Average`

Parametro	Descrizione
ServerErrorCount	Il numero di errori del server. L'intervallo di codici di risposta per un errore del server è compreso tra 500 a 599. Unità: numero Statistiche valide: `Sum, Average`
UserErrorCount	Il numero di errori utente (parametri non validi, immagine non valida, nessuna autorizzazione e così via). L'intervallo di codici di risposta per un errore utente è compreso tra 400 e 499. Unità: numero Statistiche valide: `Sum,Average`
ConcurrentSessions	Il numero di sessioni simultanee attualmente utilizzate dal tuo account. Unità: numero Statistiche valide: `Max,Average`

CloudWatch dimensione per Rekognition

Per recuperare i parametri specifici delle operazioni, utilizza il namespace di `Rekognition` e fornisci una dimensione per l'operazione.

Per ulteriori informazioni sulle dimensioni, consulta Dimensions nella Amazon CloudWatch User Guide.

CloudWatch dimensione per le etichette personalizzate Rekognition

La tabella seguente mostra le CloudWatch dimensioni disponibili per l'uso con le etichette personalizzate Rekognition:

Dimensione	Descrizione
ProjectName	Il nome del progetto Rekognition Custom Labels che hai creato con `CreateProject` .

Dimensione	Descrizione
VersionName	Il nome della versione di progetto Rekognition Custom Labels che hai creato con `CreateProjectVersion` .

Per ulteriori informazioni sulle dimensioni, consulta Dimensions nella Amazon CloudWatch User Guide.

Registrazione delle chiamate API Amazon Rekognition con AWS CloudTrail

Amazon Rekognition è AWS CloudTrail integrato con un servizio che fornisce un registro delle azioni intraprese da un utente, ruolo o AWS servizio in Amazon Rekognition. CloudTrail acquisisce tutte le chiamate API per Amazon Rekognition come eventi. Le chiamate acquisite includono le chiamate dalla console Amazon Rekognition e le chiamate di codice alle operazioni API di Amazon Rekognition. Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per Amazon Rekognition. Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella console in Cronologia eventi. CloudTrail Utilizzando le informazioni raccolte da CloudTrail, puoi determinare la richiesta che è stata fatta ad Amazon Rekognition, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e ulteriori dettagli.

Per ulteriori informazioni CloudTrail, consulta la Guida per l'AWS CloudTrail utente.

Informazioni su Amazon Rekognition in CloudTrail

CloudTrail è abilitato sul tuo AWS account al momento della creazione dell'account. Quando si verifica un'attività in Amazon Rekognition, tale attività viene registrata CloudTrail in un evento AWS insieme ad altri eventi di servizio nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare eventi recenti nel tuo account. AWS Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi nel tuo AWS account, inclusi gli eventi per Amazon Rekognition, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le

regioni AWS. Il trail registra gli eventi di tutte le regioni della AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, consulta gli argomenti seguenti:

- Panoramica della creazione di un trail
- CloudTrail Servizi e integrazioni supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

Tutte le azioni di Amazon Rekognition vengono CloudTrail registrate e documentate nel riferimento all'API Amazon Rekognition. Ad esempio, le chiamate a e le azioni generano voci nei `CreateCollection` file di `CreateStreamProcessor` registro`DetectCustomLabels`. CloudTrail

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con credenziali utente root o AWS Identity and Access Management (IAM).
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro AWS servizio.

Per ulteriori informazioni, consulta Elemento CloudTrail userIdentity.

