aws

Guia do Desenvolvedor

Amazon Rekognition



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

Amazon Rekognition: Guia do Desenvolvedor

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

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não são propriedade da Amazon pertencem aos respectivos proprietários, os quais podem ou não ser afiliados, estar conectados ou ser patrocinados pela Amazon.

Table of Contents

O que é o Amazon Rekognition? 1
Capacidades gerais 1
Casos de uso 2
Benefícios 4
Amazon Rekognition e elegibilidade para HIPAA 4
Você é usuário do Amazon Rekognition pela primeira vez? 5
Como o Rekognition funciona 6
Entender os tipos de análise do Rekognition 7
Rótulos 9
Etiquetas personalizadas 10
Detecção de vivacidade facial 11
Detecção e análise facial 11
Pesquisa facial 11
Tráfego de pessoas 12
Equipamento de proteção individual 12
Celebridades 12
Detecção de texto 12
Conteúdo impróprio ou ofensivo 13
Personalização 13
Análise em massa 14
Entender as operações de imagem e vídeo do Rekognition 14
Operações do Amazon Rekognition Image 14
Operações de vídeo do Amazon Rekognition Video 14
Operações não baseadas em armazenamento e baseadas em armazenamento 15
Usando o AWS SDK ou HTTP para chamar as operações da API Amazon Rekognition 16
Operações de API sem armazenamento e armazenamento 16
Operações que não são de armazenamento 16
Operações de API baseadas em armazenamento 19
Noções básicas do versionamento de modelo 20
Começar 22
Etapa 1: Configure uma conta da AWS e crie um usuário 22
Crie uma AWS conta e um usuário 23
Etapa 2: configurar o AWS CLI e AWS SDKs 25
Conceder acesso programático 27

Trabalhando com AWS SDKs ... 31
Etapa 3: Começar a usar a API AWS CLI e AWS SDK ... 32
Formatando os exemplos AWS CLI ... 33
Próxima etapa ... 33
Etapa 4: Começando a usar o console ... 33
Configurar permissões do console ... 34
Exercício 1: Detecte objetos e cenas (console) ... 38
Exercício 2: Analisar faces (console) ... 46
Exercício 3: Compare faces (console) ... 48
Exercício 4: Ver métricas agregadas (console) ... 51
Trabalhando com imagens e vídeos ... 53
Como trabalhar com imagens ... 53
Especificações de imagem ... 54
Analisando imagens em um bucket do Amazon S3 ... 56
Usar um sistema de arquivos local ... 72
Exibir caixas delimitadoras ... 88
Obter a orientação e as coordenadas da caixa delimitadora da imagem ... 101
Trabalhar com operações de análise de vídeo armazenado ... 112
Tipos de análise ... 113
Visão geral da API Amazon Rekognition Video ... 113
Chamando as operações de vídeo do Amazon Rekognition Video ... 116
Configuração do Amazon Rekognition Video ... 123
Analisando um vídeo armazenado (SDK) ... 127
Analisar um vídeo (AWS CLI) ... 156
Referência: Notificação de resultados de análise de vídeo ... 160
Vídeo sobre solução de problemas do Amazon Rekognition ... 161
Trabalhando com eventos de streaming de vídeo ... 164
Visão geral das operações do processador de stream do Amazon Rekognition Video ... 164
Adicionar tags a um novo processador de stream ... 165
Adicionar tags a um processador de stream existente ... 166
Listar tags em um processador de stream ... 167
Excluir tags de um processador de stream ... 167
Gerenciamento de erros ... 168
Componentes de erros ... 168
Mensagens e códigos de erro ... 169
Tratamento de erros na aplicação ... 174

Usar o Amazon Rekognition com o FedRAMP 175
Melhores práticas para sensores, imagens de entrada e vídeos 179
Latência de operação do Amazon Rekognition Image 180
Recomendações para imagens de entrada de comparação facial 180
Recomendações gerais para entrada de imagens para operações faciais 180
Recomendações para pesquisar faces em uma coleção 181
Recomendações para a configuração da câmera (imagem e vídeo) 182
Recomendações para a configuração da câmera (vídeo armazenado e streaming) 183
Recomendações para a configuração da câmera (streaming de vídeo) 184
Recomendações para o uso do Face Liveness 185
Detectando objetos e conceitos 186
Objetos de resposta de rótulos 187
Caixas delimitadoras 187
Escore de confiança 188
Pais 188
Categorias 189
Aliases 189
Propriedades da imagem 190
Versão do modelo 191
Filtros de inclusão e exclusão 191
Classificação e agregação de resultados 192
Detectar rótulos em uma imagem 192
DetectLabels solicitação de operação 203
DetectLabels resposta 204
Transformando a resposta DetectLabels 208
Detectando rótulos em um vídeo 212
StartLabelDetectionSolicitação 213
GetLabelDetection Resposta da operação 214
Transformando a resposta GetLabelDetection 221
Detectar rótulos em eventos de streaming de vídeo 229
Configurando seus recursos do Amazon Rekognition Video e do Amazon Kinesis 230
Chamar operações de detecção de rótulo para eventos de streaming de vídeo 235
Detectar rótulos personalizados 242
Detectando e analisanda faces 243
Visão geral da detecção facial e comparação de faces 244
Diretrizes sobre atributos faciais 246

Detectanda faces em uma imagem ... 247
DetectFaces solicitação de operação ... 259
DetectFaces resposta da operação ... 259
Comparanda faces em imagens ... 266
CompareFaces solicitação de operação ... 279
CompareFaces resposta da operação ... 279
Detectanda faces em um vídeo armazenado ... 282
GetFaceDetection resposta da operação ... 292
Pesquisa de faces em uma coleção ... 297
Gerenciar coleções de faces, faces e usuários ... 300
Gerenciar uma coleção ... 300
Gerenciar faces em uma coleção ... 301
Gerenciar usuários em uma coleção ... 301
Usar limites de similaridade para associar e combinar rostos ... 302
Usando limites de similaridade para associar faces ... 302
Usar limites de similaridade para combinar faces ... 302
Orientação para indexar faces em cenários comuns ... 303
Aplicações críticas ou de segurança pública ... 303
Aplicativos de compartilhamento de fotos e mídias sociais ... 303
Uso geral ... 304
Pesquisa de faces e usuários em uma coleção ... 304
Casos de uso que envolvem segurança pública ... 305
Usando o Amazon Rekognition para ajudar na segurança pública ... 306
Criar uma coleção ... 307
CreateCollection solicitação de operação ... 314
CreateCollection resposta da operação ... 314
Aplicação de tags nas coleções ... 314
Adicionar tags a uma nova coleção ... 315
Adicionar tags a uma coleção existente ... 316
Listar tags em uma coleção ... 317
Excluir tags de uma coleção ... 318
Listar coleções ... 319
ListCollections solicitação de operação ... 325
ListCollections resposta da operação ... 326
Descrever uma coleção ... 326
DescribeCollection solicitação de operação ... 333

DescribeCollectionresposta da operação 334
Excluir uma coleção 334
DeleteCollection solicitação de operação 341
DeleteCollection resposta da operação 341
Adicionar faces a uma coleção 341
Filtrar faces 342
IndexFaces solicitação de operação 352
IndexFaces resposta da operação 352
Listanda faces e usuários associados em uma coleção 361
ListFaces solicitação de operação 367
ListFaces resposta da operação 367
Excluir faces de uma coleção 369
DeleteFaces solicitação de operação 374
DeleteFaces resposta da operação 374
Criar um usuário 375
Excluir um usuário 379
Associando faces a um usuário 381
AssociateFaces resposta da operação 384
Desassocianda faces de um usuário 386
DisassociateFaces resposta da operação 389
Listando usuários em uma coleção 390
ListUsers resposta da operação 392
Procurando por uma face (ID facial) 393
SearchFaces solicitação de operação 399
SearchFaces resposta da operação 400
Procurando uma face (imagem) 401
SearchFacesByImage solicitação de operação 409
SearchFacesByImage resposta da operação 409
Pesquisando usuários (ID facial/ID de usuário) 410
SearchUsers solicitação de operação 414
SearchUsers resposta da operação 415
Pesquisar usuários (imagem) 416
SearchUsersByImage solicitação de operação 420
SearchUsersByImage resposta da operação 420
Pesquisanda faces em vídeos armazenados 422
GetFaceSearch resposta da operação 431

Pesquisando faces em uma coleção em um vídeo de streaming 436
Configurando seus recursos do Amazon Rekognition Video e do Amazon Kinesis 437
Pesquisanda faces em um streaming de vídeo 440
Streaming usando um GStreamer plug-in 465
Solução de problemas de streaming de vídeo 468
Pessoas trafegando 475
GetPersonTracking resposta da operação 485
Detecção de equipamento de proteção individual 489
Tipos de EPI 490
Proteção facial 490
Proteções de mão 490
Proteções de cabeça 490
Confiança na detecção de EP 491
Resumindo o EPI detectado em uma imagem 491
Tutorial: Criando uma AWS Lambda função que detecta imagens com PPE 492
Entendendo a API de detecção de PPE 492
Fornecer uma imagem 492
Entendendo a DetectProtectiveEquipment resposta 494
Detectando EPI em uma imagem 499
Exemplo: caixas delimitadoras e capas faciais 511
Reconhecendo celebridades 527
Reconhecimento de celebridades comparado à busca facial 528
Reconhecer celebridades em uma imagem 528
Chamando RecognizeCelebrities 529
RecognizeCelebrities solicitação de operação 539
RecognizeCelebrities resposta da operação 539
Reconhecendo celebridades em um vídeo armazenado 542
GetCelebrityRecognition resposta da operação 558
Obter informações sobre celebridades 560
Chamando GetCelebrityInfo 560
GetCelebrityInfo solicitação de operação 565
GetCelebrityInfo resposta da operação 566
Como moderar um conteúdo 567
Usando a moderação de imagens e vídeos APIs 569
Categorias de etiquetas 570
Tipo de conteúdo 582

Confiança 583
Versionamento 583
Classificação e agregação 583
Status de adaptador de moderação personalizado 584
Testar a versão 7 da moderação de conteúdo e transformar a resposta da API 584
AWS SDK e guia de uso para moderação de conteúdo, versão 7 585
Mapeamentos de rótulos para as versões 6.1 a 7 585
Detectando imagens inapropriadas 591
Detectando conteúdo impróprio em uma imagem 591
..... 591
DetectModerationLabels solicitação de operação 598
DetectModerationLabels resposta da operação 598
Detectando vídeos armazenados inapropriados 600
GetContentModeration resposta da operação 608
Aprimorando a precisão com moderação personalizada 611
Criação e uso de adaptadores 612
Preparando seus conjuntos de dados 615
Gerenciando adaptadores com a AWS CLI e SDKs 617
Tutorial de adaptador de moderação personalizado 624
Avaliar e melhorar o adaptador 642
Formatos de arquivo manifesto 644
Práticas recomendadas para adaptadores de treinamento 649
Configurando AutoUpdate permissões 650
AWS Notificação do Health Dashboard para Rekognition 653
Revisão de conteúdo inadequado com a Amazon A2I 654
Detectar texto 660
Detectar texto em uma imagem 662
DetectText solicitação de operação 670
DetectText resposta da operação 671
Detectar texto em um vídeo armazenado 677
Filtros 687
GetTextDetection resposta 687
Detectando segmentos de vídeo 693
Dicas técnicas 694
Molduras pretas 694
Créditos 694

Barras de cores ... 695
Barreiras ... 695
Logotipos de estúdio ... 695
Conteúdo ... 695
Detecção de captura ... 696
Sobre a API de detecção de segmentos do Amazon Rekognition Video ... 697
Usar a API Amazon Rekognition Segment ... 697
Iniciando a análise do segmento ... 698
Obtendo resultados de análise de segmentos ... 699
Exemplo: Detectando segmentos em um vídeo armazenado ... 704
Detectando a vivacidade da face ... 717
Requisitos de vivacidade facial do lado do usuário ... 719
Diagramas de arquitetura e sequência ... 720
Pré-requisitos ... 722
Etapa 1: Configurar uma conta da AWS ... 722
Etapa 2: configurar o Face Liveness AWS SDKs ... 722
Etapa 3: configurar os recursos do AWS Amplify ... 723
Melhores práticas para detectar a vivacidade facial ... 723
Programando o Amazon Rekognition Face Liveness APIs ... 723
Etapa 1: CreateFaceLivenessSession ... 724
Etapa 2: StartFaceLivenessSession ... 725
Etapa 3: GetFaceLivenessSessionResults ... 725
Etapa 4: Responda aos resultados ... 727
Chamando o rosto de vivacidade APIs ... 727
Configurando e personalizando seu aplicativo ... 733
Configurar seu aplicativo ... 733
Personalize seu aplicativo ... 734
Modelo de responsabilidade compartilhada Face Liveness ... 734
Diretrizes de atualização do Face Liveness ... 738
Versionamento e prazos ... 738
Lançamento da versão e matriz de compatibilidade ... 739
Comunicação de novos lançamentos ... 740
Perguntas frequentes sobre o Face Liveness ... 740
Análise em massa ... 744
Processamento de imagens em massa ... 744
Criar uma tarefa de análise em massa (CLI) ... 744

Análise em massa de manifestos de saída ... 746
Tipo de conteúdo ... 747
Verificar previsões e adaptadores de treinamento ... 747
Tutoriais ... 748
Armazenamento de dados do Amazon Rekognition com Amazon RDS e DynamoDB ... 748
Pré-requisitos ... 749
Obter rótulos para imagens em um bucket do Amazon S3 ... 749
Criação de uma tabela do Amazon DynamoDB ... 751
Carregar dados para o DynamoDB ... 752
Criação de um banco de dados MySQL no Amazon RDS ... 754
Upload de dados para uma tabela MySQL do Amazon RDS ... 755
Usando o Amazon Rekognition e o Lambda para marcar ativos em um bucket do Amazon S3 ... 757
Pré-requisitos ... 759
Configurar o perfil IAM do Lambda ... 759
Criar o projeto ... 760
Escreva o código ... 763
Empacote o projeto ... 773
Implante a função do Lambda ... 774
Teste o método Lambda ... 775
Criação AWS de aplicativos de análise de vídeo ... 776
Pré-requisitos ... 777
Procedimento ... 777
Criar uma função do Amazon Rekognition Lambda ... 778
Pré-requisitos ... 780
Criar o tópico do SNS ... 780
Criar a função do Lambda ... 780
Configurar a função do Lambda ... 781
Configurar o perfil IAM do Lambda ... 782
Crie o projeto AWS Toolkit for Eclipse Lambda ... 783
Testar a função do Lambda ... 787
Como usar o Amazon Rekognition para verificação de identidade ... 788
Pré-requisitos ... 789
Criar uma coleção ... 789
Registro do novo usuário ... 790
Login de usuário existente ... 799

Usar o Amazon Rekognition (REK) para detectar rótulos para aplicativos de marketing 801
Pré-requisitos 802
Criação e preenchimento de uma tabela do DynamoDB 803
Criar uma função do Lambda com registro em log e notificação. 805
Configurar permissões e a função do Lambda. 808
Fazer o upload de imagens e metadados com o Amazon S3. 809
CloudWatch Registros de pesquisa do cliente 811
Detectando rótulos em uma imagem usando Lambda e Python 812
Crie uma função do Lambda (console) 813
(Opcional) Crie uma camada (console) 814
Adicionar código Python (console) 815
Para adicionar código Python (console) 818
Exemplos de código 822
Conceitos básicos 826
Olá, Amazon Rekognition 827
Ações 830
Cenários 990
Criar uma coleção e encontrar faces nela 991
Criar uma aplicação com tecnologia sem servidor para gerenciar fotos 1003
Detectar EPI em imagens 1008
Detectar e exibir elementos em imagens 1008
Detectar faces em uma imagem 1024
Detectar informações em vídeos 1025
Detectar objetos em imagens 1065
Detectar pessoas e objetos em um vídeo 1068
Salvar o EXIF e outras informações de imagem 1070
Referência da API 1071
Segurança 1072
Gerenciamento de identidade e acesso 1072
Público 1073
Autenticar com identidades 1073
Gerenciar o acesso usando políticas 1076
Como o Amazon Rekognition funciona com o IAM 1079
Políticas gerenciadas pela AWS 1085
Práticas recomendadas de política 1094
Utilizar o console 1095

Exemplos de uso de políticas baseadas em identidade ... 1095
Exemplos de políticas baseadas em recursos ... 1098
Solução de problemas ... 1099
Proteção de dados ... 1101
Criptografia de dados ... 1102
Privacidade do tráfego entre redes ... 1105
Usando o Amazon Rekognition com endpoints da VPC da Amazon ... 1105
Criar endpoints do Amazon VPC para o Amazon Rekognition ... 1106
Criar uma política de endpoint VPC para o Amazon Rekognition ... 1107
Validação de conformidade ... 1108
Resiliência ... 1109
Análise de configuração e vulnerabilidade ... 1109
Prevenção do problema do substituto confuso entre serviços ... 1109
Segurança da infraestrutura ... 1112
Monitoramento ... 1113
Monitorando o Rekognition com a Amazon CloudWatch ... 1113
Usando CloudWatch métricas para Rekognition ... 1114
Métricas de reconhecimento de acesso ... 1115
Criar um alarme ... 1116
CloudWatchmétricas para Rekognition ... 1118
Registro de chamadas da API Amazon Rekognition com AWS CloudTrail ... 1123
Informações sobre o Amazon Rekognition em CloudTrail ... 1123
Entendendo as entradas do arquivo de log do Amazon Rekognition ... 1124
Diretrizes e cotas ... 1127
Regiões compatíveis ... 1127
Definir cotas ... 1127
Amazon Rekognition Image ... 1127
Análise em massa do Amazon Rekognition Image ... 1127
Vídeo armazenado no Amazon Rekognition Video ... 1129
Streaming de vídeo com o Amazon Rekognition Video ... 1129
Cotas padrão ... 1129
Calcular a alteração da cota de TPS ... 1130
Melhores práticas para cotas de TPS ... 1130
Crie um caso para alterar as cotas de TPS ... 1131
Histórico de documentos ... 1134
AWS Glossário ... 1149

..... mcl

O que é o Amazon Rekognition?

O Amazon Rekognition é um serviço de análise de imagens e vídeos baseado em nuvem que facilita a adição de recursos avançados de visão computacional aos seus aplicativos. O serviço é baseado em tecnologia comprovada de aprendizado profundo e não requer experiência em machine learning para ser usado. O Amazon Rekognition inclui uma API easy-to-use simples que pode analisar rapidamente qualquer arquivo de imagem ou vídeo armazenado no Amazon S3.

Você pode adicionar recursos que detectam objetos, texto, conteúdo não seguro, analisam imagens/vídeos e comparam rostos ao seu aplicativo usando o Rekognition. APIs Com o reconhecimento facial do Amazon Rekognition APIs, você pode detectar, analisar e comparar faces em uma ampla variedade de casos de uso, incluindo verificação de usuários, catalogação, contagem de pessoas e segurança pública.

O serviço é baseado na mesma tecnologia comprovada e altamente dimensionável de aprendizado profundo desenvolvida pelos cientistas de visão computacional da Amazon, uma tecnologia que consegue analisar diariamente bilhões de imagens e vídeos. O Rekognition aprende rotineiramente com novos dados, e frequentemente adicionamos novos rótulos e recursos ao serviço.

Para obter mais informações, consulte o Amazon Rekognition FAQs.

Capacidades gerais

Análise de imagem:

- Detecção de objetos, cenas e conceitos: detecte e classifique objetos, cenas, conceitos e celebridades em imagens.
- Detecção de texto: detecte e reconheça texto impresso e manuscrito em imagens em vários idiomas.
- Conteúdo não seguro: detecte e filtre conteúdo e imagens explícitos, inapropriados e violentos. Detecte rótulos granulares de conteúdo inseguro.
- Reconhecimento de celebridades: reconheça dezenas de milhares de celebridades em suas imagens em diferentes categorias, como políticos, atletas, atores e músicos.
- Análise facial: detecte, analise e compare rostos, além de atributos faciais, como sexo, idade e emoções. Os casos de uso podem incluir verificação de usuários, catalogação, contagem de pessoas e segurança pública.

- Rótulos personalizados: crie classificadores personalizados para detectar objetos específicos para seu caso de uso, como logotipos, produtos e caracteres.
- Propriedades da imagem: analise as propriedades da imagem, como qualidade, cor, nitidez e contraste.

Análise de vídeo:

- Detecção de objetos, cenas e conceitos: detecte e classifique objetos, cenas, conceitos e celebridades em vídeos.
- Detecção de texto: detecte e reconheça texto impresso e manuscrito em vídeos em vários idiomas.
- Trajetória de pessoas: acompanhe as pessoas identificadas enquanto elas se movem pelos quadros de vídeo.
- Análise facial: detecte, analise e compare rostos em streaming ou vídeos armazenados.
- Reconhecimento de celebridades: reconheça dezenas de milhares de celebridades em vídeos armazenados em diferentes categorias, como políticos, atletas, atores e músicos.
- Detecção de conteúdo inseguro: detecte conteúdo explícito, impróprio e violento em vídeos.
- Segmentação de vídeo: identifique automaticamente segmentos úteis de vídeo, como quadros pretos e créditos finais.
- Vivacidade facial: detecte se um usuário ao vivo está presente durante a verificação facial.

Casos de uso

Bibliotecas de mídia pesquisáveis: o Rekognition detecta rótulos, objetos, conceitos e cenas em imagens e vídeos. Você pode tornar esses rótulos pesquisáveis com base nessa análise visual de conteúdo. Útil para criar bibliotecas de imagens e vídeos pesquisáveis.

Verificação de identidade do usuário com base no rosto: confirme as identidades do usuário comparando rostos em imagens com imagens faciais de referência. Útil para verificação de identidade em aplicativos.

Detecção de vivacidade facial: o Rekognition Face Liveness é um atributo de machine learning (ML) totalmente gerenciado projetado para ajudar os desenvolvedores a impedir fraudes na verificação de identidade baseada na face. O recurso ajuda a verificar se um usuário está fisicamente presente na frente da câmera e se não é um malfeitor falsificando a face do usuário. O uso do Rekognition Face Liveness pode ajudar você a detectar ataques falsos apresentados a uma câmera, como

fotos impressas, fotos/vídeos digitais ou máscaras 3D. Também ajuda a detectar ataques falsos que contornam uma câmera, como vídeos pré-gravados ou deepfake injetados diretamente no subsistema de captura de vídeo.

Pesquisa facial: com o Rekognition, você pode pesquisar imagens, vídeos armazenados e streaming de vídeo em busca de faces que correspondam aos armazenados em um contêiner conhecido como coleção facial. Coleção de faces é um índice de faces que você detém e gerencia. Pesquisar pessoas com base em seus rostos exige que você Indexe e depois Pesquise os rostos.

Detecção de conteúdo inseguro: detecte e filtre conteúdo explícito, impróprio e violento em imagens e vídeos. Usa rótulos para filtragem granular com base nas necessidades comerciais. A API de moderação de conteúdo também retorna uma lista hierárquica de todos os rótulos detectados (objetos e conceitos), além de pontuações de confiança. Esses objetos/rótulos indicam categorias específicas de conteúdo não seguro, o que permite a filtragem granular e o gerenciamento de grandes volumes de conteúdo gerado pelo usuário (UGC). Você pode personalizar a saída da API de moderação de conteúdo com adaptadores, que aprimoram o desempenho de imagens como as que você fornece como dados de treinamento.

Detecção de equipamento de proteção individual: detecte equipamentos de proteção individual em imagens para monitorar a conformidade de segurança em vários setores. Você pode sinalizar automaticamente condições inseguras ao detectar equipamentos impróprios e receber alertas sobre essas condições, o que pode melhorar a conformidade e o treinamento.

Reconhecimento de celebridades: reconheça celebridades em suas imagens e vídeos em várias categorias, como políticos, atletas, atores e músicos. Você pode identificar aparições de celebridades sem precisar fornecer nomes.

Detecção de texto: detecte e extraia texto em imagens para pesquisa visual ou extração de metadados. Isso funciona em diferentes fontes e estilos. Detecta a orientação para lidar com texto em placas e banners.

Rótulos personalizados: identifique objetos, conceitos e cenas personalizados específicos para casos de uso comercial, como detecção de logotipo. Você pode treinar classificadores personalizados para lidar com objetos de nicho ou proprietários, o que melhora a precisão em objetos-chave versus classificadores gerais. Para obter mais informações, consulte O que são Amazon Rekognition Custom Labels? no Guia do desenvolvedor do Amazon Rekognition Custom Labels.

Benefícios

Integrar análises avançadas de imagem e vídeo em seu aplicativo: adicione análises precisas de imagem e vídeo aos aplicativos sem experiência. A API do Amazon Rekognition permite a análise por meio de aprendizado profundo sem exigir nenhum conhecimento de machine learning. Você pode incorporar rapidamente a visão computacional em aplicativos da Web, móveis e de dispositivos.

Análise de imagens e vídeos baseada em aprendizado profundo: analisa imagens e vídeos usando aprendizado profundo para obter alta precisão. O Amazon Rekognition pode detectar rótulos, objetos, cenas, rostos e celebridades. Filtre os resultados para incluir/excluir rótulos específicos.

Análise de imagem dimensionável: analisa milhões de imagens para organizar grandes conjuntos de dados visuais. Ajusta a escala para lidar com o aumento do tráfego e das bibliotecas de imagens. Você não precisa planejar a capacidade e paga apenas pelo que usar.

Analise e filtre imagens com base em propriedades: analise e filtre imagens por propriedades, como qualidade, cor e conteúdo visual, e detecte a nitidez, o brilho e o contraste da imagem.

Integração com outros AWS serviços — o Amazon Rekognition se integra imediatamente ao S3 e ao Lambda. Você pode chamar o Amazon Rekognition APIs do Lambda e processar imagens no Amazon S3 sem mover dados. O Rekognition tem escalabilidade e segurança integradas usando o IAM. AWS

Baixo custo - Pay-as-you-go preços, sem mínimos ou compromissos. Nível gratuito disponível para começar. Economize mais à medida que o uso aumenta por meio de preços diferenciados. Econômico em relação às soluções internas.

Personalização simples: personalize a precisão para seu caso de uso com adaptadores. Forneça imagens de amostra para treinar os adaptadores. Melhore a detecção de objetos e rótulos para determinados domínios. Um jeito fácil de personalizar a análise sem experiência em ML.

Para obter mais informações, consulte o Amazon Rekognition FAQs.

Amazon Rekognition e elegibilidade para HIPAA

Este é um serviço qualificado da HIPAA. Para obter mais informações sobre AWS a Lei de Portabilidade e Responsabilidade de Seguros de Saúde dos EUA de 1996 (HIPAA) e o uso de AWS serviços para processar, armazenar e transmitir informações de saúde protegidas (PHI), consulte Visão geral da HIPAA.

Você é usuário do Amazon Rekognition pela primeira vez?

Se você for um usuário iniciante do Amazon Rekognition, recomendamos que você leia as seguintes seções na ordem:

1. Como o Amazon Rekognition funciona— Esta seção apresenta vários componentes do Amazon Rekognition com os quais você trabalha para criar uma experiência. end-to-end
2. Comece a usar o Amazon Rekognition — Nesta seção, você configura sua conta, instala o SDK que reflete o idioma de sua escolha e testa a API do Amazon Rekognition. Para obter uma lista das linguagens de programação compatíveis com o Amazon Rekognition, consulte Usando o Rekognition com um SDK AWS.
3. Como trabalhar com imagens — Esta seção fornece informações sobre o uso do Amazon Rekognition com imagens armazenadas em buckets do Amazon S3 e imagens carregadas de um sistema de arquivos local.
4. Trabalhar com operações de análise de vídeo armazenado — Esta seção fornece informações sobre o uso do Amazon Rekognition com vídeos armazenados em um bucket do Amazon S3.
5. Trabalhando com eventos de streaming de vídeo — Esta seção fornece informações sobre o uso do Amazon Rekognition com streaming de vídeos.

Como o Amazon Rekognition funciona

O Amazon Rekognition fornece dois conjuntos de APIs para análise visual:

- Amazon Rekognition Image para análise de imagem
- Amazon Rekognition Video para análise de vídeo

Análise de imagem

Com o Amazon Rekognition Image, seus aplicativos podem:

- Detectar objetos, cenas e conceitos em imagens
- Reconhecer celebridades
- Detectar texto em várias linguagens
- Detectar conteúdo ou imagens explícitos, impróprios ou violentos
- Detectar, analisar e comparar faces e atributos faciais, como idade e emoções
- Detectar a presença de EPI

Os casos de uso incluem aprimoramento de aplicativos de fotos, catalogação de imagens e moderação de conteúdo.

Análise de vídeo

Com o Amazon Rekognition Video, seus aplicativos podem:

- Rastrear pessoas e objetos em quadros de vídeo
- Reconhecer objetos
- Reconhecer celebridades
- Pesquisar vídeos armazenados e transmitidos por pessoas de interesse
- Analisar rostos em busca de atributos como idade e emoções
- Detectar conteúdo ou imagens explícitos, impróprios ou violentos
- Agregar e classificar os resultados da análise por registros de data e hora e segmentos
- Detectar pessoas, animais de estimação e pacotes em streaming de vídeo

Os casos de uso incluem análise de vídeo, catalogação de vídeos e filtragem de conteúdo impróprio.

Recursos principais

- Análise avançada de aprendizado profundo
- Detecção de alta precisão para objetos, cenas, faces, texto
- API fácil de usar para integração em aplicativos
- Modelos personalizáveis ajustados aos seus dados
- Análise dimensionável de bibliotecas de mídia

O Amazon Rekognition permite que você aprimore a precisão de determinados modelos de aprendizado profundo treinando um adaptador personalizado. Por exemplo, com o Amazon Rekognition Custom Moderation, você pode adaptar o modelo básico de análise de imagens do Amazon Rekognition treinando um adaptador personalizado com suas imagens. Consulte Enhancing accuracy with Custom Moderation para obter mais informações.

As seções a seguir abrangem os tipos de análise que o Amazon Rekognition fornece e uma visão geral das operações do Amazon Rekognition Image e do Amazon Rekognition Video. Inclui também a diferença entre as operações de armazenamento e de não armazenamento.

Para demonstrar o APIs Amazon Rekognition, você pode ver a Etapa 3: Começar a usar a AWS CLI e a API do AWS SDK, que abrange o teste do Rekognition no console. AWS

Tópicos

- Entender os tipos de análise do Rekognition
- Entender as operações de imagem e vídeo do Rekognition
- Entender operações de API sem armazenamento e com armazenamento
- Noções básicas do versionamento de modelo

Entender os tipos de análise do Rekognition

A seguir estão os tipos de análise que a API Amazon Rekognition Image e a API Amazon Rekognition Video podem realizar. Para obter informações sobre o APIs, consulteEntender as operações de imagem e vídeo do Rekognition.

A tabela a seguir lista as operações que você precisa usar com relação ao tipo de mídia com a qual você está trabalhando e seu caso de uso:

Caso de uso	Tipo de mídia	Operações
Como moderar um conteúdo	Imagens	DetectModerationLabels, StartMediaAnalysisJob, GetMediaAnalysisJob, ListMediaAnalysisJobs
	Vídeo armazenado	StartContentModeration, GetContentModeration
Verificação de identidade	Imagens	CreateCollection, CreateUser, IndexFaces, AssociateFaces, SearchFacesByImage, SearchUsersByImage
	Vídeo armazenado	CreateCollection, IndexFaces, StartFaceSearch, GetFaceSearch
	Streaming de vídeo (Detectando a vivacidade da face)	CreateFaceLivenessSession, StartFaceLivenessSession, GetFaceLivenessSessionResults,
Análise facial	Imagens	DetectFaces, CompareFaces
	Vídeo armazenado	StartFaceDetection, GetFaceDetection
	Streaming de vídeo	CreateStreamProcessor, StartStreamProcessor
Reconhecimento de objetos e atividades	Imagens	DetectLabels
	Vídeos armazenados	StartLabelDetection, GetLabelDetection
Casa conectada	Streaming de vídeo	StartStreamProcessor

Caso de uso	Tipo de mídia	Operações
Análise de mídia	Vídeo armazenado	StartSegmentDetection, GetSegmentDetection
Segurança no local de trabalho	Imagens	DetectProtectiveEquipment
Detecção de texto	Imagens	DetectText
	Vídeo	StartTextDetection, GetTextDetection
Pessoas trafegando	Vídeo	StartPersonTracking, GetPersonTracking
Reconhecimento de celebridades	Imagens	RecognizeCelebrities
	Vídeo	StartCelebrityRecognition, GetCelebrityRecognition
Detecção de rótulo personalizado	Imagens	DetectCustomLabels
	Treinamento de modelos	Consulte o guia do desenvolvedor de etiquetas personalizadas

Rótulos

Um rótulo se refere a qualquer um dos seguintes: objetos (por exemplo, flor, árvore ou mesa), eventos (por exemplo, um casamento, formatura ou festa de aniversário), conceitos (por exemplo, paisagem, noite e natureza) ou atividades (por exemplo, correr ou jogar basquete). O Amazon Rekognition pode detectar rótulos em imagens e vídeos. Para obter mais informações, consulte Detectando objetos e conceitos.

O Rekognition pode detectar uma grande lista de rótulos na imagem e no vídeo armazenado. O Rekognition também pode detectar um pequeno número de rótulos no streaming de vídeo.

Use as seguintes operações para detectar rótulos com base no seu caso de uso:

- Para detectar rótulos em imagens: Use DetectLabels. Você pode identificar as propriedades da imagem, como as cores dominantes e a qualidade da imagem. Para conseguir isso, use DetectLabelswith `IMAGE_PROPERTIES` como parâmetro de entrada.
- Para detectar marcadores em vídeos armazenados: Use StartLabelDetection. A detecção de cores e qualidade de imagem dominantes não é compatível com o vídeo armazenado.
- Para detectar rótulos em streaming de vídeo: Use CreateStreamProcessor. A detecção de cores e qualidade de imagem dominantes não é compatível com streaming de vídeo.

Você pode especificar quais tipos de rótulos deseja retornar para a detecção de rótulos de imagens e vídeos armazenados usando opções de filtragem inclusivas e exclusivas.

Etiquetas personalizadas

Os Amazon Rekognition Custom Labels podem identificar objetos e cenas em imagens que são específicos para suas necessidades comerciais treinando um modelo de machine learning. Por exemplo, é possível treinar um modelo para detectar logotipos ou detectar peças de máquinas de engenharia em uma linha de montagem.

> **Note**
>
> Para obter informações sobre as Amazon Rekognition Custom Labels, consulte o Guia do desenvolvedor de Amazon Rekognition Custom Labels.

O Amazon Rekognition fornece um console que você usa para criar, treinar, avaliar e executar um modelo de machine learning. Para mais informações, veja Conceitos básicos dos Amazon Rekognition Custom Labels no Guia de desenvolvimento de Amazon Rekognition Custom Labels. Você também pode usar a API Amazon Rekognition Custom Labels para treinar e executar um modelo. Para obter mais informações, consulte Conceitos básicos do Amazon Rekognition Custom Labels SDK no Guia do desenvolvedor do Amazon Rekognition. CustomLabels

Para analisar imagens usando um modelo treinado, use DetectCustomLabels.

Detecção de vivacidade facial

O Amazon Rekognition Face Liveness pode ajudar você a verificar se um usuário que está passando pela verificação de identidade facial está fisicamente presente na frente da câmera e não é um malfeitor falsificando a face do usuário. Ele detecta ataques falsos que são apresentados a uma câmera e ataques que ignoram uma câmera. Um usuário pode concluir uma verificação de vivacidade facial tirando uma pequena selfie em vídeo, e uma pontuação de vivacidade é retornada para a verificação. A vivacidade facial é determinada com um cálculo probabilístico e uma pontuação de confiança (entre 0 e 100) é retornada após a verificação. Quanto maior a pontuação, maior a confiança de que a pessoa que recebe o cheque está viva.

Para obter mais informações sobre o Face Liveness, consulte Detectando a vivacidade da face.

Detecção e análise facial

O Amazon Rekognition pode detectar faces em imagens e vídeos armazenados. Com o Amazon Rekognition, você pode obter informações sobre:

- Onde as faces são detectadas em uma imagem ou vídeo
- Marcos faciais, como a posição dos olhos
- A presença de oclusão facial nas imagens
- Emoções detectadas, como felicidade ou tristeza
- Direção do olhar de uma pessoa em imagens

Você também pode interpretar informações demográficas, como sexo ou idade. Você pode comparar uma face em uma imagem com faces detectadas em outra imagem. Informações sobre faces também podem ser armazenadas para recuperação posterior. Para obter mais informações, consulte Detectando e analisanda faces.

Para detectar faces em imagens, use DetectFaces. Para detectar faces em vídeos armazenados, use StartFaceDetection.

Pesquisa facial

O Amazon Rekognition pode pesquisar faces. As informações faciais são indexadas em um contêiner conhecido como uma coleção. As informações da face na coleção podem ser então correspondidas com as faces detectadas em imagens, vídeos armazenados e streaming de vídeo. Para obter mais informações, Pesquisa de faces em uma coleção.

Para pesquisar faces conhecidas em imagens, use DetectFaces. Para pesquisar faces conhecidas em vídeos armazenados, use StartFaceDetection. Para pesquisar faces conhecidas em streaming de vídeo, use CreateStreamProcessor.

Tráfego de pessoas

O Amazon Rekognition pode rastrear os caminhos das pessoas detectadas em um vídeo armazenado. O Amazon Rekognition Video fornece rastreamento de caminhos, detalhes faciais e informações de localização no quadro para pessoas detectadas em um vídeo. Para obter mais informações, consulte Pessoas trafegando.

Para detectar pessoas em vídeos armazenados, use StartPersonTracking.

Equipamento de proteção individual

O Amazon Rekognition pode detectar equipamentos de proteção individual (EPI) usados por pessoas detectadas em uma imagem. O Amazon Rekognition detecta coberturas faciais, protetores para mãos e coberturas para a cabeça. O Amazon Rekognition prevê se um item de EPI cobre a parte apropriada do corpo. Você também pode obter caixas delimitadoras para pessoas detectadas e itens de EPI. Para obter mais informações, consulte Detecção de equipamento de proteção individual.

Para detectar PPE em imagens, use DetectProtectiveEquipment.

Celebridades

O Amazon Rekognition pode reconhecer milhares de celebridades em imagens e vídeos armazenados. É possível obter informações sobre onde a face de uma celebridade está localizada em uma imagem, pontos de referência faciais e a pose da face de uma celebridade. Você pode obter informações de rastreamento de celebridades conforme elas aparecem em um vídeo armazenado. Você também pode obter mais informações sobre uma celebridade reconhecida, como a emoção expressa e a apresentação do gênero. Para obter mais informações, consulte Reconhecendo celebridades.