Informazioni sulle voci dei file di log Amazon Rekognition

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro. Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'azione richiesta, la data e l'ora dell'azione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra una voce di CloudTrail registro con azioni per la seguente API: `StartLabelDetection` and`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"
        }
    ]
}
```

Linee guida e quote in Amazon Rekognition

In queste sezioni vengono illustrate le linee guida e le quote relative all'utilizzo di Amazon Rekognition. Sono disponibili due tipi di quote. Le quote impostate ad esempio la dimensione massima dell'immagine non possono essere modificate. Le quote predefinite elencate nella pagina AWS Service Quotas possono essere modificate seguendo la procedura descritta nella sezione Quote di default.

Argomenti

- Regioni supportate
- Quote impostate
- Quote di default

Regioni supportate

Per un elenco delle AWS regioni in cui è disponibile Amazon Rekognition, consulta Regioni ed endpoints AWS nell'Amazon Web Services General Reference.

Quote impostate

Di seguito è riportato un elenco di limiti in Amazon Rekognition che non possono essere modificati. Per informazioni sui limiti che puoi cambiare come le Transazioni per secondo (TPS), consulta Quote di default.

Per i limiti Etichette personalizzate Amazon Rekognition, consulta le linee guida e le quote in Etichette personalizzate Amazon Rekognition.

Immagini Amazon Rekognition

- La dimensione massima di un'immagine archiviata come oggetto di Amazon S3 è limitata a 15 MB.
- La dimensione massima dell'immagine è di 10.000 pixel `DetectModerationLabels` sia in larghezza che in altezza.
- La dimensione massima dell'immagine è di 10.000 pixel `DetectLabels` sia in larghezza che in altezza.

- Affinché venga rilevato, un volto non deve occupare meno di 40 x 40 pixel in un'immagine di dimensioni pari a 1920 x 1080 pixel. Nelle immagini di dimensioni maggiori di 1920 x 1080 pixel la dimensione minima del volto deve essere aumentata in proporzione.
- La dimensione minima dell'immagine per l'altezza e la larghezza è pari a 80 pixel. La dimensione minima dell'immagine per `DetectProtectiveEquipment` è pari a 64 pixel sia in altezza che in larghezza.
- La dimensione massima dell'immagine è di 4096 pixel `DetectProtectiveEquipment` sia in larghezza che in altezza.
- Affinché venga rilevato da `DetectProtectiveEquipment`, un volto non deve occupare meno di 100x100 pixel in un'immagine di dimensioni pari a 800x1300. Nelle immagini di dimensioni maggiori di 800x1300 pixel la dimensione minima della persona deve essere aumentata in proporzione.
- La dimensione massima delle immagini come byte non elaborati trasferiti come parametri a un'API è pari a 5 MB. Il limite è di 4 MB per l'API `DetectProtectiveEquipment`.
- Amazon Rekognition supporta i formati di immagine PNG e JPEG. In altre parole, le immagini fornite come input alle varie operazioni dell'API, ad esempio `DetectLabels` e `IndexFaces`, devono avere uno dei formati supportati.
- Il numero massimo di vettori di volti che è possibile archiviare in una singola raccolta è pari a 20 milioni.
- Il numero massimo predefinito di vettori di volti che è possibile archiviare in una singola raccolta di volti è pari a 10 milioni.
- Il numero massimo di vettori di volti corrispondenti restituiti da search API è pari a 4096.
- Il numero massimo di vettori di utenti corrispondenti restituiti da search API è pari a 4096.
- `DetectText` è in grado di rilevare fino a 100 parole in un'immagine.
- `DetectProtectiveEquipment` è in grado di rilevare dispositivi di protezione individuale su un massimo di 15 persone.

Per informazioni sulle best practice relative alle immagini e al confronto dei volti, consulta Best practice per sensori, immagini di input e video.

Analisi di massa delle immagini di Amazon Rekognition

- Amazon Rekognition Image Bulk Analysis può analizzare batch di immagini di dimensioni fino a 10.000 immagini.

- Amazon Rekognition Image Bulk Analysis supporta manifesti di input di dimensioni fino a 50 MB.

Video archiviato di Video Amazon Rekognition

- Video Amazon Rekognition può analizzare video archiviati di dimensioni fino a 10 GB.
- Video Amazon Rekognition può analizzare video archiviati di durata fino a 6 ore.