Para reconhecer celebridades em imagens, use RecognizeCelebrities. Para reconhecer celebridades em vídeos armazenados, use StartCelebrityRecognition.

Detecção de texto

O Amazon Rekognition Text in Image pode detectar texto em imagens e convertê-lo em texto legível por máquina. Para obter mais informações, consulte Detectar texto.

Para detectar texto em imagens, use DetectText.

Conteúdo impróprio ou ofensivo

O Amazon Rekognition pode analisar imagens e vídeos armazenados para conteúdo adulto e violento. Para obter mais informações, consulte Como moderar um conteúdo.

Para detectar imagens desprotegidas, use DetectModerationLabels. Para detectar vídeos armazenados desprotegidos, use StartContentModeration.

Personalização

Algumas análises de imagem APIs oferecidas pelo Rekognition permitem que você aprimore a precisão dos modelos de aprendizado profundo criando adaptadores personalizados treinados com base em seus próprios dados. Os adaptadores são componentes que se conectam ao modelo pré-treinado de aprendizado profundo do Rekognition, aprimorando sua precisão com conhecimento de domínio baseado em suas imagens. Você treina um adaptador para atender às suas necessidades fornecendo e anotando imagens de amostra.

Depois de criar um adaptador, você recebe um AdapterId. Você pode fornecer isso AdapterId a uma operação para especificar que deseja usar o adaptador que você criou. Por exemplo, você fornece o AdapterId para a DetectModerationLabelsAPI para análise síncrona de imagens. Fornecendo o AdapterId como parte da solicitação, o Rekognition o usará automaticamente para aprimorar as previsões de suas imagens. Isso permite que você aproveite os recursos do Rekognition enquanto o personaliza para atender às suas necessidades.

Você também tem a opção de obter previsões para imagens em massa com a StartMediaAnalysisJobAPI. Consulte Análise em massa para obter mais informações.

Você pode avaliar a precisão das operações do Rekognition fazendo upload de imagens para o console do Rekognition e executando análises nessas imagens. O Rekognition fará anotações em suas imagens usando o recurso selecionado e, em seguida, você poderá revisar as previsões usando as previsões verificadas para determinar quais rótulos se beneficiariam com a criação de um adaptador.

Atualmente, você pode usar adaptadores com o. DetectModerationLabels Para obter mais informações sobre como criar e usar adaptadores, consulte Aprimorando a precisão com moderação personalizada.

Análise em massa

O Rekognition Bulk Analysis permite processar uma grande coleção de imagens de forma assíncrona usando um arquivo de manifesto junto com a operação. StartMediaAnalysisJob Consulte Análise em massa para obter mais informações.

Entender as operações de imagem e vídeo do Rekognition

O Amazon Rekognition oferece dois conjuntos principais de APIs para análise de imagens e vídeos:

- Amazon Rekognition Image: essa API foi projetada para analisar imagens.
- Amazon Rekognition Video: essa API se concentra na análise de vídeos armazenados e em streaming.

Ambos APIs podem detectar várias entidades, como rostos e objetos. Para uma compreensão abrangente dos tipos de comparação e detecção com suporte, consulte a seção sobre Entender os tipos de análise do Rekognition.

Operações do Amazon Rekognition Image

As operações do Amazon Rekognition Image são síncronas. A entrada e a resposta estão em formato JSON. As operações do Amazon Rekognition Image analisam uma imagem de entrada que está no formato de imagem.jpg ou.png. A imagem passada para uma operação do Amazon Rekognition Image pode ser armazenada em um bucket do Amazon S3. Se você não estiver usando a AWS CLI, você também pode passar bytes de imagens codificadas em Base64 diretamente para uma operação do Amazon Rekognition. Para obter mais informações, consulte Trabalhar com imagens.

Operações de vídeo do Amazon Rekognition Video

A API do Amazon Rekognition Video facilita a análise de vídeos armazenados em um bucket do Amazon S3 ou transmitidos por meio do Amazon Kinesis Video Streams.

Para operações de vídeo armazenadas, observe o seguinte:

- As operações são assíncronas.
- A análise deve ser iniciada com uma operação “Iniciar” (por exemplo, StartFaceDetectionpara detecção facial em vídeos armazenados).

- O status de conclusão da análise é publicado em um tópico do Amazon SNS.
- Para recuperar os resultados de uma análise, use a operação “Obter” correspondente (por exemplo, GetFaceDetection).
- Para ter mais informações, consulte Trabalhar com operações de análise de vídeo.

Para análise de streaming de vídeo:

- Os recursos incluem pesquisa facial nas coleções de vídeos do Rekognition e detecção de rótulos (objetos ou conceitos).
- Os resultados da análise de rótulos são enviados como notificações do Amazon SNS e do Amazon S3.
- Os resultados de pesquisa facial são enviados a um fluxo de dados do Kinesis.
- O gerenciamento da análise de streaming de vídeo é feito por meio de um processador de streaming de vídeo Amazon Rekognition Video (por exemplo, crie um processador usando). CreateStreamProcessor
- Para ter mais informações consulte Trabalhar com eventos de vídeo em streaming.

Cada operação de análise de vídeo retorna metadados sobre o vídeo que está sendo analisado, bem como um ID de trabalho e uma tag de trabalho. Operações como Detecção de rótulo e Moderação de conteúdo para vídeo permitem classificar por carimbo de data/hora ou nome do rótulo e agregar os resultados por carimbo de data/hora ou segmento.

Operações não baseadas em armazenamento e baseadas em armazenamento

As operações do Amazon Rekognition estão agrupadas nas seguintes categorias.

- Operações de API que não são de armazenamento — Nessas operações, o Amazon Rekognition não mantém nenhuma informação. Você fornece imagens e vídeos de entrada, a operação realiza a análise e retorna os resultados, mas nada é salvo pelo Amazon Rekognition. Para obter mais informações, consulte Operações que não são de armazenamento.
- Operações de API baseadas em armazenamento — Os servidores Amazon Rekognition podem armazenar informações faciais detectadas em contêineres conhecidos como coleções. O Amazon Rekognition fornece operações adicionais de API que você pode usar para pesquisar

as informações persistentes da face em busca de correspondências faciais. Para obter mais informações, consulte Operações de API baseadas em armazenamento.

Usando o AWS SDK ou HTTP para chamar as operações da API Amazon Rekognition

Você pode chamar as operações da API Amazon Rekognition usando o AWS SDK ou diretamente usando HTTP. Você deve usar sempre o AWS SDK, a menos que tenha um bom motivo para não fazer isso. Os exemplos de Java nesta seção usam o AWS SDK. Embora um arquivo de projeto Java não seja fornecido, é possível usar o AWS Toolkit for Eclipse para desenvolver aplicativos da AWS usando Java.

Os exemplos do .NET nesta seção usam o AWS SDK para .NET. É possível usar o AWS Toolkit for Visual Studio para desenvolver aplicativos da AWS que usam o .NET. Ele inclui modelos úteis e o AWS Explorer para implantar aplicativos e gerenciar serviços.

A referência de API neste guia aborda a chamada de operações do Amazon Rekognition usando HTTP. Para obter informações de referência sobre Java, consulte AWS SDK para Java.

Os endpoints do serviço Amazon Rekognition que você pode usar estão documentados nas Regiões e endpoints da AWS.

Ao chamar o Amazon Rekognition com HTTP, use as operações HTTP POST.

Entender operações de API sem armazenamento e com armazenamento

O Amazon Rekognition fornece dois tipos de operações de API. São operações sem armazenamento, nas quais nenhuma informação é armazenada pelo Amazon Rekognition, e operações de armazenamento, nas quais determinadas informações faciais são armazenadas pelo Amazon Rekognition.

Operações que não são de armazenamento

O Amazon Rekognition fornece as seguintes operações de API sem armazenamento para imagens:

- DetectLabels

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

O Amazon Rekognition fornece as seguintes operações de API sem armazenamento para vídeos:

- StartLabelDetection
- StartFaceDetection
- StartPersonTracking
- StartCelebrityRecognition
- StartContentModeration

Essas são chamadas de operações de API sem armazenamento porque, quando você faz a chamada da operação, o Amazon Rekognition não mantém nenhuma informação descoberta sobre a imagem de entrada. Como todas as outras operações da API Amazon Rekognition, nenhum byte de imagem de entrada é persistido por operações de API que não sejam de armazenamento.

Os cenários de exemplo a seguir mostram onde você pode integrar operações da API de não armazenamento no aplicativo. Esses cenários pressupõem que você tem um repositório local de imagens.

Example 1: um aplicativo que encontra imagens no repositório local que contêm rótulos específicos

Primeiro, você detecta rótulos (objetos e conceitos) usando a operação `DetectLabels` do Amazon Rekognition em cada uma das imagens em seu repositório e cria um índice do lado do cliente, conforme mostrado a seguir:

```
Label        ImageID

tree          image-1
flower        image-1
mountain      image-1
```

```
tulip          image-2
flower         image-2
apple          image-3
```

Em seguida, o aplicativo pode pesquisar esse índice para encontrar imagens no repositório local que contenham um rótulo específico. Por exemplo, exibir imagens que contenham uma árvore.

Cada rótulo que o Amazon Rekognition detecta tem um valor de confiança associado. Ele indica o nível de confiança da imagem de entrada que contém esse rótulo. Você pode usar esse valor de confiança para, como opção, realizar a filtragem do lado do cliente adicional em rótulos de acordo com os requisitos do aplicativo relacionados ao nível de confiança na detecção. Por exemplo, caso você precise de rótulos precisos, convém filtrar e escolher apenas os rótulos com maior valor de confiança (como 95% ou superior). Caso o aplicativo não exija um valor de confiança mais alto, convém optar por filtrar rótulos com um valor de confiança mais baixo (próximo a 50%).

Example 2: um aplicativo para exibir imagens faciais aprimoradas

Primeiro, você pode detectar faces em cada uma das imagens em seu repositório local usando a operação `DetectFaces` do Amazon Rekognition e criar um índice do lado do cliente. Para cada face, a operação retorna metadados dentre os quais estão uma caixa delimitadora, pontos de referência faciais (por exemplo, a posição da boca e da orelha) e atributos faciais (por exemplo, sexo). Você pode armazenar esses metadados em um índice local do lado do cliente, conforme mostrado a seguir:

```
ImageID     FaceID     FaceMetaData

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

Neste índice, a chave primária é uma combinação de `ImageID` e `FaceID`.

Assim, você pode usar as informações no índice para aprimorar as imagens quando o aplicativo as exibe no repositório local. Por exemplo, convém adicionar uma caixa delimitadora em torno da face ou destacar traços faciais.

Operações de API baseadas em armazenamento

O Amazon Rekognition Image IndexFacessuporta a operação, que você pode usar para detectar faces em uma imagem e manter informações sobre características faciais detectadas em uma coleção do Amazon Rekognition. Este é um exemplo de uma operação da API com base em armazenamento, porque o serviço mantém informações no servidor.

O Amazon Rekognition Image fornece as seguintes operações de API de armazenamento:

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

O Amazon Rekognition Video fornece as seguintes operações de API de armazenamento:

- StartFaceSearch
- CreateStreamProcessor

Para armazenar informações faciais, você deve primeiro criar uma coleção de faces em uma das regiões da AWS em sua conta. Você especifica essa coleção de faces ao chamar a operação `IndexFaces`. Depois de criar uma coleção de faces e armazenar informações sobre o traço facial de todas as faces, você poderá pesquisar a coleção em busca de correspondências de face. Por exemplo, você pode detectar a maior face em uma imagem e pesquisar faces correspondentes em uma coleção chamando `searchFacesByImage.`

As informações faciais armazenadas em coleções pelo `IndexFaces` estão acessíveis às operações de vídeo do Amazon Rekognition Video. Por exemplo, é possível pesquisar em um vídeo pessoas cujas faces correspondem às de uma coleção existente chamando StartFaceSearch.

Para obter informações sobre a criação e o gerenciamento de coleções, consulte Pesquisa de faces em uma coleção.

> **Note**
>
> As coleções armazenam vetores faciais, que são representações matemáticas de faces. As coleções não armazenam imagens de faces.

Example 1: Um aplicativo que autentica o acesso a um edifício

Você inicia criando uma coleção de faces para armazenar imagens de crachás digitalizadas usando a operação `IndexFaces`, que extrai e armazena faces como vetores de imagem pesquisáveis. Em seguida, quando um funcionário entra no prédio, uma imagem da face dele é capturada e enviada para a operação `SearchFacesByImage`. Se a correspondência de face produzir uma pontuação de similaridade suficientemente alta (por exemplo, 99%), você poderá autenticar o funcionário.

Noções básicas do versionamento de modelo

O Amazon Rekognition usa modelos de aprendizado profundo para realizar a detecção facial e pesquisar faces em coleções. Ele continua melhorando a precisão de seus modelos com base no feedback dos clientes e os avanços na pesquisa de aprendizado profundo. Essas melhorias são fornecidas como atualizações no modelo. Por exemplo, com a versão 1.0 do modelo, IndexFaces pode indexar as 15 maiores faces em uma imagem. As versões posteriores do modelo permitem que `IndexFaces` faça a indexação das 100 maiores faces em uma imagem.

Quando você cria uma nova coleção, ela é associada à versão mais recente do modelo. Para melhorar a precisão, o modelo é ocasionalmente atualizado.

Quando uma nova versão do modelo é lançada, acontece o seguinte:

- Novas coleções que você cria são associados ao modelo mais recente. As faces que você adicionar às novas coleções usando IndexFaces serão detectadas usando o modelo mais recente.
- Suas coleções existentes continuam usando a versão do modelo com o qual elas foram criadas. Os vetores de face armazenados nessas coleções não são atualizados automaticamente para a versão mais recente do modelo.
- As novas faces que são adicionadas a uma coleção existente são detectadas usando o modelo que já está associado à coleção.

As diferentes versões do modelo não são compatíveis entre elas. Especificamente, se uma imagem for indexada em várias coleções que usam diferentes versões do modelo, os identificadores de faces para as mesmas faces detectadas serão diferentes. Se uma imagem for indexada em várias coleções associadas ao mesmo modelo, os identificadores de faces serão os mesmos.

O aplicativo poderá enfrentar problemas de compatibilidade se o gerenciamento da coleção não considerar as atualizações do modelo. É possível determinar a versão do modelo que uma coleção usa utilizando o campo `FaceModelVersion` que é retornado na resposta de uma operação de coleção (por exemplo, `CreateCollection`). Você pode obter a versão do modelo de uma coleção existente chamando o DescribeCollection. Para obter mais informações, consulte Descrever uma coleção.

Os vetores de faces existentes em uma coleção não podem ser atualizados para uma versão posterior do modelo. Como o Amazon Rekognition não armazena bytes da imagem de origem, ele não pode reindexar imagens automaticamente usando uma versão posterior do modelo.

Para usar o modelo posterior em faces que são armazenados em uma coleção existente, crie uma nova coleção (CreateCollection) e faça a reindexação das imagens de origem na nova coleção (`Indexfaces`). Você precisa atualizar quaisquer identificadores de faces que foram armazenados pelo seu aplicativo, porque na nova coleção os identificadores de face são diferentes dos identificadores da coleção antiga. Caso você não precise mais da coleção antiga, poderá excluí-la usando DeleteCollection.

Operações stateless, como DetectFaces, usam a versão mais recente do modelo.

Comece a usar o Amazon Rekognition

Esta seção fornece tópicos para você começar a usar o Amazon Rekognition. Se você é novo no Amazon Rekognition, recomendamos que você primeiro revise os conceitos e a terminologia apresentados em Como o Amazon Rekognition funciona.

Antes de usar o Rekognition, você precisará criar uma conta da AWS e obter um ID de conta da AWS. Você também desejará criar um usuário, que permita ao sistema Amazon Rekognition determinar se você tem as permissões necessárias para acessar seus recursos.

Depois de criar suas contas, você vai querer instalar e configurar o AWS CLI AWS SDKs e. Isso AWS CLI permite que você interaja com o Amazon Rekognition e outros serviços por meio da linha de comando, AWS SDKs enquanto permite que você use linguagens de programação como Java e Python para interagir com o Amazon Rekognition.

Depois de configurar o AWS CLI e AWS SDKs, você pode ver alguns exemplos de como usar os dois. Você também pode ver alguns exemplos de como interagir com o Amazon Rekognition usando o console.

O console do Amazon Rekognition permite que você use os fluxos de trabalho de análise em massa e moderação personalizada. Além desses recursos, você também pode usar o console para demonstrar vários recursos diferentes do Amazon Rekognition. Observe que o uso da API é necessário para usar a maioria dos recursos do Amazon Rekognition.

Tópicos

- Etapa 1: Configure uma conta da AWS e crie um usuário
- Etapa 2: configurar o AWS CLI e AWS SDKs
- Etapa 3: Começar a usar a API AWS CLI e AWS SDK
- Etapa 4: Começando a usar o console do Amazon Rekognition

Etapa 1: Configure uma conta da AWS e crie um usuário

Antes de usar o Amazon Rekognition pela primeira vez, você deve concluir as seguintes tarefas:

1. Inscreva-se em uma AWS conta.
2. Crie um usuário.

Esta seção do guia do desenvolvedor explica por que e como você criará uma conta e um usuário da AWS .

Tópicos

- Crie uma AWS conta e um usuário

Crie uma AWS conta e um usuário

Contas da AWS

Quando você se inscreve na Amazon Web Services (AWS), sua conta da AWS é automaticamente cadastrada em todos os serviços na AWS, incluindo o Amazon Rekognition. Você será cobrado apenas pelos serviços que usar.

Com o Amazon Rekognition, você paga somente pelos recursos que usa.

Se você é um novo AWS cliente, pode começar a usar o Amazon Rekognition gratuitamente. Para obter mais informações, consulte Nível de uso gratuito da AWS.

Consulte a próxima seção Inscreva-se para um Conta da AWS para obter instruções sobre a criação da conta.

Se você já tiver uma AWS conta, pule a configuração da conta e crie um usuário administrativo.

Usuários

Os serviços da AWS, como o Amazon Rekognition, exigem que você forneça credenciais ao acessá-los. Dessa maneira, o serviço pode determinar se você tem permissões para acessar os recursos próprios desse serviço.

Você pode criar chaves de acesso para sua AWS conta para acessar AWS CLI ou, APIs enquanto estiver usando o console, requer sua senha. No entanto, não recomendamos acessar a AWS usando as credenciais do usuário raiz da conta da AWS. Em vez disso, recomendamos que você use AWS Identity and Access Management (IAM) para criar um usuário administrativo.

Em seguida, você pode acessar a AWS usando uma URL especial e as credenciais desse usuário administrativo.

Se você se inscreveu na AWS, mas ainda não criou um usuário para si mesmo, você pode criar um usando o console do IAM. Consulte a próxima seção Criar um usuário com acesso administrativo para obter instruções sobre como criar um usuário administrativo.

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Abra a https://portal.aws.amazon.com/billing/inscrição.
2. Siga as instruções online.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica ou uma mensagem de texto e inserir um código de verificação pelo teclado do telefone.

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, você pode visualizar a atividade atual da sua conta e gerenciar sua conta acessando https://aws.amazon.com/e escolhendo Minha conta.

Criar um usuário com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS AWS IAM Identity Center, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login AWS Management Consolecomo proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, insira a senha.

   Para obter ajuda ao fazer login usando o usuário-raiz, consulte Fazer login como usuário-raiz no Guia do usuário do Início de Sessão da AWS .
2. Habilite a autenticação multifator (MFA) para o usuário-raiz.

   Para obter instruções, consulte Habilitar um dispositivo de MFA virtual para seu usuário Conta da AWS raiz (console) no Guia do usuário do IAM.

Criar um usuário com acesso administrativo

1. Habilita o Centro de Identidade do IAM.

   Para obter instruções, consulte Habilitar o AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center .
2. No Centro de Identidade do IAM, conceda o acesso administrativo a um usuário.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no Guia AWS IAM Identity Center do usuário.

Iniciar sessão como o usuário com acesso administrativo

- Para fazer login com o seu usuário do Centro de Identidade do IAM, use o URL de login enviado ao seu endereço de e-mail quando o usuário do Centro de Identidade do IAM foi criado.

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no portal de AWS acesso no Guia Início de Sessão da AWS do usuário.

Atribuir acesso a usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

   Para obter instruções, consulte Criar um conjunto de permissões no Guia do usuário do AWS IAM Identity Center .
2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Adicionar grupos no Guia do usuário do AWS IAM Identity Center .

Etapa 2: configurar o AWS CLI e AWS SDKs

Tópicos

- Conceder acesso programático
- Usando o Rekognition com um SDK AWS

As etapas a seguir mostram como instalar o AWS Command Line Interface (AWS CLI) e o AWS SDKs que os exemplos desta documentação usam. Há várias maneiras diferentes de autenticar chamadas do AWS SDK. Os exemplos deste guia pressupõem que você esteja usando um perfil de credenciais padrão para chamar AWS CLI comandos e operações de API do AWS SDK.

Para obter uma lista das AWS regiões disponíveis, consulte Regiões e endpoints no Referência geral da Amazon Web Services.

Siga as etapas para fazer download e configurar a AWS SDKs.

Para configurar o AWS CLI e o AWS SDKs

1. Baixe e instale o AWS CLIe o AWS SDKs que você deseja usar. Este guia fornece exemplos para Java AWS CLI, Python, Ruby, Node.js, PHP, .NET e. JavaScript Para obter informações sobre a instalação AWS SDKs, consulte Tools for Amazon Web Services.
2. Crie uma chave de acesso para o usuário criado por você em Crie uma AWS conta e um usuário.

   a. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.

   b. No painel de navegação, escolha Users.

   c. Selecione o nome do usuário criado por você em Crie uma AWS conta e um usuário.

   d. Selecione a guia Credenciais de segurança.

   e. Selecione Create access key (Criar chave de acesso). Em seguida, selecione o arquivo Download.csv para salvar o ID de chave de acesso e a chave de acesso secreta em um arquivo CSV em seu computador. Armazene o arquivo em um lugar seguro. Você não terá mais acesso à chave de acesso secreta depois que essa caixa de diálogo for fechada. Depois que tiver feito download do arquivo CSV, selecione Fechar.
3. Se você instalou o AWS CLI, você pode configurar as credenciais e a região para a maioria AWS SDKs digitando `aws configure` no prompt de comando. Caso contrário, siga todas as instruções abaixo.
4. Em seu computador, navegue para seu diretório inicial e crie um diretório `.aws`. Nos sistemas baseados em Unix, como Linux ou macOS, isso está no seguinte local:

   ```
   ~/.aws
   ```

   No Windows, isso está no seguinte local:

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

5. No diretório `.aws`, crie um novo arquivo chamado `credentials`.
6. Abra o arquivo CSV de credenciais que você criou na etapa 2 e copie o conteúdo dele para o arquivo `credentials` usando o seguinte formato:

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

   Substitua o ID de sua chave de acesso e a chave de acesso secreta por your_access_key_id e your_secret_access_key.
7. Salve o arquivo `Credentials` e exclua o arquivo CSV.
8. No diretório `.aws`, crie um novo arquivo chamado `config`.
9. Abra o arquivo `config` e insira sua região no seguinte formato.

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

   Substitua sua região da AWS desejada (por exemplo, `us-west-2`) por your_aws_region.

   > **Note**
   >
   > Se você não selecionar uma região, us-east-1 será usada por padrão.

10. Salve o arquivo `config`.

Conceder acesso programático

Você pode executar os exemplos de código AWS CLI e os exemplos deste guia em seu computador local ou em outros AWS ambientes, como uma instância do Amazon Elastic Compute Cloud. Para executar os exemplos, você precisa conceder acesso às operações do AWS SDK que os exemplos usam.

Tópicos

- Executando código em seu computador local

- Executando código em AWS ambientes

Executando código em seu computador local

Para executar código em um computador local, recomendamos que você use credenciais de curto prazo para conceder ao usuário acesso às operações do AWS SDK. Para obter informações específicas sobre como executar o AWS CLI e exemplos de código em um computador local, consulteUsando um perfil em seu computador local.

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando AWS.

Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
Identidade da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporárias para assinar solicitações programáticas para o AWS CLI AWS SDKs, ou. AWS APIs	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI para uso AWS IAM Identity Center no Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferramentas e AWS APIs, consulte a autenticação do IAM Identity Center no Guia de referência de ferramentas AWS SDKs e ferramentas.
IAM	Use credenciais temporárias para assinar solicitações programáticas para o AWS	Siga as instruções em Como usar credenciais temporárias com AWS recursos no Guia do usuário do IAM.

Qual usuário precisa de acesso programático?	Para	Por
	CLI AWS SDKs, ou. AWS APIs	
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitaç ões programáticas para o AWS CLI, AWS SDKs, ou. AWS APIs	Siga as instruções da interface que deseja utilizar. • Para isso AWS CLI, consulte Autenticação usando credenciais de usuário do IAM no Guia do AWS Command Line Interface usuário. • Para ferramentas AWS SDKs e ferramentas, consulte Autenticar usando credenciais de longo prazo no Guia de referência de ferramentas AWS SDKs e ferramentas. • Para isso AWS APIs, consulte Gerenciamento de chaves de acesso para usuários do IAM no Guia do usuário do IAM.

Usando um perfil em seu computador local

Você pode executar os exemplos de código AWS CLI e de código neste guia com as credenciais de curto prazo que você criou. Executando código em seu computador local Para obter as credenciais e outras informações de configurações, os exemplos usam um perfil chamado `profile-name`, por exemplo:

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

O usuário que o perfil representa deve ter permissões para chamar as operações do SDK do Rekognition e outras operações do SDK exigidas pelos exemplos. AWS

Para criar um perfil que funcione com os exemplos de código AWS CLI e, escolha uma das opções a seguir. Verifique se o nome do perfil que você criou é `profile-name`.

- Usuários gerenciados pelo IAM — Siga as instruções em Mudar para um perfil do IAM (AWS CLI).
- Identidade da força de trabalho (usuários gerenciados por AWS IAM Identity Center) — Siga as instruções em Configuração da AWS CLI para uso. AWS IAM Identity Center Para os exemplos de código, recomendamos o uso de um ambiente de desenvolvimento integrado (IDE), que oferece suporte ao AWS Toolkit, permitindo a autenticação por meio do IAM Identity Center. Para ver os exemplos de Java, consulte Começar a criar com Java. Para ver os exemplos de Python, consulte Começar a criar com Python. Para obter mais informações, consulte Credenciais do IAM Identity Center.

> **Note**
>
> Você pode usar o código para obter credenciais de curto prazo. Para obter mais informações, consulte Mudar para um perfil do IAM (API da AWS). Para o IAM Identity Center, obtenha as credenciais de curto prazo para uma função seguindo as instruções em Obter credenciais de perfil do IAM para acesso à CLI.

Executando código em AWS ambientes

Você não deve usar as credenciais do usuário para assinar chamadas do AWS SDK em AWS ambientes, como código de produção executado em uma AWS Lambda função. Em vez disso, você configura uma função que define as permissões de que seu código precisa. Em seguida, você atribui a função ao ambiente em que seu código é executado. A forma como você atribui a função e disponibiliza credenciais temporárias varia de acordo com o ambiente em que seu código é executado:

- AWS Lambda função — Use as credenciais temporárias que o Lambda fornece automaticamente à sua função quando assume a função de execução da função Lambda. As credenciais estão disponíveis nas variáveis de ambiente do Lambda. Você não precisa especificar um perfil. Para obter mais informações, consulte Função de execução do Lambda.

- Amazon EC2 — Use o provedor de credenciais de endpoint de metadados da EC2 instância Amazon. O provedor gera e atualiza automaticamente as credenciais para você usando o perfil de EC2 instância da Amazon que você anexa à instância da Amazon EC2 . Para obter mais informações, consulte Como usar uma função do IAM para conceder permissões a aplicativos executados em EC2 instâncias da Amazon
- Amazon Elastic Container Service — Use o provedor de credenciais do Container. O Amazon ECS envia e atualiza as credenciais para um endpoint de metadados. Um perfil do IAM de tarefa que você especifica fornece uma estratégia para gerenciar as credenciais que seu aplicativo usa. Para obter mais informações, consulte Interagir com os serviços da AWS.

Para obter mais informações sobre provedores de credenciais, consulte Provedores padronizados de credenciais.

Usando o Rekognition com um SDK AWS

AWS kits de desenvolvimento de software (SDKs) estão disponíveis para muitas linguagens de programação populares. Cada SDK fornece uma API, exemplos de código e documentação que permitem que os desenvolvedores criem facilmente aplicações em seu idioma de preferência.

Documentação do SDK	Exemplos de código
AWS SDK para C++	AWS SDK para C++ exemplos de código
AWS CLI	AWS CLI exemplos de código
AWS SDK para Go	AWS SDK para Go exemplos de código
AWS SDK para Java	AWS SDK para Java exemplos de código
AWS SDK para JavaScript	AWS SDK para JavaScript exemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin exemplos de código
AWS SDK para .NET	AWS SDK para .NET exemplos de código
AWS SDK para PHP	AWS SDK para PHP exemplos de código
Ferramentas da AWS para PowerShell	Ferramentas da AWS para PowerShell exemplos de código

Documentação do SDK	Exemplos de código
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) exemplos de código
AWS SDK para Ruby	AWS SDK para Ruby exemplos de código
AWS SDK para Rust	AWS SDK para Rust exemplos de código
SDK da AWS para SAP ABAP	SDK da AWS para SAP ABAP exemplos de código
AWS SDK for Swift	AWS SDK for Swift exemplos de código

Para exemplos específicos do Rekognition, consulte Exemplos de código para o Amazon Rekognition usando AWS SDKs.

> **Exemplo de disponibilidade**
>
> Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na parte inferior desta página.

Etapa 3: Começar a usar a API AWS CLI e AWS SDK

Depois de configurar o AWS CLI e AWS SDKs que você deseja usar, você pode criar aplicativos que usam o Amazon Rekognition. A maioria das interações com o Amazon Rekognition acontece usando as operações de API, embora alguns recursos selecionados do Amazon Rekognition tenham fluxos de trabalho de console.

Os tópicos a seguir mostram como começar a usar o Amazon Rekognition Image e o Amazon Rekognition Video por meio do ou do. AWS CLI AWS SDKs

- Como trabalhar com imagens: aborda o processo de analisar imagens com o Amazon Rekognition Image.
- Trabalhar com operações de análise de vídeo armazenado: aborda o processo de analisar vídeo armazenado e sem streaming com o Amazon Rekognition Video.

- Trabalhando com eventos de streaming de vídeo: aborda o processo de analisar streaming de vídeo com o Amazon Rekognition Video.