- Video Amazon Rekognition supporta un massimo di 20 processi contemporanei per ogni account.
- I video memorizzati devono essere codificati utilizzando il codec H.264. I formati di file supportati sono MPEG-4 e MOV.
- Qualsiasi API Video Amazon Rekognition che analizza i dati audio supporta solo codec audio AAC.
- Il periodo TTL (Time To Live) per i token di paginazione è di 24 ore. I token di paginazione sono nel campo `NextToken` restituito dalle operazioni Get come `GetLabeldetection`.

Video Amazon Rekognition video in streaming

- Un flusso di input del video Kinesis può essere associato con al massimo 1 processore di flusso Video Amazon Rekognition.
- Un flusso di output di dati Kinesis può essere associato con al massimo 1 processore di flusso Video Amazon Rekognition.
- Il flusso di input del video Kinesis e un flusso di output dei dati Kinesis associati a un processore di flusso Video Amazon Rekognition non possono essere condivisi da più processori.
- Qualsiasi API Video Amazon Rekognition che analizza i dati audio supporta solo codec audio ACC.

Quote di default

Un elenco di quote predefinite è disponibile in AWS Service Quotas. Questi limiti sono di default e possono essere modificati. Per richiedere un aumento del limite, crea una richiesta. Per visualizzare i limiti di quota attuali (valori di quota applicati), consulta Amazon Rekognition Service Quotas. Per visualizzare la cronologia di utilizzo del TPS per Amazon Rekognition Image APIs, consulta la pagina Amazon Rekognition Service Quotas e scegli un'operazione API specifica per visualizzare la cronologia di tale operazione.

Argomenti

- Calcolo della modifica della quota TPS

- Best practice per le quote TPS
- Crea un caso per modificare le quote TPS

Calcolo della modifica della quota TPS

Qual è il nuovo limite che stai richiedendo? Le transazioni al secondo (TPS) sono più rilevanti al culmine del carico di lavoro previsto. È importante comprendere il numero massimo di chiamate API simultanee al picco di un carico di lavoro e il tempo di risposta (5-15 secondi). Tieni presente che 5 secondi dovrebbero essere il minimo. Di seguito sono riportati due esempi:

- Esempio 1: il numero massimo di utenti simultanei di Face Authentication (CompareFaces API) che mi aspetto all'inizio del mio orario di punta è 1000. Queste risposte verranno distribuite su un periodo di 10 secondi. Pertanto, il TPS richiesto è 100 (1000/10) per l'API nella CompareFaces mia regione pertinente.
- Esempio 2: il numero massimo di chiamate simultanee di Object Detection (DetectLabels API) previste all'inizio dell'ora di punta è 250. Queste risposte verranno distribuite su un periodo di 5 secondi. Pertanto, il TPS richiesto è 50 (250/5) per l'API nella DetectLabels mia regione pertinente.

Best practice per le quote TPS

Le best practice consigliate per Transazioni al secondo (TPS) includono l'attenuazione del traffico intenso, la configurazione di nuovi tentativi e la configurazione di backoff esponenziale e jitter.

1. Traffico regolare e con picchi. Il traffico intenso influisce sulla velocità di throughput. Per ottenere il massimo throughput per le transazioni assegnate al secondo (TPS), utilizza un'architettura serverless di accodamento o un altro meccanismo per «fluidificare» il traffico in modo che sia più coerente. Per esempi di codice e riferimenti per l'elaborazione di immagini e video su larga scala serverless con Rekognition, consulta Elaborazione di immagini e video su larga scala con Amazon Rekognition.
2. Configura nuovi tentativi. Segui le linee guida riportate in the section called “Gestione degli errori” per configurare i nuovi tentativi in base agli errori che li consentono.
3. Configurazione backoff esponenziale e jitter. La configurazione del backoff esponenziale e del jitter durante la configurazione dei nuovi tentativi consente di migliorare il throughput ottenibile. Vedi Ritentativi di errore e backoff esponenziale in. AWS

Crea un caso per modificare le quote TPS

Per creare un caso, vai a Crea caso e rispondi alle seguenti domande:

- Hai implementato the section called “Best practice per le quote TPS” per ridurre i picchi di traffico e configurare nuovi tentativi, backoff esponenziale e jitter?