As seções listadas acima têm exemplos que usam a AWS CLI. Se você pretende usar o AWS CLI, consulte a seção a seguir para obter informações sobre como formatar suas chamadas de API.

Formatando os exemplos AWS CLI

Os AWS CLI exemplos neste guia estão formatados para o sistema operacional Linux. Para usar as amostras com o Microsoft Windows, você precisa alterar a formatação JSON do parâmetro `--image` e mudar as quebras de linha de barras invertidas (\) para acentos circunflexos (^). Para obter mais informações sobre a formatação JSON, consulte Especificação dos valores de parâmetro para a interface da linha de comando da AWS.

Veja a seguir um exemplo de AWS CLI comando formatado para o Microsoft Windows (observe que esses comandos não serão executados como estão, são apenas exemplos de formatação):

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

Você também pode fornecer uma versão abreviada do JSON que funcione no Microsoft Windows e no Linux.

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

Para obter mais informações, consulte Como usar a sintaxe abreviada com a interface da linha de comando da AWS.

Próxima etapa

Etapa 4: Começando a usar o console do Amazon Rekognition

Etapa 4: Começando a usar o console do Amazon Rekognition

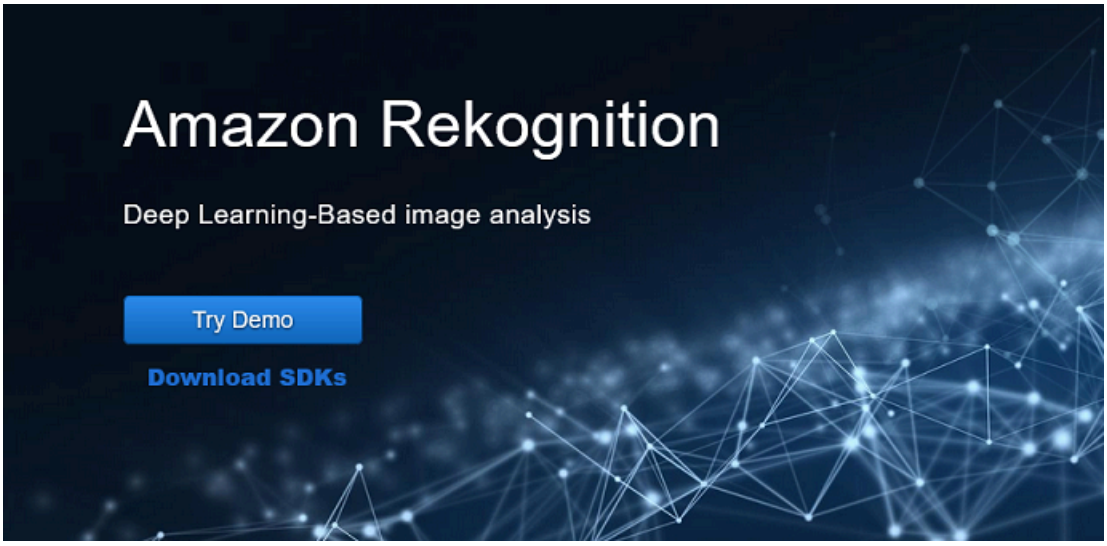
O console do Amazon Rekognition permite que você gerencie recursos relacionados aos rótulos personalizados do Rekognition e ao atributo de moderação personalizada. O console fornece apenas demonstrações de outros atributos do Rekognition.

Esta seção mostra como usar um subconjunto dos recursos do Amazon Rekognition, como detecção de objetos e cenas, análise facial e comparação de faces em um conjunto de imagens. Para obter mais informações, consulte Como o Amazon Rekognition funciona. Você também pode usar a AWS CLI API Amazon Rekognition ou para detectar objetos e cenas, detectar rostos e comparar e pesquisar faces. Para obter mais informações, consulte Etapa 3: Começar a usar a API AWS CLI e AWS SDK.

Esta seção também mostra como ver as CloudWatch métricas agregadas da Amazon para o Rekognition usando o console do Rekognition.

Tópicos

- Configurar permissões do console
- Exercício 1: Detecte objetos e cenas (console)
- Exercício 2: Analisar faces em uma imagem (console)
- Exercício 3: Compare faces em imagens (console)
- Exercício 4: Ver métricas agregadas (console)



Configurar permissões do console

Para usar o console do Rekognition, você precisa ter as permissões apropriadas para a função ou conta que está acessando o console. Para algumas operações, o Rekognition criará automaticamente um bucket do Amazon S3 para armazenar arquivos manipulados durante a operação. Se quiser armazenar seus arquivos de treinamento em um bucket diferente desse bucket de console, você precisará de permissões adicionais.

Permitindo acesso ao console

Para usar o console do Rekognition, você pode usar uma política do IAM como a seguinte, que abrange o Amazon S3 e o console do Rekognition. Para obter informações sobre como atribuir permissões, consulte Atribuição de permissões.

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

Acessando buckets externos do Amazon S3

Quando você abre o console do Rekognition pela primeira vez em uma nova AWS região, o Rekognition cria um bucket (bucket do console) que é usado para armazenar arquivos do projeto. Como alternativa, você pode usar seu próprio bucket do Amazon S3 (bucket externo) para carregar as imagens ou o arquivo de manifesto no console. Para usar um bucket externo, adicione o seguinte bloco de políticas à política anterior.

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

Atribuindo permissões

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos no Centro de Identidade do AWS IAM (sucessor do AWS Single Sign-On):

Crie um conjunto de permissões. Siga as instruções em Criar um conjunto de permissões no Guia do usuário do Centro de Identidade do AWS IAM (sucessor do AWS Single Sign-On).

- Usuários gerenciados no IAM com provedor de identidades:

  Crie um perfil para a federação de identidades. Siga as instruções em Criando um perfil para um provedor de identidades de terceiros (federação) no Guia do Usuário do IAM.

- Usuários do IAM:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de um perfil para um usuário do IAM no Guia do usuário do IAM.
  - (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adição de permissões a um usuário (console) no Guia do usuário do IAM.

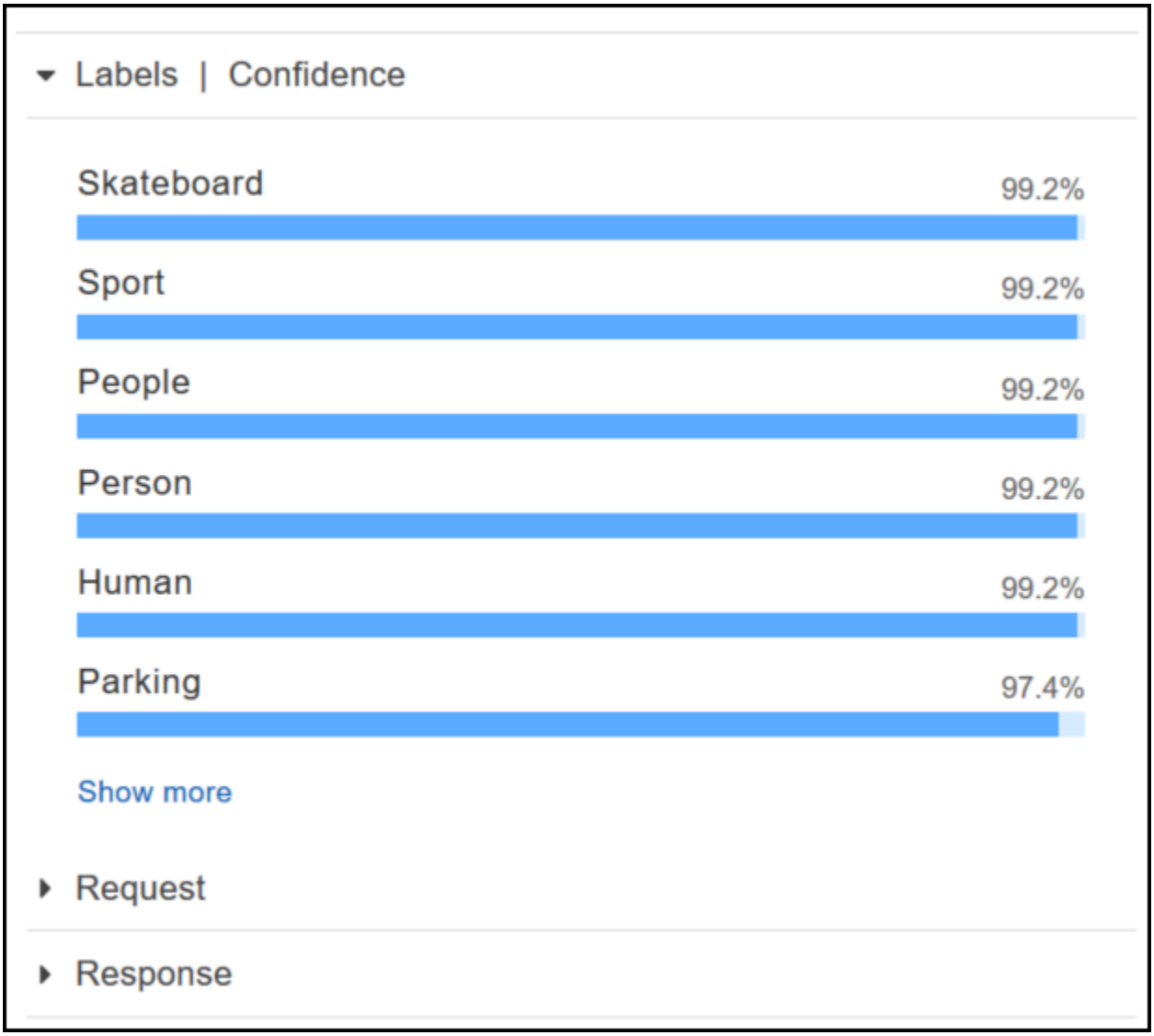
Exercício 1: Detecte objetos e cenas (console)

Esta seção mostra como, em um nível muito alto, a capacidade de detecção de objetos e cenas do Amazon Rekognition funciona. Quando você especifica uma imagem como entrada, o serviço detecta os objetos e as cenas na imagem e os retorna com uma pontuação de confiança percentual para cada objeto e cena.

Por exemplo, o Amazon Rekognition detecta os seguintes objetos e cenas na imagem de amostra: skate, esporte, pessoa, automóvel, carro e veículo.



O Amazon Rekognition também retorna uma pontuação de confiança para cada objeto detectado na imagem de amostra, conforme mostrado na resposta de amostra a seguir.



Para ver todas as pontuações de confiança mostradas na resposta, selecione Show more (Mostrar mais) no painel Labels | Confidence (Rótulos | Confiança).

Você também pode observar a solicitação para a API e a resposta da API como referência.

Solicitação

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

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

Resposta

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


Para obter mais informações, consulte Como o Amazon Rekognition funciona.

Detecte objetos e cenas em uma imagem que você fornece

Você pode fazer upload de uma imagem de sua propriedade ou fornecer o URL de uma imagem como entrada no console do Amazon Rekognition. O Amazon Rekognition retorna o objeto e as cenas, as pontuações de confiança de cada objeto e a cena que ele detecta na imagem que você fornece.

> **Note**
>
> A imagem deve ter menos de 5 MB e ser do formato JPEG ou PNG.

Para detectar objetos e cenas em uma imagem fornecida por você

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Detecção de rótulos.
3. Execute um destes procedimentos:
   - Carregue uma imagem – escolha Upload, vá até o local onde você armazenou a imagem e selecione-a.
   - Use uma URL – digite a URL na caixa de texto e escolha Go.
4. Exiba a pontuação de confiança de cada rótulo detectado no painel Labels | Confidence.

Para obter mais opções de análise de imagem, consulte the section called “Como trabalhar com imagens”.

Detecte objetos e pessoas em um vídeo que você fornece

Você pode fazer upload de um vídeo que você fornece como entrada no console do Amazon Rekognition. O Amazon Rekognition retorna as pessoas, objetos e rótulos detectados no vídeo.

> **Note**
>
> O vídeo de demonstração não deve ter mais de um minuto ou mais de 30 MB. Ele deve estar em formato de MP4 arquivo e codificado usando o codec H.264.

Para detectar objetos e pessoas em um vídeo, forneça

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Análise de vídeo armazenado na barra de navegação.
3. Em Escolha uma amostra ou faça upload de sua própria, selecione Seu próprio vídeo no menu suspenso.
4. Arraste e solte seu vídeo ou selecione-o no local onde você o armazenou.

Para obter mais opções de análise de vídeo, consulte the section called “Trabalhar com operações de análise de vídeo armazenado” ou the section called “Trabalhando com eventos de streaming de vídeo”.

Exercício 2: Analisar faces em uma imagem (console)

Esta seção mostra como usar o console do Amazon Rekognition para detectar faces e analisar atributos faciais em uma imagem. Quando você fornece uma imagem que contém uma face como entrada, o serviço detecta a face na imagem, analisa os atributos faciais e retorna uma pontuação de confiança percentual para a face e os atributos faciais detectados na imagem. Para obter mais informações, consulte Como o Amazon Rekognition funciona.

Por exemplo, se você escolher a seguinte imagem de amostra como entrada, o Amazon Rekognition a detecta como um rosto e retorna pontuações de confiança para a face e os atributos faciais detectados.



A tabela a seguir mostra a resposta de amostra.

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 houver várias faces na imagem de entrada, o Rekognition detectará até 100 faces na imagem. Toda face detectada é marcada com um quadrado. Quando você clica na área marcada com um quadrado em uma face, o Rekognition exibe a pontuação de confiança dessa face e seus atributos detectados no painel Faces | Confiança.

Analise faces em uma imagem que você fornece

Você pode fazer upload de sua própria imagem ou fornecer o URL da imagem no console do Amazon Rekognition.

> **Note**
>
> A imagem deve ter menos de 5 MB e ser do formato JPEG ou PNG.

Para analisar uma face em uma imagem fornecida por você

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Facial analysis.
3. Execute um destes procedimentos:
   - Carregue uma imagem – escolha Upload, vá até o local onde você armazenou a imagem e selecione-a.
   - Use uma URL – digite a URL na caixa de texto e escolha Go.
4. Exiba a pontuação de confiança de uma das faces detectadas e seus traços faciais no painel Faces | Confidence pane.
5. Se houver várias faces na imagem, escolha uma das outras faces para ver os atributos e as pontuações.

Exercício 3: Compare faces em imagens (console)

Esta seção mostra como usar o console do Amazon Rekognition para comparar faces em um conjunto de imagens com várias faces. Quando você especifica uma imagem de face de referência (origem) e uma imagem de Comparação de faces (de destino), o Rekognition compara a maior face na imagem de origem (ou seja, a face de referência) com até 100 faces detectadas na imagem de destino (ou seja, as faces de comparação) e, em seguida, descobre até que ponto a face na fonte

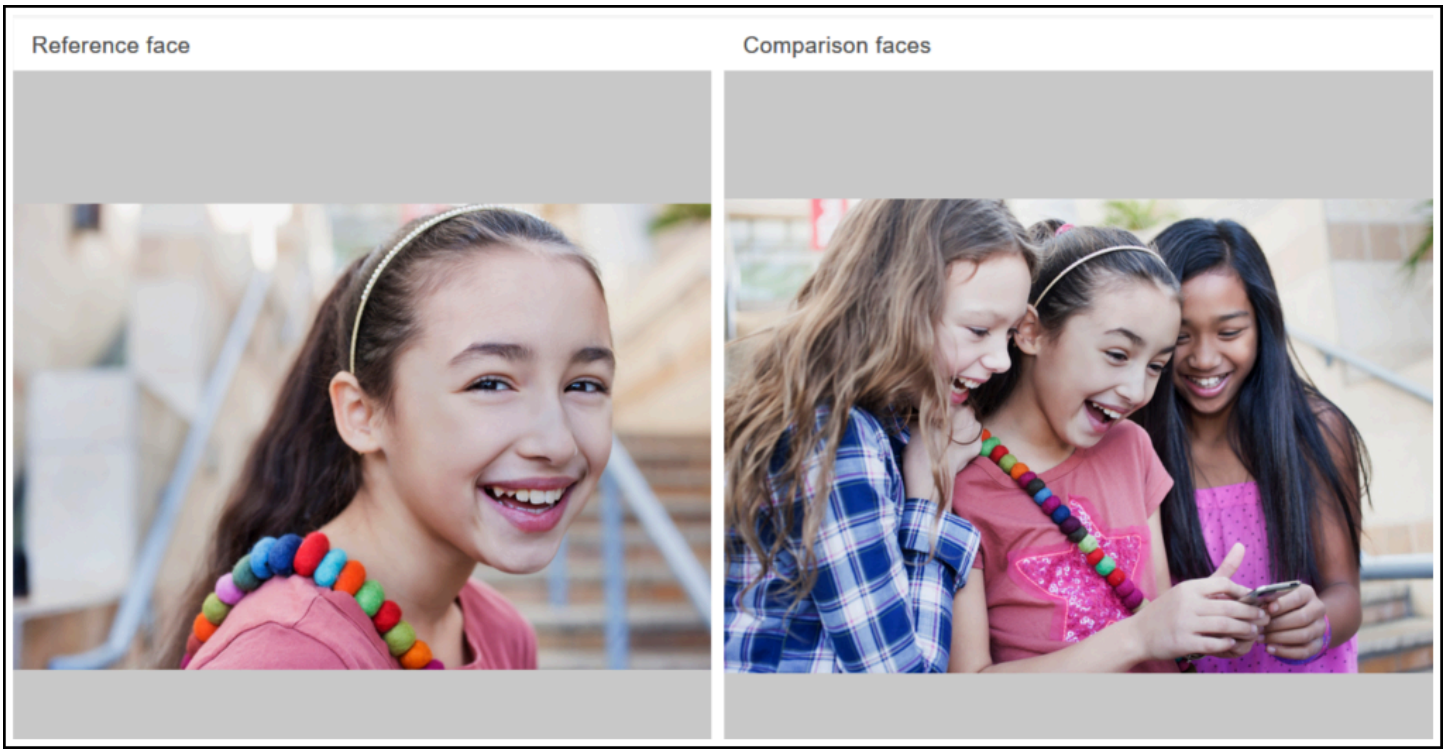
corresponde às faces na imagem de destino. A pontuação da semelhança de cada comparação é exibida no painel Results.

Se a imagem de destino contiver várias faces, o Rekognition combina a face na imagem de origem com até 100 faces detectadas na imagem de destino e, em seguida, atribui uma pontuação de similaridade a cada correspondência.

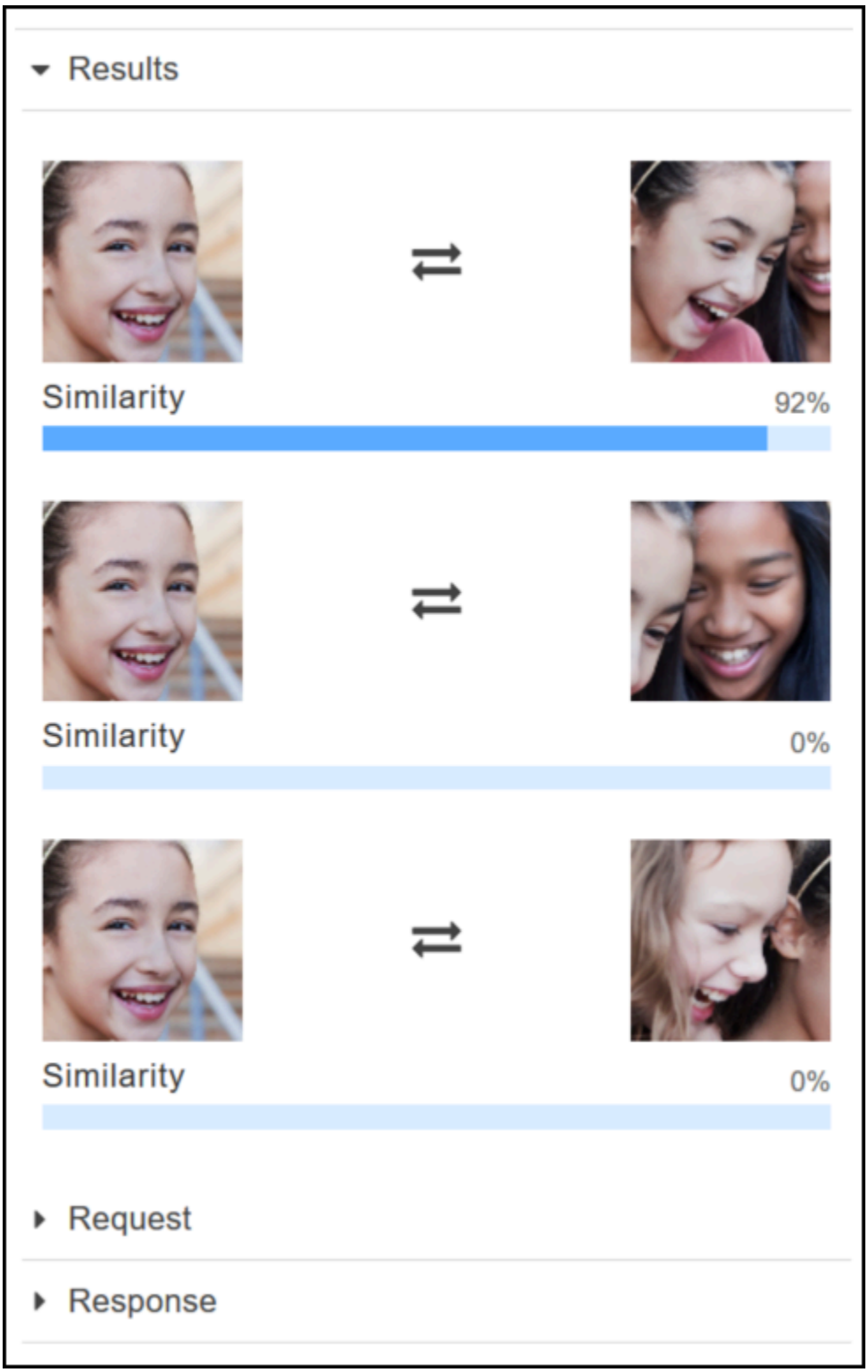
Se a imagem de origem contiver várias faces, o serviço detectará a maior na imagem de origem e a usará na comparação com cada face detectada na imagem de destino.

Para obter mais informações, consulte Comparanda faces em imagens.

Por exemplo, com a imagem de amostra mostrada à esquerda como imagem de origem e a imagem de amostra à direita como imagem de destino, o Rekognition detecta a face na imagem de origem, a compara com cada face detectada na imagem alvo e exibe uma pontuação de similaridade para cada par.



A tabela a seguir mostra as faces detectadas na imagem de destino e a pontuação de similaridade para cada face.



Compare faces em uma imagem que você fornece

Você pode enviar suas próprias imagens de origem e destino para o Rekognition para comparar as faces nas imagens ou especificar um URL para a localização das imagens.

> **Note**
>
> A imagem deve ter menos de 5 MB e ser do formato JPEG ou PNG.

Para comparar faces nas imagens

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. Escolha Face comparison.
3. Para a imagem de origem, faça o seguinte:
   - Carregar uma imagem – escolha Upload à esquerda, vá até o local onde você armazenou a imagem de origem e selecione a imagem.
   - Usar uma URL – digite a URL da imagem de origem na caixa de texto e escolha Go.
4. Para a imagem de destino, faça o seguinte:
   - Carregar uma imagem – escolha Upload à direita, vá até o local onde você armazenou a imagem de origem e selecione a imagem.
   - Usar uma URL – digite a URL da imagem de origem na caixa de texto e escolha Go.
5. O Rekognition combina a maior face em sua imagem de origem com até 100 faces na imagem de destino e, em seguida, exibe a pontuação de similaridade de cada par no painel Resultados.

Exercício 4: Ver métricas agregadas (console)

O painel de métricas do Amazon Rekognition mostra gráficos de atividades para um agregado de métricas individuais do Rekognition durante um período de tempo especificado. Por exemplo, a métrica agregada `SuccessfulRequestCount` mostra o número total de solicitações bem-sucedidas para todas as operações da API Rekognition nos últimos sete dias.

A tabela a seguir lista os gráficos exibidos no painel de métricas do Rekognition e a métrica correspondente do Rekognition. Para obter mais informações, consulte CloudWatchmétricas para Rekognition.

Gráfico	Métrica agregada
Chamadas bem-sucedidas	SuccessfulRequestCount
Erros de cliente	UserErrorCount
Erros de servidor	ServerErrorCount
Limitados	ThrottledCount
Rótulos detectados	DetectedLabelCount
Faces detectadas	DetectedFaceCount

Cada gráfico mostra dados de métrica agregada coletados por um período especificado. A contagem total de dados de métrica agregada para o período também é exibida. Para ver métricas de chamadas à API individuais, escolha o link abaixo de cada gráfico.

Para permitir que os usuários acessem o painel de métricas do Rekognition, certifique-se de que o usuário tenha as permissões apropriadas e do Rekognition. CloudWatch Por exemplo, um usuário com permissões de política gerenciada `AmazonRekognitionReadOnlyAccess` e `CloudWatchReadOnlyAccess` pode ver o painel de métricas. Caso um usuário não tenha as permissões obrigatórias, quando o usuário abre o painel de métricas, nenhum gráfico é exibido. Para obter mais informações, consulte Gerenciamento de identidade e acesso para o Amazon Rekognition.

Para obter mais informações sobre o monitoramento do Rekognition com, consulte. CloudWatch Monitorando o Rekognition com a Amazon CloudWatch

Para ver métricas agregadas (console)

1. Abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. No painel de navegação, selecione Métricas.
3. Na lista suspensa, selecione o período cuja métrica você deseja ver.
4. Para atualizar os gráficos, escolha o botão Refresh.
5. Para ver CloudWatch métricas detalhadas de uma métrica agregada específica, escolha Ver detalhes CloudWatch abaixo do gráfico métrico.

Trabalhando com imagens e vídeos

Você pode usar as operações da API do Amazon Rekognition com três tipos diferentes de mídia: imagens, vídeos armazenados e streaming de vídeos. Esta seção fornece informações gerais sobre como escrever código que acessa o Amazon Rekognition para processar os diferentes tipos de mídia. Para obter orientação sobre as melhores práticas e considerações, consulte as respectivas seções listadas abaixo, dependendo do tipo de mídia que você está processando.

Outras seções deste guia fornecem informações sobre tipos específicos de análise de imagem e vídeo, como a detecção de faces.

Tópicos

- Como trabalhar com imagens
- Trabalhar com operações de análise de vídeo armazenado
- Trabalhando com eventos de streaming de vídeo
- Gerenciamento de erros
- Usar o Amazon Rekognition como um serviço autorizado pelo FedRAMP

Como trabalhar com imagens

Esta seção aborda os tipos de análise que o Amazon Rekognition Image pode realizar em imagens.

- Detecção de objetos e cenas
- Detecção e comparação de faces
- Pesquisar faces em uma coleção
- Reconhecimento de celebridades
- Moderação de imagem
- Texto na detecção de imagens

Elas são realizadas por operações de API sem armazenamento, nas quais o Amazon Rekognition Image não mantém nenhuma informação descoberta pela operação. Nenhum byte da imagem de entrada é mantido pelas operações da API de não armazenamento. Para obter mais informações, consulte Entender operações de API sem armazenamento e com armazenamento.

O Amazon Rekognition Image também pode armazenar metadados faciais em coleções para recuperação posterior. Para obter mais informações, consulte Pesquisa de faces em uma coleção.

Nesta seção, você usa as operações da API Amazon Rekognition Image para analisar imagens armazenadas em um bucket do Amazon S3 e bytes de imagens carregadas do sistema de arquivos local. Esta seção também aborda a obtenção das informações de orientação de uma imagem .jpg.

O Rekognition usa apenas canais RGB para realizar inferências. AWS recomenda que os usuários removam o Alpha Channel antes de usar um monitor para inspecionar visualmente (manualmente por um humano) a comparação.

Tópicos

- Especificações de imagem
- Analisando imagens armazenadas em um bucket do Amazon S3
- Analisar uma imagem carregada de um sistema de arquivos local
- Exibir caixas delimitadoras
- Obter a orientação e as coordenadas da caixa delimitadora da imagem

Especificações de imagem

As operações do Amazon Rekognition Image podem analisar imagens no formato .jpg ou .png.

Você passa bytes de imagem para uma operação do Amazon Rekognition Image como parte da chamada ou faz referência a um objeto existente do Amazon S3. Para obter um exemplo de análise de uma imagem armazenada em um bucket do Amazon S3, consulte Analisando imagens armazenadas em um bucket do Amazon S3. Para obter um exemplo de transmissão de bytes de imagem para uma operação da API Amazon Rekognition Image, consulte Analisar uma imagem carregada de um sistema de arquivos local.

Se você usar HTTP e passar os bytes de imagem como parte de uma operação do Amazon Rekognition Image, os bytes de imagem deverão ser uma cadeia de caracteres codificada em base64. Se você usar o AWS SDK e passar bytes da imagem como parte da chamada à operação da API, a necessidade de codificar os bytes da imagem em base64 dependerá do idioma usado.

As seguintes imagens comuns codificam AWS SDKs automaticamente em base64, e você não precisa codificar bytes de imagem antes de chamar uma operação da API Amazon Rekognition Image.

- Java
- JavaScript
- Python
- PHP

Se você estiver usando outro AWS SDK e receber um erro de formato de imagem ao chamar uma operação da API Rekognition, tente codificar os bytes de imagem com base 64 antes de passá-los para uma operação da API Rekognition.

Se você usar o AWS CLI para chamar as operações do Amazon Rekognition Image, a transmissão de bytes de imagem como parte da chamada não é suportada. Você deve primeiramente carregar a imagem em um bucket do Amazon S3 e, em seguida, chamar a operação fazendo referência à imagem carregada.

> **Note**
>
> A imagem não precisará estar codificada em base64 se você passar uma imagem armazenada em um `S3Object`, em vez dos bytes da imagem.

Para obter mais informações sobre como garantir a menor latência possível para as operações do Amazon Rekognition Image, consulte Latência de operação do Amazon Rekognition Image.

Corrigir a orientação da imagem

Em várias operações da API Rekognition, a orientação de uma imagem analisada é retornada. É importante saber a orientação da imagem, pois permite que você reoriente as imagens para exibição. As operações da API Rekognition que analisam faces também retornam caixas delimitadoras para a localização de faces em uma imagem. Você pode usar caixas delimitadoras para exibir uma caixa em torno de uma face em uma imagem. As coordenadas da caixa delimitadora retornadas são afetadas pela orientação da imagem e pode ser necessário converter as coordenadas da caixa delimitadora para exibir corretamente uma caixa em torno de uma face. Para obter mais informações, consulte Obter a orientação e as coordenadas da caixa delimitadora da imagem.

Redimensionamento de imagem

Durante a análise, o Amazon Rekognition redimensiona internamente as imagens usando um conjunto de intervalos predefinidos que melhor se adequam a um modelo ou algoritmo específico.

Por esse motivo, o Amazon Rekognition pode detectar um número diferente de objetos ou fornecer resultados diferentes, dependendo da resolução da imagem de entrada. Por exemplo, suponha que você tenha duas imagens. A primeira imagem tem uma resolução de 1024x768 pixels. A segunda imagem, uma versão redimensionada da primeira imagem, tem uma resolução de 640x480 pixels. Se você enviar as imagens para DetectLabels, as respostas das duas chamadas para `DetectLabels` podem ser um pouco diferentes.

Analisando imagens armazenadas em um bucket do Amazon S3

O Amazon Rekognition Image pode analisar imagens armazenadas em um bucket do Amazon S3 ou imagens fornecidas como bytes de imagem.

Neste tópico, você usa a operação de DetectLabelsAPI para detectar objetos, conceitos e cenas em uma imagem (JPEG ou PNG) armazenada em um bucket do Amazon S3. Você passa uma imagem para uma operação da API Amazon Rekognition Image usando o parâmetro de entrada Image. Em `Image`, especifique a propriedade de objeto S3Object para fazer referência a uma imagem armazenada em um bucket S3. Os bytes de imagem para imagens armazenadas nos buckets do Amazon S3 não precisam ser codificados em base64. Para obter mais informações, consulte Especificações de imagem.

Exemplo de solicitação

Nesta solicitação JSON de exemplo de `DetectLabels`, a imagem de origem (`input.jpg`) é carregada de um `amzn-s3-demo-bucket` nomeado pelo bucket do Amazon S3. A região do bucket do S3 que contém o objeto S3 deve corresponder à região que você usa para operações do Amazon Rekognition Image.

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

Os exemplos a seguir usam various AWS SDKs e the AWS CLI to call`DetectLabels`. Para obter informações sobre a resposta da operação `DetectLabels`, consulte DetectLabels resposta.

Para detectar rótulos em uma imagem

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs. Certifique-se de ter concedido ao usuário que chama as operações de API as permissões adequadas para o acesso programático; consulte Conceder acesso programático para obter instruções sobre como fazer isso.

2. Faça upload de uma imagem que contenha um ou mais objetos, como árvores, casas e barcos, para seu bucket do S3. A imagem deve estar no formato .jpg ou .png.

   Para obter instruções, consulte Como fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

3. Use os exemplos a seguir para chamar a operação `DetectLabels`.

   Java

   Este exemplo exibe uma lista de rótulos que foram detectados na imagem de entrada. Substitua os valores de `bucket` e `photo` pelos nomes do bucket do Amazon S3 e da imagem usados na etapa 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

Esse exemplo exibe a saída JSON da operação da CLI `detect-labels`. Substitua os valores de `bucket` e `photo` pelos nomes do bucket do Amazon S3 e da imagem usados na etapa 2. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
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 você estiver usando o Windows, talvez seja necessário escapar das aspas, conforme mostrado no exemplo abaixo.