- Hai calcolato la modifica della quota TPS di cui hai bisogno? In caso contrario, vedithe section called “Calcolo della modifica della quota TPS”.
- Hai controllato la cronologia di utilizzo del TPS per prevedere con maggiore precisione le tue esigenze future? Per visualizzare la cronologia di utilizzo del TPS, consulta la pagina Amazon Rekognition Service Quotas.
- Qual è il tuo caso d'uso?
- APIs Cosa intendi usare?
- In quali regioni intendi utilizzarli APIs ?
- Riesci a distribuire il carico su più regioni?
- Quante immagini elabori ogni giorno?
- Per quanto tempo pensi di mantenere questo volume (si tratta di un picco occasionale o continuo)?
- In che modo sei bloccato entro il limite predefinito? Consulta la seguente tabella delle eccezioni per confermare lo scenario che stai riscontrando.

Codice di errore	Eccezione	Messaggio	Che cosa significa?	Si può riprovare ?
Codice di stato HTTP 400	ProvisionedThroughputExceededException	Velocità assegnata superata.	Indica una limitazione (della larghezza di banda della rete). Puoi riprovare o valutare una richiesta di aumento del limite.	Sì

Codice di errore	Eccezione	Messaggio	Che cosa significa?	Si può riprovare ?
Codice di stato HTTP 400	Throttlin gException	Rallenta; per improvviso aumento del numero di richieste.	È possibile che tu stia inviando traffico intenso e che tu stia riscontrando una limitazione (della larghezza di banda della rete). Dovresti modellare il traffico e renderlo più fluido e coerente. Quindi configura ulteriori tentativi. Consulta le best practice.	Sì
Codice di stato HTTP 5xx	Throttlin gException (HTTP 500)	Servizio non disponibile	Indica che il backend sta aumentando per supportar e l'azione. È necessario ritentare la richiesta.	Sì

Per una comprensione dettagliata dei codici di errore, vedere the section called “Gestione degli errori”.

> **Note**
>
> Questi limiti dipendono dalla regione in cui ti trovi. La creazione di un caso per modificare un limite interessa l'operazione API richiesta nella regione in cui si richiede. Le altre operazioni API e le regioni non sono interessate.

Cronologia dei documenti per Amazon Rekognition

La tabella che segue descrive le modifiche importanti apportate a ogni versione della Guida per sviluppatori di Amazon Rekognition. Per ricevere notifiche sugli aggiornamenti di questa documentazione, è possibile sottoscrivere un feed RSS.

- Ultimo aggiornamento della documentazione: 15 giugno 2023

Modifica	Descrizione	Data
Amazon Rekognition ora supporta nuove etichette di moderazione e una maggiore precisione per la moderazione dei contenuti delle immagini	La funzione di moderazione dei contenuti di Amazon Rekognition è stata migliorata per una maggiore precisione, il rilevamento di nuove etichette e la capacità di identificare contenuti animati e/o illustrati.	1 febbraio 2024
Amazon Rekognition ora supporta l'analisi delle immagini in blocco	Amazon Rekognition ora supporta l'elaborazione di un'ampia raccolta di immagini in modo asincrono utilizzando un file manifest con l'operazione. StartMediaAnalysisJob	23 ottobre 2023
Amazon Rekognition ora supporta la moderazione dei contenuti personalizzata con adattatori	Amazon Rekognition ora supporta una maggiore precisione dell'API utilizzando adattatori che estendono le funzionalità DetectModerationLabels dei modelli di deep learning Rekognition esistenti.	12 ottobre 2023
Rekognition ora supporta i vettori utente con le raccolte	Le raccolte di volti Rekognition ora supportano la creazione di vettori utente. I vettori utente	12 giugno 2023

	aggregano più vettori di volti dello stesso utente, migliorando la precisione con rappresentazioni più affidabili di un utente.	
Le azioni per coinvolgere la gestione degli utenti sono state aggiunte alle seguenti politiche gestite: AmazonRekognitionReadOnlyAccess	Amazon Rekognition ha aggiunto le seguenti operazioni alle policy gestite `AmazonRekognitionReadOnlyAccess` : `ListUsers` , `SearchUsers` , `SearchUsersByImage`	12 giugno 2023
Immagini Amazon Rekognition ora può dedurre la direzione dello sguardo	Sono stati apportati miglioramenti alle operazioni di riconoscimento del volto di Immagini Amazon Rekognition, che ora possono dedurre la direzione dello sguardo di un volto rilevato.	31 maggio 2023