```
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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
//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

Este exemplo exibe os rótulos que foram detectados na imagem de entrada. Substitua os valores de `bucket` e `photo` pelos nomes do bucket do Amazon S3 e da imagem usados na etapa 2. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Este exemplo exibe informações sobre os rótulos detectados em uma imagem.

Altere o valor de `photo` para o caminho e o nome de arquivo de um arquivo de imagem que contenha uma ou mais faces de celebridades. Altere o valor de `bucket` para o nome do bucket S3 que contém o arquivo de imagem fornecido. Altere o valor de `REGION` para o nome da região associada à sua conta. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
// 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

Este exemplo exibe uma lista de rótulos que foram detectados na imagem de entrada. Substitua os valores de `bucket` e `photo` pelos nomes do bucket do Amazon S3 e da imagem usados na etapa 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

Este exemplo exibe uma lista de rótulos que foram detectados na imagem de entrada. Substitua os valores de `bucket` e `photo` pelos nomes do bucket do Amazon S3 e da imagem usados na etapa 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
```

Exemplo de resposta

A resposta de `DetectLabels` é uma matriz de rótulos detectados na imagem e o nível de confiança em que foram detectadas.

Quando você executa a `DetectLabels` operação em uma imagem, o Amazon Rekognition retorna uma saída semelhante ao exemplo de resposta a seguir.

A resposta mostra que a operação detectou vários rótulos, incluindo Pessoa, Veículo e Carro. Cada rótulo tem um nível de confiança associado. Por exemplo, o algoritmo de detecção apresenta 98,991432% de certeza de que a imagem contém uma pessoa.

A resposta também inclui os rótulos ancestrais de um rótulo na matriz `Parents`. Por exemplo, o rótulo Automóvel tem dois rótulos pai chamados Veículo e Transporte.

A resposta para rótulos de objetos comuns inclui as informações da caixa delimitadora para a localização do rótulo na imagem de entrada. Por exemplo, o rótulo Pessoa tem uma matriz de instâncias que contém duas caixas delimitadoras. Essas são as localizações de duas pessoas detectadas na imagem.

O campo `LabelModelVersion` contém o número da versão do modelo de detecção usado por `DetectLabels`.

Para obter mais informações sobre o uso dessa operação `DetectLabels`, consulte Detectando objetos e conceitos.

```
{

    {
    "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"
}


}
```

Analisar uma imagem carregada de um sistema de arquivos local

As operações do Amazon Rekognition Image podem analisar imagens que são fornecidas como bytes de imagem ou imagens armazenadas em um bucket do Amazon S3.

Estes tópicos fornecem exemplos de fornecimento de bytes de imagens para as operações da API Amazon Rekognition Image usando um arquivo carregado de um sistema de arquivos local. Você passa bytes de imagem para uma operação da API Amazon Rekognition usando o parâmetro de entrada Image. Dentro de `Image`, você especifica a propriedade `Bytes` para passar bytes de imagem codificados em base64.

Os bytes de imagem passados para uma operação da API do Amazon Rekognition usando o parâmetro de entrada `Bytes` devem ser codificados em base64. A AWS SDKs que esses exemplos usam imagens codificadas automaticamente em base64. Não é necessário codificar bytes de imagem antes de chamar uma operação da API Amazon Rekognition. Para obter mais informações, consulte Especificações de imagem.

Nesta solicitação JSON de exemplo de `DetectLabels`, os bytes da imagem de origem são passados no parâmetro de entrada `Bytes`.

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

Os exemplos a seguir usam various AWS SDKs e the AWS CLI to call`DetectLabels`. Para obter informações sobre a resposta da operação `DetectLabels`, consulte DetectLabels resposta.

Para obter um JavaScript exemplo do lado do cliente, consulte. Usando JavaScript

Para detectar rótulos em uma imagem local

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use os exemplos a seguir para chamar a operação `DetectLabels`.

   Java

   O exemplo de Java a seguir mostra como carregar uma imagem do sistema de arquivos local e detectar rótulos usando a operação detectLabels do SDK da AWS. Altere o valor de `photo` para o caminho e o nome de arquivo de um arquivo de imagem (formato .jpg ou .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

O exemplo de AWS SDK para Python a seguir mostra como carregar uma imagem do sistema de arquivos local e chamar a operação detect_labels. Altere o valor de `photo` para o caminho e o nome de arquivo de um arquivo de imagem (formato .jpg ou .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

O exemplo a seguir mostra como carregar uma imagem do sistema de arquivos local e detectar rótulos usando a operação `DetectLabels`. Altere o valor de `photo` para o caminho e o nome de arquivo de um arquivo de imagem (formato .jpg ou .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

O exemplo a seguir do AWS SDK for PHP mostra como carregar uma imagem do sistema de arquivos local e chamar DetectFacesa operação da API. Altere o valor de `photo` para o caminho e o nome de arquivo de um arquivo de imagem (formato .jpg ou .png).

```
<?php
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

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

    require 'vendor/autoload.php';

    use Aws\Rekognition\RekognitionClient;

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

    $rekognition = new RekognitionClient($options);

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


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

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

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

```
?>
```

Ruby

Este exemplo exibe uma lista de rótulos que foram detectados na imagem de entrada. Altere o valor de `photo` para o caminho e o nome de arquivo de um arquivo de imagem (formato .jpg ou .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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
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

O exemplo de JavaScript página da web a seguir permite que um usuário escolha uma imagem e visualize a idade estimada dos rostos detectados na imagem. As idades estimativas são retornadas por uma chamada à DetectFaces.

A imagem escolhida é carregada usando a JavaScript `FileReader.readAsDataURL` função, que codifica a imagem em base64. Isso é útil para exibir a imagem em uma tela HTML. Porém, isso significa que os bytes da imagem precisam ser descodificados antes de serem passados para uma operação do Amazon Rekognition Image. Este exemplo mostra como decodificar os bytes da imagem carregada. Se os bytes da imagem codificada não forem úteis para você, use `FileReader.readAsArrayBuffer` no lugar, porque a imagem carregada não está codificada. Isso significa que as operações do Amazon Rekognition Image podem ser chamadas sem primeiro descodificar os bytes da imagem. Para obter um exemplo, consulte Usando o readAsArray Buffer.

Para executar o JavaScript exemplo

1. Carregue o código-fonte de exemplo em um editor.
2. Selecione o banco de identidades de identidade do Amazon Cognito. Para obter mais informações, consulte Selecione o banco de identidades do Amazon Cognito..

3. Na função `AnonLog` do código de exemplo, altere `IdentityPoolIdToUse` e `RegionToUse` para os valores que você anotou na etapa 9 de Selecione o banco de identidades do Amazon Cognito..
4. Na função `DetectFaces`, altere `RegionToUse` para o valor que você usou na etapa anterior.
5. Salve o código fonte de exemplo como um arquivo `.html`.
6. Carregue o arquivo em seu navegador.
7. Escolha o botão Procurar... e escolha uma imagem que contenha uma ou mais faces. Uma tabela é mostrada contendo as idades estimadas de cada face detectada na imagem.

> **Note**
>
> O exemplo de código a seguir usa dois scripts que não fazem mais parte do Amazon Cognito. Para obter esses arquivos, siga os links para aws-cognito-sdk.min.js e .min.js e amazon-cognito-identity salve o texto de cada um como arquivos separados. `.js`

JavaScript código de exemplo

O exemplo de código a seguir usa JavaScript V2. Para ver um exemplo na JavaScript V3, consulte o exemplo no repositório de exemplos GitHub do SDK de AWS documentação.

```
<!--
Copyright 2018 Amazon.com, Inc. or its 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>
```

Usando o readAsArray Buffer

O trecho de código a seguir é uma implementação alternativa da `ProcessImage` função no código de amostra, usando JavaScript V2. Ele usa `readAsArrayBuffer` para carregar uma imagem e chamar `DetectFaces`. Como `readAsArrayBuffer` não codifica o arquivo carregado em base64, não é necessário decodificar os bytes da imagem antes de chamar uma operação do Amazon Rekognition Image.

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

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

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

```
        for (var i = 0; i < data.FaceDetails.length; i++) {
          table += '<tr><td>' + data.FaceDetails[i].AgeRange.Low +
            '</td><td>' + data.FaceDetails[i].AgeRange.High + '</td></tr>';
        }
        table += "</table>";
        document.getElementById("opResult").innerHTML = table;
      }
    });

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

Selecione o banco de identidades do Amazon Cognito.

Para simplificar, o exemplo usa um banco de identidades anônimo do Amazon Cognito para fornecer acesso não autenticado à API Amazon Rekognition Image. Isso pode ser adequado para suas necessidades. Por exemplo, você pode usar o acesso não autenticado para fornecer acesso gratuito ou avaliação gratuita a seu site antes de os usuários se cadastrarem. Para fornecer acesso autenticado, use um grupo de usuários do Amazon Cognito. Para obter mais informações, consulte grupo de usuários do Amazon Cognito.

O procedimento a seguir mostra como criar um grupo de identidades que permite acesso a identidades não autenticadas e como obter o identificador do grupo de identidades que é necessário no código de exemplo.

Para obter o identificador do grupo de identidades

1. Abra o console do Amazon Cognito.
2. Escolha Criar novo banco de identidades.
3. Em Identity pool name* (Nome do grupo de identidades), digite um nome para o grupo de identidades.
4. Em Unauthenticated identities (Identidades não autenticadas), escolha Enable access to unauthenticated identities (Permitir acesso a identidades não autenticadas).
5. Selecione Criar grupo.
6. Escolha View Details (Visualizar detalhes) e anote o nome da função para identidades não autenticadas.
7. Selecione Permitir.

8. Em Plataforma, escolha JavaScript.
9. Em Get AWS Credentials (Obter credenciais da AWS), anote os valores de `AWS.config.region` e `IdentityPooldId` que são mostrados no trecho de código.
10. Abra o console do IAM em https://console.aws.amazon.com/iam/.
11. No painel de navegação, selecione Perfis.
12. Escolha o nome da função que você anotou na etapa 6.
13. Na guia Permissions (Permissões), escolha Attach policies (Anexar políticas).
14. Selecione AmazonRekognitionReadOnlyAccess.
15. Escolha Attach Policy.

Exibir caixas delimitadoras

As operações do Amazon Rekognition Image podem retornar as coordenadas das caixas delimitadoras para os itens que são detectados em imagens. Por exemplo, a operação DetectFaces retornará uma caixa delimitadora (BoundingBox) para cada face detectada em uma imagem. Você pode usar as coordenadas da caixa delimitadora para exibir uma caixa em torno de itens detectados. Por exemplo, a imagem a seguir mostra uma caixa delimitadora em torno de uma face.



Uma `BoundingBox` tem as seguintes propriedades:

- Altura: a altura da caixa delimitadora como uma proporção da altura total da imagem.
- Esquerda: a coordenada esquerda da caixa delimitadora como uma proporção da largura total da imagem.
- Superior: a coordenada superior da caixa delimitadora como uma proporção da altura total da imagem.
- Largura: a largura da caixa delimitadora como uma proporção da largura total da imagem.

Cada BoundingBox propriedade tem um valor entre 0 e 1. Cada valor de propriedade é uma proporção da largura (`Left` e `Width`) ou da altura (`Height` e `Top`) da imagem total. Por exemplo, se a imagem de entrada tiver 700 x 200 pixels e a coordenada superior esquerda da caixa delimitadora tiver 350 x 50 pixels, a API retornará um valor `Left` de 0,5 (350/700) e um valor `Top` de 0,25 (50/200).

O diagrama a seguir mostra a faixa de uma imagem que cada propriedade da caixa delimitadora abrange.

Para exibir a caixa delimitadora com a localização e o tamanho corretos, você precisa multiplicar os BoundingBox valores pela largura ou altura da imagem (dependendo do valor desejado) para obter os valores de pixels. Você pode usar os valores de pixel para exibir a caixa delimitadora. Por exemplo, as dimensões em pixels da imagem anterior são 608 de largura x 588 de altura. Os valores da caixa delimitadora para a face são:

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

A localização da caixa delimitadora da face em pixels é calculada da seguinte forma:

```
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
```

Você pode usar esses valores para exibir uma caixa delimitadora em torno da face.

> **Note**
>
> Uma imagem pode ter várias formas de orientação. Seu aplicativo pode precisar girar a imagem para exibi-la com a orientação correta. As coordenadas da caixa delimitadora são afetadas pela orientação da imagem. Pode ser necessário converter as coordenadas antes que você possa exibir uma caixa delimitadora no local correto. Para obter mais informações, consulte Obter a orientação e as coordenadas da caixa delimitadora da imagem.

Os exemplos a seguir mostram como exibir uma caixa delimitadora em torno de faces que são detectadas chamando DetectFaces. Os exemplos pressupõem que a orientação das imagens é de 0 graus. Os exemplos também mostram como fazer download da imagem de um bucket do Amazon S3.

Para exibir uma caixa delimitadora

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use os exemplos a seguir para chamar a operação `DetectFaces`.

   Java

   Altere o valor de `bucket` para o bucket do Amazon S3 que contém o arquivo de imagem. Altere o valor de `photo` para o nome de arquivo de um arquivo de imagem (formato .jpg ou .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

Altere o valor de `bucket` para o bucket do Amazon S3 que contém o arquivo de imagem. Altere o valor de `photo` para o nome de arquivo de um arquivo de imagem (formato .jpg ou .png). Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

Observe que `s3` se refere ao cliente Amazon S3 do AWS SDK e `rekClient` ao cliente Amazon Rekognition do AWS SDK.