API di moderazione dei contenuti Rekognition migliorata	Rekognition ha migliorato il modello di moderazione dei contenuti per la moderazione di immagini e video. Il miglioramento amplia notevolmente il rilevamento di contenuti espliciti, violenti e allusivi. I clienti possono ora rilevare contenuti espliciti e violenti con maggiore precision e per migliorare l'esperienza dell'utente finale, protegger e l'identità del marchio e garantire che tutti i contenuti siano conformi alle normative e alle policy di settore.	9 maggio 2023
Immagini Amazon Rekognition è ora in grado di rilevare volti occlusi	Immagini Amazon Rekogniti on è ora in grado di rilevare l'occlusione dei volti. Un nuovo FaceOccluded attributo viene restituito da Amazon Rekogniti on DetectFaces Image and, che indica se il volto in un'immagine IndexFaces APIs è parzialmente catturato o non è completamente visibile a causa della sovrapposizione di oggetti, vestiti e parti del corpo.	5 maggio 2023

Rekognition è ora in grado di effettuare il riconoscimento facciale	Video Amazon Rekognition può ora essere usato per effettuare il riconoscimento in un video, verificando che un utente davanti a una telecamera sia fisicamente presente. Il rilevatore Face Liveness rileva anche gli attacchi di tipo spoof rivolti a una telecamera o il tentativo di bypassare una telecamera.	11 aprile 2023
Aggiornamento ad Video Amazon Rekognition.	Video Amazon Rekognition è ora in grado di rilevare più etichette e restituire maggiori informazioni sugli attributi di immagini ed etichette. L' GetLabelDetection API ora restituisce informazioni su alias e categorie. Le informazioni sulle etichette restituite e possono essere filtrate con opzioni di filtro inclusive ed esclusive. I risultati possono essere aggregati per timestamp o segmenti video.	7 dicembre 2022

Aggiornamento ad Immagini Amazon Rekognition.	Immagini Amazon Rekognition è ora in grado di rilevare più etichette e restituire maggiori informazioni sugli attributi di immagini ed etichette . L' DetectLabels API ora restituisce informazioni su alias, categorie e proprietà delle immagini, come i colori dominanti. Le informazioni sulle etichette restituite possono essere filtrate con opzioni di filtro inclusive ed esclusive.	11 novembre 2022
Actions for ProjectPolicy e Custom Labels Model Copy sono stati aggiunti alle seguenti politiche gestite: AmazonRekognitionReadOnlyAccess	Amazon Rekognition ha aggiunto le seguenti operazioni alla policy gestita `AmazonRekognitionReadOnlyAccess` : `ListProjectPolicies`	21 luglio 2022
Actions for ProjectPolicy e Custom Labels Model Copy sono state aggiunte alle seguenti politiche gestite: AmazonRekognitionFullAccess, AmazonRekognitionCustomLabelsFullAccess	Rekognition ha aggiunto le seguenti operazioni alle policy gestite `AmazonRekognitionCustomLabelsFullAccess` e `AmazonRekognitionFullAccess` : `CopyProjectVersion` , `PutProjectPolicy` , `ListProjectPolicies` , `DeleteProjectPolicy`	21 luglio 2022

Video Amazon Rekognition è ora in grado di rilevare le etichette nei video in streaming	Video Amazon Rekognition è in grado di rilevare etichette come animali domestici e pacchetti nei video in streaming. Questa operazione viene eseguita utilizzando le impostazioni `ConnectedHome` degli elaboratori di flussi creati con l'operazione `CreateStreamProcessor` .	28 aprile 2022
La documentazione di riferimento delle API è stato rimosso dalla guida per gli sviluppatori di Amazon Rekognition	La documentazione di riferimento delle API Amazon Rekognition è ora disponibile in Documentazione di riferimento delle API Amazon Rekognition.	24 febbraio 2022
Aggiornamento della gestione dei set di dati per le seguenti policy gestite: Policy gestita da AWS: AmazonRekognitionReadOnlyAccess, Policy gestita da AWS: AmazonRekognitionFullAccess, Policy gestita da AWS: AmazonRekognitionCustomLabelsFullAccess	Amazon Rekognition ha aggiunto le seguenti azioni `AmazonRekognitionReadOnlyAccess` alle `AmazonRekognitionFullOnlyAccess` `CreateDataset` politiche e gestite`ListDatasetEntries` : `AmazonRekognitionCustomLabelsFullAccess` `ListDatasetLabels` ,,,,, `DescribeDataset` `UpdateDatasetEntries` `DistributeDatasetEntries` DeleteDataset	1° novembre 2021

Un nuovo nodo nel sommario mostra gli esempi di Amazon Rekognition ospitati su GitHub	Gli esempi di codice aggiornati dal AWS Code Examples Repository ora vengono visualizzati in un nodo separato nella guida per sviluppatori di Amazon Rekognition per un accesso più semplice.	22 ottobre 2021
Amazon Rekognition è in grado di rilevare i fotogrammi neri e il contenuto principale del programma nei segmenti video	Amazon Rekognition è in grado di identificare fotogrammi neri, barre colore, titoli di testa, titoli di coda, loghi degli studi e il contenuto principale del programma come segnali d'azione tecnici in un video utilizzando le operazioni `StartSegmentDetection` e `GetSegmentDetection` .	7 giugno 2021
Aggiornamento della gestione dei set di dati per le seguenti policy gestite:	Puoi utilizzare l'operazione `DetectText` di Amazon Rekognition per rilevare fino a 100 parole in un'immagine.	21 maggio 2021
Aggiornamento dei tag per e AmazonRekognitionReadOnlyAccessAmazonRekognitionFullAccess	Rekognition ha aggiunto nuove azioni di tagging alle policy `AmazonRekognitionFullAccess` e `AmazonRekognitionReadOnlyAccess` .	2 aprile 2021

Amazon Rekognition ora supporta il tagging	Ora puoi utilizzare i tag per identificare, organizzare, cercare e filtrare le raccolte, gli elaboratori di flussi e i modelli Custom Labels di Amazon Rekognition.	25 marzo 2021
Amazon Rekognition è ora in grado di rilevare i dispositivi di protezione individuale	Amazon Rekognition è ora in grado di rilevare protezioni per mani, volto e testa sulle persone in un'immagine.	15 ottobre 2020
Amazon Rekognition ha nuove categorie di moderazione dei contenuti	Le categorie di moderazione dei contenuti di Amazon Rekognition ora includono 6 nuove categorie: droghe, tabacco, alcol, gioco d'azzardo, gesti maleducati e simboli di odio.	12 ottobre 2020
Nuovo tutorial per la visualizzazione locale dei risultati di Video Amazon Rekognition da Kinesis Video Streams	Puoi visualizzare l'output di Video Amazon Rekognition da un video in streaming in Kinesis Video Streams in un feed video locale.	20 luglio 2020
Nuovo tutorial di Amazon Rekognition per l'utilizzo di Gstreamer	Utilizzando Gstreamer, puoi importare un video in live streaming dall'origine della fotocamera di un dispositivo ad Video Amazon Rekognition tramite Kinesis Video Streams.	17 luglio 2020

Amazon Rekognition ora supporta la segmentazione dei video archiviati	Con l'API asincrona di segmentazione Video Amazon Rekognition è possibile rilevare fotogrammi neri, barre colore, titoli di coda e riprese nei video archiviati.	22 giugno 2020
Amazon Rekognition ora supporta le policy dell'endpoint VPC Amazon	Specificando una policy puoi limitare l'accesso a un endpoint VPC Amazon di Amazon Rekognition.	3 marzo 2020
Amazon Rekognition ora supporta il rilevamento del testo nei video archiviati	Puoi utilizzare l'API Video Amazon Rekognition per rilevare in modo asincrono il testo in un video archiviato.	17 febbraio 2020
Amazon Rekognition ora supporta l'IA aumentata (anteprima) ed Etichette personalizzate Amazon Rekognition	Con Amazon Rekognition Custom Labels puoi rilevare oggetti, scene e concetti specializzati nelle immagini creando il tuo modello di apprendimento automatico. DetectModerationLabels ora supporta Amazon Augmented AI (Preview).	3 dicembre 2019
Amazon Rekognition ora supporta AWS PrivateLink	Con AWS PrivateLink puoi stabilire una connessio ne privata tra il tuo VPC e Amazon Rekognition.	12 settembre 2019

Filtraggio dei volti di Amazon Rekognition	Amazon Rekognition aggiunge un supporto avanzato per il filtraggio dei volti IndexFaces al funzionamento dell'API e introduce il filtraggio facciale per le operazioni delle API. CompareFaces SearchFacesByImage	12 settembre 2019
Aggiornati gli esempi di Video Amazon Rekognition	Codice di esempio di Video Amazon Rekognition aggiornati per creare e configurare l'argomento Amazon SNS e la coda Amazon SQS.	5 settembre 2019
Aggiunti esempi Ruby e Node.js	Aggiunti esempi Ruby e Node.js di Immagini Amazon Rekognition per il rilevamento sincrono di etichette e volti.	19 agosto 2019
Rilevamento di contenuti non sicuri aggiornato	Il rilevamento di contenuti non sicuri di Amazon Rekogniti on ora è in grado di rilevare contenuti violenti.	9 agosto 2019
GetContentModeration operazione aggiornata	GetContentModeration ora restituisce la versione del modello di rilevamento della moderazione utilizzato per rilevare contenuti non sicuri.	13 febbraio 2019