```
//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]
}
```

Obter a orientação e as coordenadas da caixa delimitadora da imagem

Os aplicativos que usam o Amazon Rekognition Image geralmente precisam exibir as imagens detectadas pelas operações do Amazon Rekognition Image e as caixas em torno das faces detectadas. Para exibir uma imagem corretamente em seu aplicativo, é necessário saber a orientação da imagem. Talvez seja necessário corrigir essa orientação. Para alguns arquivos .jpg, a orientação da imagem está contida nos metadados do formato Exif.

Para exibir uma caixa em torno de uma face, você precisa das coordenadas da caixa delimitadora. Se a caixa não estiver orientada corretamente, talvez seja necessário ajustar essas coordenadas. As operações de detecção de faces do Amazon Rekognition Image retornam coordenadas de caixa delimitadora para cada face detectada, mas não estimam coordenadas para arquivos .jpg sem metadados Exif.

Os exemplos a seguir mostram como obter as coordenadas da caixa delimitadora para as faces detectadas em uma imagem.

Use as informações deste exemplo para garantir que suas imagens estejam orientadas corretamente e que as caixas delimitadoras estejam exibidas no local correto em seu aplicativo.

Como o código usado para girar e exibir imagens e caixas delimitadoras depende do idioma e do ambiente que você usa, não explicamos como exibir imagens e caixas delimitadoras em seu código ou como obter informações de orientação de metadados do Exif.

Descobrir a orientação de uma imagem

Para exibir uma imagem corretamente em seu aplicativo, poder ser necessário girá-la. A imagem a seguir está orientada para 0 grau e é exibida corretamente.



No entanto, a imagem a seguir está girada 90 graus no sentido anti-horário. Para exibi-la corretamente, você precisa encontrar a orientação da imagem e usar essas informações em seu código para girar a imagem para 0 grau.



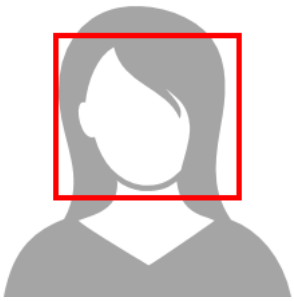
Algumas imagens em formato .jpg contêm informações de orientação nos metadados do Exif. Se disponíveis, os metadados Exif da imagem contêm a orientação. Nos metadados do Exif, você pode encontrar a orientação da imagem no campo `orientation`. Embora o Amazon Rekognition Image identifique a presença de informações da orientação da imagem nos metadados do Exif, ele não fornece acesso a eles. Para acessar os metadados do Exif em uma imagem, use uma biblioteca de terceiros ou escreva seu próprio código. Para obter mais informações, consulte Exif Versão 2.32.

Quando você sabe a orientação de uma imagem, você pode escrever código para girar e exibi-la corretamente.

Exibir caixas delimitadoras

As operações do Amazon Rekognition Image que analisam faces em uma imagem também retornam as coordenadas das caixas delimitadoras que circundam as faces. Para obter mais informações, consulte BoundingBox.

Para exibir uma caixa delimitadora em torno de uma face semelhante à caixa mostrada na imagem a seguir em seu aplicativo, use as coordenadas da caixa delimitadora em seu código. As coordenadas da caixa delimitadora retornadas por uma operação refletem a orientação da imagem. Se for necessário girar a imagem para exibi-la corretamente, você poderá converter as coordenadas da caixa delimitadora.



Exibir caixas delimitadoras quando as informações de orientação estão presentes nos metadados do Exif

Se a orientação de uma imagem estiver incluída nos metadados do Exif, as operações do Amazon Rekognition Image farão o seguinte:

- Retornarão nulo no campo de correção da orientação na resposta da operação. Para girar a imagem, use a orientação fornecida nos metadados do Exif em seu código.
- Retornarão as coordenadas da caixa delimitadora já orientadas para 0 grau. Para mostrar a caixa delimitadora na posição correta, use as coordenadas que foram retornadas. Você não precisa convertê-las.

Exemplo: obter a orientação e as coordenadas da caixa delimitadora de uma imagem

Os exemplos a seguir mostram como usar o AWS SDK para obter os dados Exif de orientação da imagem e as coordenadas da caixa delimitadora para as celebridades detectadas pela operação `RecognizeCelebrities`.

> **Note**
>
> O suporte para estimar a orientação da imagem usando o campo `OrientationCorrection` foi interrompido a partir de agosto de 2021. Todos os valores retornados para esse campo incluídos em uma resposta da API sempre serão NULL.

Java

O exemplo carrega uma imagem do sistema de arquivos local, chama a operação `RecognizeCelebrities`, determina a altura e a largura da imagem e calcula as coordenadas da caixa delimitadora da face para a imagem girada. O exemplo não mostra como processar as informações de orientação que estão armazenadas nos metadados do Exif.

Na função `main`, substitua o valor de `photo` pelo nome e o caminho de uma imagem armazenada localmente no formato .jpg ou .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 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

Este exemplo usa a biblioteca de imagens PIL/Pillow para obter a largura e a altura da imagem. Para obter mais informações, consulte Pillow. Este exemplo preserva metadados do exif que você talvez precise em outros lugares no aplicativo.

Na função `main`, substitua o valor de `photo` pelo nome e o caminho de uma imagem armazenada localmente no formato .jpg ou .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
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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

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

Trabalhar com operações de análise de vídeo armazenado

O Amazon Rekognition Video é uma API que você pode usar para analisar vídeos. Com o Amazon Rekognition Video, você pode detectar rótulos, faces, pessoas, celebridades e conteúdo adulto (sugestivo e explícito) em vídeos armazenados em um bucket do Amazon Simple Storage Service (Amazon S3). Você pode usar o Amazon Rekognition Video em categorias como mídia/entretenimento e segurança pública. Anteriormente, escanear vídeos em busca de objetos ou pessoas exigia muitas horas de visualização propensa a erros por um ser humano. O Amazon Rekognition Video automatiza a detecção de itens e quando eles ocorrem em um vídeo.

Esta seção aborda os tipos de análise que o Amazon Rekognition Video pode realizar, uma visão geral da API e exemplos de uso do Amazon Rekognition Video.

Tópicos

- Tipos de análise
- Visão geral da API Amazon Rekognition Video
- Chamando as operações de vídeo do Amazon Rekognition Video
- Configuração do Amazon Rekognition Video
- Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK)
- Analisando um vídeo com o AWS Command Line Interface
- Referência: Notificação de resultados de análise de vídeo
- Vídeo sobre solução de problemas do Amazon Rekognition

Tipos de análise

Você pode usar o Amazon Rekognition Video para analisar vídeos e obter as seguintes informações:

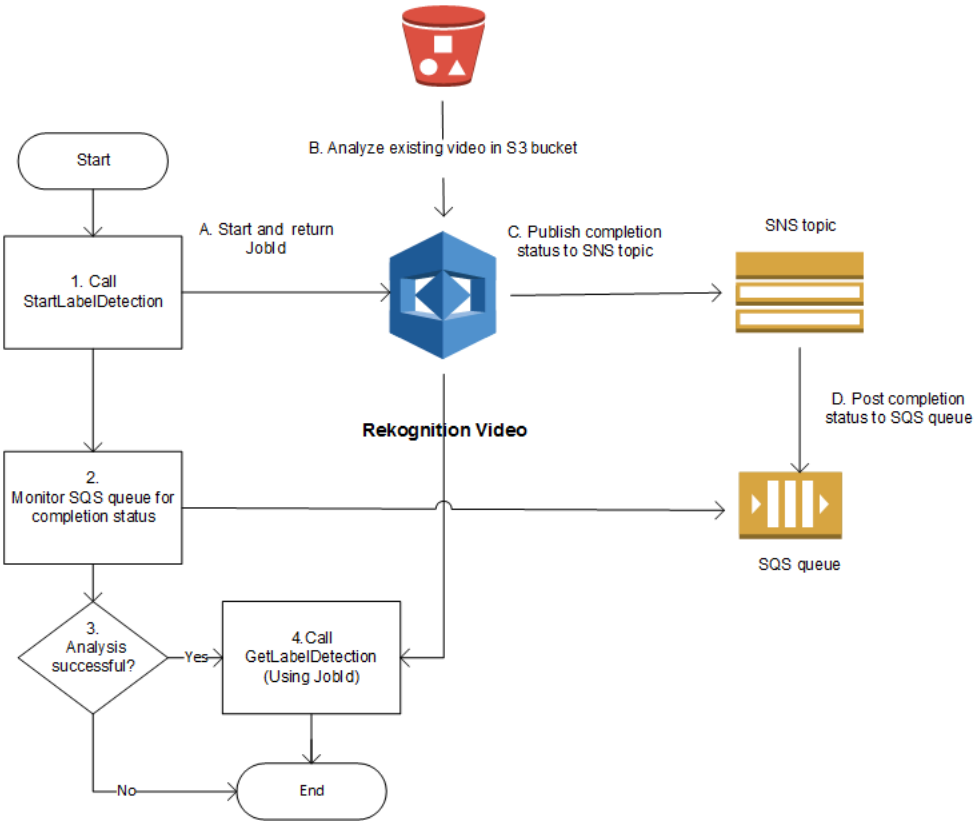
- Segmentos de vídeo
- Rótulos
- Conteúdo adulto sugestivo e explícito
- Texto
- Celebridades
- Faces
- Pessoas

Para obter mais informações, consulte Como o Amazon Rekognition funciona.

Visão geral da API Amazon Rekognition Video

O Amazon Rekognition Video processa um vídeo armazenado em um bucket do Amazon S3. O padrão de design é um conjunto assíncrono de operações. Inicie uma análise de vídeo chamando uma operação `Start`, como a StartLabelDetection. O status de conclusão da solicitação é publicado em um tópico do Amazon Simple Notification Service (Amazon SNS). Para obter o status de conclusão do tópico do Amazon SNS, você pode usar uma fila ou uma função do Amazon Simple Queue Service (Amazon SQS). AWS Lambda Ao receber o status de conclusão, chame uma operação `Get`, como a GetLabelDetection para obter os resultados da solicitação.

O diagrama a seguir mostra o processo de detecção de rótulos em um vídeo armazenado em um bucket do Amazon S3. No diagrama, uma fila do Amazon SQS obtém o status de conclusão do tópico Amazon SNS. Como alternativa, você pode usar uma AWS Lambda função.



O processo é o mesmo para outras operações de vídeo do Amazon Rekognition Video. A tabela a seguir lista as operações `Get` e `Start` para cada uma das operações do Amazon Rekognition que não são de armazenamento.

Detecção	Operação Start	Operação Get
Segmentos de vídeo	StartSegmentDetection	GetSegmentDetection
Rótulos	StartLabelDetection	GetLabelDetection
Conteúdo adulto sugestivo e explícito	StartContentModeration	GetContentModeration
Texto	StartTextDetection	GetTextDetection
Celebridades	StartCelebrityRecognition	GetCelebrityRecognition
Faces	StartFaceDetection	GetFaceDetection
Pessoas	StartPersonTracking	GetPersonTracking

Para outras operações `Get` que não sejam `GetCelebrityRecognition`, o Amazon Rekognition Video retorna informações de rastreamento para quando entidades são detectadas em todo o vídeo de entrada.

Para obter mais informações sobre como usar o Amazon Rekognition Video, consulte Chamando as operações de vídeo do Amazon Rekognition Video. Para ver um exemplo que faz análise de vídeo usando o Amazon SQS, consulte Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK). Para obter AWS CLI exemplos, consulteAnalisando um vídeo com o AWS Command Line Interface.

Formatos e armazenamento de vídeo

As operações do Amazon Rekognition podem analisar vídeos armazenados em buckets do Amazon S3. Para obter uma lista de todos os limites da operação de análise de vídeo, consulte Diretrizes e cotas.

O vídeo deve ser codificado usando o codec H.264. Os formatos de arquivo compatíveis são MPEG-4 e MOV.

Um codec é um software ou hardware que comprime os dados para uma entrega mais rápida e descompacta os dados recebidos em sua forma original. O codec H.264 é comumente usado para gravação, compactação e distribuição de conteúdo de vídeo. O formato de arquivo de vídeo pode conter um ou mais codecs. Se o seu arquivo de vídeo no formato MOV ou MPEG-4 não funcionar com o Amazon Rekognition Video, verifique se o codec usado para codificar o vídeo é H.264.

Qualquer API do Amazon Rekognition Video que analise dados de áudio só oferece suporte a codecs de áudio AAC.

O tamanho máximo do arquivo de um vídeo armazenado é de 10 GB.

Procurando pessoas

Você pode usar metadados faciais que estão armazenados em uma coleção para procurar pessoas em um vídeo. Por exemplo, você pode pesquisar um vídeo arquivado para buscar uma pessoa específica ou várias pessoas. Armazene metadados faciais de imagens de origem em uma coleção usando a operação IndexFaces. Em seguida, você pode usar StartFaceSearch para iniciar uma procura assíncrona de faces na coleção. Use GetFaceSearch para obter os resultados de pesquisa. Para obter mais informações, consulte Pesquisanda faces em vídeos armazenados. Pesquisar pessoas é um exemplo de operação do Amazon Rekognition baseada em armazenamento. Para obter mais informações, consulte Operações de API baseadas em armazenamento.

Você também pode procurar pessoas em um vídeo de streaming. Para obter mais informações, consulte Trabalhando com eventos de streaming de vídeo.

Chamando as operações de vídeo do Amazon Rekognition Video

O Amazon Rekognition Video é uma API assíncrona que você pode usar para analisar vídeos armazenados em um bucket do Amazon Simple Storage Service (Amazon S3). Você inicia a análise de um vídeo chamando uma operação do Amazon `Start` Rekognition Video, como. StartPersonTracking O Amazon Rekognition Video publica o resultado da solicitação de análise em um tópico do Amazon Simple Notification Service (Amazon SNS). Você pode usar uma fila do Amazon Simple Queue Service (Amazon SQS) ou AWS Lambda uma função para obter o status de conclusão da solicitação de análise de vídeo do tópico do Amazon SNS. Por fim, você obtém os resultados da solicitação de análise de vídeo chamando uma operação do `Get` Amazon Rekognition, como. GetPersonTracking

As informações nas seções a seguir usam operações de detecção de rótulos para mostrar como o Amazon Rekognition Video detecta rótulos (objetos, eventos, conceitos e atividades) em um vídeo armazenado em um bucket do Amazon S3. A mesma abordagem funciona para as outras operações do Amazon Rekognition Video, por exemplo, e. StartFaceDetectionStartPersonTracking O exemplo Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK) mostra como analisar um vídeo usando uma fila do Amazon SQS para obter o status de conclusão do tópico Amazon SNS. Ele também é usado como base para outros exemplos de vídeos do Amazon Rekognition, como Pessoas trafegando. Para obter AWS CLI exemplos, consulteAnalisando um vídeo com o AWS Command Line Interface.

Tópicos

- Iniciar a análise de vídeo
- Obter o status de conclusão de uma solicitação de análise do Amazon Rekognition Video
- Obter os resultados da análise do Amazon Rekognition Video

Iniciar a análise de vídeo

Você inicia uma solicitação de detecção de etiquetas do Amazon Rekognition Video ligando. StartLabelDetection Veja a seguir um exemplo de uma solicitação JSON que é transmitida por `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"
}
```

O parâmetro de entrada `Video` fornece o nome do arquivo de vídeo e o bucket do Amazon S3 do qual recuperá-lo. `NotificationChannel` contém o nome de recurso da Amazon (ARN) do tópico do Amazon SNS que o Amazon Rekognition Video notifica quando a solicitação de análise de vídeo é concluída. O tópico do Amazon SNS deve estar na mesma região da AWS do endpoint do Amazon Rekognition Video para o qual você está ligando. `NotificationChannel` também contém o ARN de uma função que permite que o Amazon Rekognition Video seja publicado no tópico do Amazon SNS. Você concede permissões de publicação do Amazon Rekognition aos seus tópicos do Amazon SNS criando um perfil de serviço do IAM. Para obter mais informações, consulte Configuração do Amazon Rekognition Video.

Você também pode especificar um parâmetro de entrada opcional, `JobTag`, que permite identificar o trabalho no status de conclusão que é publicado no tópico do Amazon SNS.

Para evitar a duplicação acidental de trabalhos de análise, você pode opcionalmente fornecer um token de idempotência, `ClientRequestToken`. Se você fornecer um valor para `ClientRequestToken`, a operação `Start` retornará o mesmo `JobId` para várias chamadas idênticas para a operação inicial, como `StartLabelDetection`. Um token `ClientRequestToken` tem uma vida útil de 7 dias. Depois de 7 dias, você pode reutilizá-lo. Se você reutilizar o token durante a vida útil, o seguinte acontecerá:

- Se você reutilizar o token com a mesma operação `Start` e os mesmos parâmetros de entrada, o mesmo `JobId` será retornado. O trabalho não é executado novamente e o Amazon Rekognition Video não envia um status de conclusão para o tópico registrado do Amazon SNS.
- Se você reutilizar o token com a mesma operação `Start` e uma pequena alteração no parâmetro de entrada, você receberá uma exceção `IdempotentParameterMismatchException` (código de status HTTP: 400).

- Você não deve reutilizar um token com operações `Start` diferentes, pois obterá resultados imprevisíveis do Amazon Rekognition.

A resposta à operação `StartLabelDetection` é um identificador de trabalho (`JobId`). Use o `JobId` para rastrear solicitações e obter os resultados da análise após o Amazon Rekognition Video publicar o status de conclusão no tópico do Amazon SNS. Por exemplo:

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

Se você iniciar muitos trabalhos ao mesmo tempo, as chamadas para `StartLabelDetection` gerarão um `LimitExceededException` (código de status HTTP: 400) até que o número de trabalhos em execução simultânea fique abaixo do limite do serviço Amazon Rekognition.

Se você descobrir que exceções `LimitExceededException` são levantadas com picos de atividade, considere usar uma fila do Amazon SQS para gerenciar as solicitações recebidas. Entre em contato com o AWS suporte se você descobrir que seu número médio de solicitações simultâneas não pode ser gerenciado por uma fila do Amazon SQS e você ainda está recebendo exceções. `LimitExceededException`

Obter o status de conclusão de uma solicitação de análise do Amazon Rekognition Video

O Amazon Rekognition Video envia uma notificação de conclusão da análise para o tópico registrado do Amazon SNS. A notificação inclui o identificador do trabalho e o status de conclusão da operação em uma string JSON. Uma solicitação de análise de vídeo bem-sucedida tem o status `SUCCEEDED`. Por exemplo, o resultado a seguir mostra o processamento bem-sucedido de um trabalho de detecção de rótulo.

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

Para obter mais informações, consulte Referência: Notificação de resultados de análise de vídeo.