GetLabelDetection e DetectModerationLabels operazioni aggiornate	GetLabelDetection ora restituisce le informazioni sui riquadri di delimitazione per gli oggetti comuni e una tassonomia gerarchica delle etichette rilevate. Viene ora restituita la versione del modello utilizzata per il rilevamento delle etichette. DetectModerationLabels ora restituisce la versione del modello utilizzata per rilevare contenuti non sicuri.	17 gennaio 2019
DetectFaces e IndexFaces operazione aggiornata	Questa versione aggiorna l' IndexFaces operazione DetectFaces and. Quando il parametro di input Attributes è impostato su ALL, i punti di riferimento per la posizione dei volti includono 5 nuovi punti di riferimento: upperJawlineLeft, midJawlineLeft, ChinBottom,, midJawlineRight. upperJawlineRight	19 novembre 2018
DetectLabels operazione aggiornata	Ora, per determinati oggetti vengono restituiti riquadri di delimitazione. Per le etichette è ora disponibile una tassonomia gerarchica. Puoi ottenere la versione del modello utilizzato per il rilevamento.	1° novembre 2018

IndexFaces operazione aggiornata	Con IndexFaces è ora possibile utilizzare il parametro QualityFilter di input per filtrare i volti rilevati con bassa qualità. È inoltre possibile utilizzare il parametro di MaxFaces input per ridurre il numero di volti restituiti in base alla qualità del rilevamento dei volti e alle dimensioni del volto rilevato.	18 settembre 2018
DescribeCollection operazione aggiunta	È ora possibile ottenere informazioni su una raccolta esistente chiamando l' DescribeCollection operazion e.	22 agosto 2018
Nuovi esempi di Python	Sono stati aggiunti esempi di Python ai contenuti di Video Amazon Rekognition oltre a un certo riordino dei contenuti.	26 giugno 2018
Layout dei contenuti aggiornat o	I contenuti di Immagini Amazon Rekognition sono stati riorganizzati insieme a nuovi esempi di Python e C#.	29 maggio 2018

Amazon Rekognition supporta AWS CloudTrail	Amazon Rekognition è AWS CloudTrail integrato con un servizio che fornisce un registro delle azioni intrapres e da un utente, ruolo o AWS servizio in Amazon Rekogniti on. Per ulteriori informazi oni, consulta Registrazione delle chiamate API Amazon Rekognition con AWS. CloudTrail	6 aprile 2018
Analisi di video archiviati e in streaming. Nuovo sommario	Per informazioni sull'analisi di video archiviati, consulta Utilizzo dei video archiviati. Per informazioni sull'analisi di video in streaming, consulta Utilizzo dei video in streaming . Il sommario della documenta zione di Amazon Rekogniti on è stato riorganizzato per accogliere le operazioni relative a immagini e video.	29 novembre 2017
Modelli di rilevamento di volti e testo nelle immagini	Amazon Rekognition è ora in grado di rilevare il testo nelle immagini. Per ulteriori informazioni, consulta Rilevamento del testo. Amazon Rekognition introduce il controllo delle versioni per il modello di deep learning per eseguire il rilevamento del volto. Per ulteriori informazioni, consulta Versioni multiple del modello.	21 Novembre 2017

Riconoscimento di volti celebri	Amazon Rekognition ora è in grado di analizzare le immagini per il riconosci mento delle celebrità. Per ulteriori informazioni, consulta Riconoscimento di volti celebri.	8 giugno 2017
Moderazione di immagini	Amazon Rekognition è ora in grado di determinare se in un'immagine sono presenti contenuti per adulti espliciti o spinti. Per ulteriori informazi oni, consulta Rilevamento di contenuti sconsigliati.	19 aprile 2017
Intervallo di età per i volti rilevati. Riquadro dei parametri aggregati di Rekognition	Amazon Rekognition ora restituisce un intervallo di età stimato, in anni, per i volti rilevati dall'API Rekogniti on. Per ulteriori informazi oni, consulta AgeRange. La console Rekognition ora dispone di un riquadro delle metriche che mostra i grafici delle attività per un aggregato di metriche Amazon for Rekognition in un periodo di CloudWatch tempo specifica to. Per ulteriori informazioni, consulta Esercizio 4: visualizz are i parametri aggregati (console).	9 febbraio 2017

Nuovo servizio e guida	Questa è la versione iniziale del servizio di analisi dell'immagine, Amazon Rekognition, e della Guida per gli sviluppatori di Amazon Rekognition.	30 novembre 2016

AWS Glossario

Per la AWS terminologia più recente, consultate il AWS glossario nella sezione Reference.Glossario AWS

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.