Para obter as informações de status publicadas no tópico Amazon SNS pelo Amazon Rekognition Video, use uma das seguintes opções:

- AWS Lambda — Você pode assinar uma função AWS Lambda que você escreve em um tópico do Amazon SNS. A função é chamada quando o Amazon Rekognition notifica o tópico do Amazon SNS de que a solicitação foi concluída. Use uma função do Lambda se quiser que o código do servidor processe os resultados de uma solicitação de análise de vídeo. Por exemplo, você pode usar o código do lado do servidor para anotar o vídeo ou criar um relatório sobre o conteúdo do vídeo antes de retornar as informações para um aplicativo cliente. Também recomendamos o processamento do lado do servidor para vídeos grandes, pois a API Amazon Rekognition pode retornar grandes volumes de dados.
- Amazon Simple Queue Service — Você pode inscrever uma fila do Amazon SQS em um tópico do Amazon SNS. Em seguida, você pesquisa a fila do Amazon SQS para recuperar o status de conclusão publicado pelo Amazon Rekognition quando uma solicitação de análise de vídeo é concluída. Para obter mais informações, consulte Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK). Use uma fila do Amazon SQS se quiser chamar as operações de vídeo do Amazon Rekognition Video somente a partir de um aplicativo cliente.

> **Important**
>
> Não recomendamos obter o status de conclusão da solicitação chamando repetidamente a operação Amazon Rekognition Video `Get`. Isso ocorre porque o Amazon Rekognition Video limita a operação `Get` se muitas solicitações forem feitas. Se você estiver processando vários vídeos simultaneamente, é mais simples e eficiente monitorar uma fila SQS para a notificação de conclusão do que pesquisar o Amazon Rekognition Video para saber o status de cada vídeo individualmente.

Obter os resultados da análise do Amazon Rekognition Video

Para obter os resultados de uma solicitação de análise de vídeo, primeiro certifique-se de que o status de conclusão recuperado do tópico do Amazon SNS seja `SUCCEEDED`. Em seguida, chame `GetLabelDetection`, que transmite o valor `JobId` que é retornado de `StartLabelDetection`. O JSON solicitado é semelhante ao exemplo a seguir:

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

JobId é o identificador da operação de análise de vídeo. Como a análise de vídeo pode gerar grandes quantidades de dados, use `MaxResults` para especificar o número máximo de resultados para retornar em uma única operação Get. O valor padrão para `MaxResults` é 1000. Se você especificar um valor maior que 1000, um máximo de 1000 resultados será retornado. Se a operação não retornar todo o conjunto de resultados, um token de paginação para a próxima página será retornado na resposta da operação. Se você tiver um token de paginação de uma solicitação Get anterior, use-o com `NextToken` para obter a próxima página de resultados.

> **Note**
>
> O Amazon Rekognition retém os resultados de uma operação de análise de vídeo por 7 dias. Você não poderá recuperar os resultados da análise após esse período.

O JSON solicitado da operação `GetLabelDetection` é similar ao seguinte:

```
{
    "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
    }
}
```

As operações `GetContentModeration` e `GetLabelDetection` permitem que você classifique os resultados da análise por data e hora ou pelo nome do rótulo. Você também pode agregar resultados por segmento de vídeo ou por data e hora.

Você pode classificar os resultados por tempo de detecção (milissegundos desde o início do vídeo) ou em ordem alfabética pela entidade detectada (objeto, face, celebridade, rótulo de moderação ou pessoa). Para classificar por tempo, defina o valor do parâmetro de entrada `SortBy` para `TIMESTAMP`. Se `SortBy` não for especificado, o comportamento padrão será classificado por tempo. O exemplo anterior é classificado por tempo. Para classificar por entidade, use o parâmetro de entrada `SortBy` com o valor adequado para a operação que você está executando. Por exemplo, para classificar por rótulo detectado em uma chamada para `GetLabelDetection`, use o valor `NAME`.

Para agregar resultados por data e hora, defina o valor do parâmetro `AggregateBy` como `TIMESTAMPS`. Para agregar por segmento de vídeo, defina o valor de `AggregateBy` para `SEGMENTS`. `SEGMENTS` o modo de agregação agregará os rótulos ao longo do tempo, ao mesmo tempo em que `TIMESTAMPS` fornece ao carimbo de data/hora em que um rótulo foi detectado, usando amostragem de 2 FPS e saída por quadro (Nota: Essa taxa de amostragem atual está sujeita a alterações, não devem ser feitas suposições sobre a taxa de amostragem atual). Se nenhum valor for especificado, o método de agregação padrão será `TIMESTAMPS`.

Configuração do Amazon Rekognition Video

Para usar a API Amazon Rekognition Video com vídeos armazenados, você precisa configurar o usuário e um perfil de serviço do IAM para acessar seus tópicos do Amazon SNS. Você também precisa inscrever uma fila do Amazon SQS para seus tópicos do Amazon SNS.

> **Note**
>
> Se estiver usando essas instruções para configurar o exemplo Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK), não precisará executar as etapas 3, 4, 5 e 6. O exemplo inclui código para criar e configurar o tópico do Amazon SNS e a fila do Amazon SQS.

Os exemplos nesta seção criam um novo tópico do Amazon SNS usando as instruções que dão ao Amazon Rekognition Video acesso a vários tópicos. Se você quiser usar um tópico existente do Amazon SNS, use Conceder acesso a um tópico existente do Amazon SNS para a etapa 3.

Configurar o Amazon Rekognition Video

1. Configure uma AWS conta para acessar o Amazon Rekognition Video. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.
2. Instale e configure o AWS SDK necessário. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.
3. Para executar os exemplos de código neste guia do desenvolvedor, certifique-se de que o usuário escolhido tenha acesso programático. Consulte Conceder acesso programático para obter mais informações.

   Seu usuário também precisa de pelo menos as seguintes permissões:

   - Amazon SQSFull Access
   - AmazonRekognitionFullAccess
   - Amazon S3 FullAccess
   - Amazon SNSFull Access

Se você estiver usando o IAM Identity Center para se autenticar, adicione as permissões ao conjunto de permissões do seu perfil, caso contrário, adicione as permissões ao seu perfil do IAM.

4. Crie um tópico do Amazon SNS usando o console do Amazon SNS. Prefixe o nome do tópico com. AmazonRekognition Anote o nome do recurso da Amazon (ARN) do tópico. Verifique se o tópico está na mesma região do endpoint da AWS que você está usando.
5. Crie uma fila padrão do Amazon SQS usando o console do Amazon SQS. Anotar o ARN da fila.
6. Inscreva a fila no tópico criado na etapa 3.
7. Dê permissão ao tópico do Amazon SNS para enviar mensagens para a fila do Amazon SQS.
8. Crie um perfil de serviço do IAM para dar ao Amazon Rekognition Video acesso aos seus tópicos do Amazon SNS. Observe o nome de recurso da Amazon (ARN) da função de serviço. Para obter mais informações, consulte Conceder acesso a vários tópicos do Amazon SNS.
9. Para garantir que sua conta esteja segura, você deve limitar o escopo do acesso do Rekognition apenas aos recursos que você está usando. Isso pode ser feito anexando uma política de confiança à suo perfil de serviço do IAM. Para obter informações sobre como fazer isso, consulte Prevenção do problema do substituto confuso entre serviços.
10. Adicione a seguinte política em linha ao usuário que você criou na etapa 1:

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

Dê um nome de sua escolha à política em linha.

11. Se você usar uma AWS Key Management Service chave gerenciada pelo cliente para criptografar os vídeos em seu bucket do Amazon S3, adicione permissões à chave que permitam que a função de serviço que você criou na etapa 7 decodifique os vídeos. No

mínimo, o perfil de serviço precisa de permissão para as ações `kms:GenerateDataKey` e `kms:Decrypt`. Por exemplo:

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

Para obter mais informações, consulte Meu bucket Amazon S3 tem criptografia padrão usando uma chave personalizada do AWS KMS. Como posso permitir que os usuários façam download e upload do bucket e Protegendo dados usando criptografia do lado do servidor com chaves KMS armazenadas no AWS Key Management Service (SSE-KMS).

12. Agora, você pode executar os exemplos em Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK) e Analisando um vídeo com o AWS Command Line Interface.

Conceder acesso a vários tópicos do Amazon SNS

Você usa um perfil de serviço do IAM para dar ao Amazon Rekognition Video acesse aos tópicos do Amazon SNS que você cria. O IAM fornece o caso de uso do Rekognition para criar um perfil de serviço Amazon Rekognition Video.

Você pode dar ao Amazon Rekognition Video acesso a vários tópicos do Amazon SNS usando a política de permissões e prefixando os nomes dos tópicos com `AmazonRekognitionServiceRole` —por exemplo,. AmazonRekognition`AmazonRekognitionMyTopicName`

Para dar ao Amazon Rekognition Video acesso a vários tópicos do Amazon SNS

1. Crie um perfil de serviço do IAM. Use as informações a seguir para criar o perfil de serviço do IAM:

    1. Escolha Rekognition como o nome de serviço.

2. Escolha Rekognition para o caso de uso da função de serviço. Você deve ver a política de AmazonRekognitionServiceRolepermissões listada. AmazonRekognitionServiceRoledá ao Amazon Rekognition Video acesso aos tópicos do Amazon SNS prefixados com. AmazonRekognition
3. Dê um nome de sua escolha à perfil de serviço.

2. Anote o ARN da função de serviço. Ele é necessário para iniciar as operações de análise de vídeo.

Conceder acesso a um tópico existente do Amazon SNS

Você pode criar uma política de permissões que permita ao Amazon Rekognition Video acessar um tópico existente do Amazon SNS.

Conceder acesso ao Amazon Rekognition Video a um tópico existente do Amazon SNS

1. Crie uma nova política de permissões com o editor de políticas IAM JSON e use a política a seguir. Substitua `topicarn` pelo nome de recurso da Amazon (ARN) do tópico desejado do Amazon SNS.

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

2. Crie um perfil de serviço do IAM ou atualize um perfil de serviço do IAM existente. Use as informações a seguir para criar o perfil de serviço do IAM:
    1. Escolha Rekognition como o nome de serviço.
    2. Escolha Rekognition para o caso de uso da função de serviço.
    3. Anexe a política de permissões que você criou na etapa 1.

3. Anote o ARN da função de serviço. Ele é necessário para iniciar as operações de análise de vídeo.

Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK)

Esse procedimento mostra como detectar rótulos em um vídeo usando as operações de detecção de rótulos do Amazon Rekognition Video, um vídeo armazenado em um bucket do Amazon S3 e um tópico do Amazon SNS. O procedimento também mostra como usar uma fila do Amazon SQS para obter o status de conclusão do tópico Amazon SNS. Para obter mais informações, consulte Chamando as operações de vídeo do Amazon Rekognition Video. Você não está restrito a usar uma fila do Amazon SQS. Por exemplo, você pode usar uma AWS Lambda função para obter o status de conclusão. Para obter mais informações, consulte Chamar funções do Lambda usando notificações do Amazon SNS.

O código de exemplo no procedimento mostra como fazer o seguinte:

1. Crie o tópico do Amazon SNS.
2. Crie a fila do Amazon SQS.
3. Dê permissão ao Amazon Rekognition Video para publicar o status de conclusão de uma operação de análise de vídeo no tópico Amazon SNS.
4. Inscreva a fila do Amazon SQS no tópico Amazon SNS.
5. Iniciar uma solicitação de análise de vídeo chamando StartLabelDetection.
6. Obtenha o status de conclusão na fila do Amazon SQS. O exemplo rastreia o identificador do trabalho (`JobId`) que é retornado em `StartLabelDetection` e somente obtém os resultados para identificadores de trabalho correspondentes que são lidos a partir do status de conclusão. Essa é uma consideração importante se outros aplicativos estiverem usando a mesma fila e tópico. Para simplificar, o exemplo exclui trabalhos não correspondentes. Considere adicioná-los a uma fila de mensagens mortas do Amazon SQS para uma investigação mais aprofundada.
7. Obter e exibir os resultados da análise de vídeo chamando GetLabelDetection.

Pré-requisitos

O código de exemplo para este procedimento é fornecido em Java e Python. Você precisa ter o AWS SDK apropriado instalado. Para obter mais informações, consulte Comece a usar o Amazon

Rekognition. A conta da AWS que você usa deve ter permissões de acesso à API do Amazon Rekognition. Para obter mais informações, consulte Ações definidas pelo Amazon Rekognition.

Para detectar rótulos em um vídeo

1. Configure o acesso do usuário ao Amazon Rekognition Video e configure o acesso do Amazon Rekognition Video ao Amazon SNS. Para obter mais informações, consulte Configuração do Amazon Rekognition Video. Não é necessário executar as etapas 3, 4, 5 e 6 porque o código de exemplo cria e configura o tópico do Amazon SNS e a fila do Amazon SQS.
2. Faça upload de um arquivo de vídeo no formato MOV ou MPEG-4 para um bucket do Amazon S3. Para fins de teste, faça upload de um vídeo com até 30 segundos de duração.

   Para obter instruções, consulte Como fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

3. Use os exemplos de código a seguir para detectar rótulos em um vídeo.

   Java

   Na função `main`:

   - Substitua `roleArn` com o ARN do perfil de serviço do IAM que você criou na etapa 7 de Configurar o Amazon Rekognition Video.
   - Substitua os valores de `amzn-s3-demo-bucket` e `video` pelo nome do bucket e do arquivo de vídeo especificados por você na etapa 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

Na função `main`:

- Substitua `roleArn` com o ARN do perfil de serviço do IAM que você criou na etapa 7 de Configurar o Amazon Rekognition Video.
- Substitua os valores de `amzn-s3-demo-bucket` e `video` pelo nome do bucket e do arquivo de vídeo especificados por você na etapa 2.

- Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.
- Você também pode incluir critérios de filtragem no parâmetro de configurações. Por exemplo, você pode usar o `LabelsInclusionFilter` ou o `LabelsExclusionFilter` com uma lista de valores desejados. No código abaixo, você pode descomentar a seção `Features` e `Settings` e fornecer seus próprios valores para limitar os resultados retornados apenas aos rótulos de seu interesse.
- Na chamada para `GetLabelDetection`, você pode fornecer valores para os argumentos `AggregateBy` e `SortBy`. Para classificar por tempo, defina o valor do parâmetro de entrada `SortBy` para `TIMESTAMP`. Para classificar por entidade, use o parâmetro de entrada `SortBy` com o valor adequado para a operação que você está executando. Para agregar resultados por data e hora, defina o valor do parâmetro `AggregateBy` como `TIMESTAMPS`. Para agregar por segmento de vídeo, use `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

No código de exemplo a seguir:

- Substitua o valor de `REGION` pelo nome da região operacional da sua conta.
- Substitua o valor de `amzn-s3-demo-bucket` pelo nome do bucket do Amazon S3 que contém seu arquivo de vídeo.
- Substitua o valor de `videoName` pelo nome do arquivo de vídeo em seu bucket do Amazon S3.
- Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.
- Substitua `roleArn` com o ARN do perfil de serviço do IAM que você criou na etapa 7 de Configurar o Amazon Rekognition Video.

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

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

// Set bucket and video variables
```

```
const bucket = "bucket-name";
const videoName = "video-name";
const roleArn = "role-arn"
var startJobId = ""

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

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

const createTopicandQueue = async () => {
  try {
    // Create SNS topic
    const topicResponse = await snsClient.send(new
 CreateTopicCommand(snsTopicParams));
    const topicArn = topicResponse.TopicArn
    console.log("Success", topicResponse);
    // Create SQS Queue
    const sqsResponse = await sqsClient.send(new
 CreateQueueCommand(sqsParams));
    console.log("Success", sqsResponse);
    const sqsQueueCommand = await sqsClient.send(new
 GetQueueUrlCommand({QueueName: sqsQueueName}))
    const sqsQueueUrl = sqsQueueCommand.QueueUrl
    const attribsResponse = await sqsClient.send(new
 GetQueueAttributesCommand({QueueUrl: sqsQueueUrl, AttributeNames:
 ['QueueArn']}))
    const attribs = attribsResponse.Attributes
    console.log(attribs)
    const queueArn = attribs.QueueArn
    // subscribe SQS queue to SNS topic
    const subscribed = await snsClient.send(new SubscribeCommand({TopicArn:
 topicArn, Protocol:'sqs', Endpoint: queueArn}))
    const policy = {
      Version: "2012-10-17",
```

```
      Statement: [
        {
          Sid: "MyPolicy",
          Effect: "Allow",
          Principal: {AWS: "*"},
          Action: "SQS:SendMessage",
          Resource: queueArn,
          Condition: {
            ArnEquals: {
              'aws:SourceArn': topicArn
            }
          }
        }
      ]
    };

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

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

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

const getLabelDetectionResults = async(startJobId) => {
  console.log("Retrieving Label Detection results")
```

```
 // Set max results, paginationToken and finished will be updated depending on
response values
 var maxResults = 10
 var paginationToken = ''
 var finished = false

 // Begin retrieving label detection results
 while (finished == false){
   var response = await rekClient.send(new GetLabelDetectionCommand({JobId:
startJobId, MaxResults: maxResults,
     NextToken: paginationToken, SortBy:'TIMESTAMP'}))
     // Log metadata
     console.log(`Codec: ${response.VideoMetadata.Codec}`)
     console.log(`Duration: ${response.VideoMetadata.DurationMillis}`)
     console.log(`Format: ${response.VideoMetadata.Format}`)
     console.log(`Frame Rate: ${response.VideoMetadata.FrameRate}`)
     console.log()
     // For every detected label, log label, confidence, bounding box, and
timestamp
     response.Labels.forEach(labelDetection => {
       var label = labelDetection.Label
       console.log(`Timestamp: ${labelDetection.Timestamp}`)
       console.log(`Label: ${label.Name}`)
       console.log(`Confidence: ${label.Confidence}`)
       console.log("Instances:")
       label.Instances.forEach(instance =>{
         console.log(`Confidence: ${instance.Confidence}`)
         console.log("Bounding Box:")
         console.log(`Top: ${instance.Confidence}`)
         console.log(`Left: ${instance.Confidence}`)
         console.log(`Width: ${instance.Confidence}`)
         console.log(`Height: ${instance.Confidence}`)
         console.log()
       })
     console.log()
     // Log parent if found
     console.log("   Parents:")
     label.Parents.forEach(parent =>{
       console.log(`    ${parent.Name}`)
     })
     console.log()
     // 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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
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. Compile e execute o código. A operação pode demorar um pouco para ser concluída. Depois de ser concluída, uma lista de rótulos detectados no vídeo é exibida. Para obter mais informações, consulte Detectando rótulos em um vídeo.

Analisando um vídeo com o AWS Command Line Interface

Você pode usar o AWS Command Line Interface (AWS CLI) para chamar as operações do Amazon Rekognition Video. O padrão de design é o mesmo de usar a API Amazon Rekognition Video com a AWS ou outra. AWS SDK para Java SDKs Para obter mais informações, consulte Visão geral da API Amazon Rekognition Video. Os procedimentos a seguir mostram como usar o AWS CLI para detectar rótulos em um vídeo.

Comece a detectar rótulos em um vídeo chamando `start-label-detection`. Quando o Amazon Rekognition termina de analisar o vídeo, o status de conclusão é enviado para o tópico do Amazon SNS especificado no parâmetro `--notification-channel` de `start-label-detection`. Você pode obter o status de conclusão inscrevendo uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico Amazon SNS. Em seguida, faça uma sondagem em receive-message para obter o status de conclusão da fila do Amazon SQS.

Ao chamar `StartLabelDetection`, você pode filtrar seus resultados fornecendo argumentos de filtragem para os argumentos `LabelsExclusionFilter` e/ou `LabelsInclusionFilter`. Para obter mais informações, consulte Detectando rótulos em um vídeo.

A notificação do status de conclusão é uma estrutura JSON dentro da resposta `receive-message`. Você precisa extrair o JSON da resposta. Para obter informações sobre o JSON do status de conclusão, consulte Referência: Notificação de resultados de análise de vídeo. Se o valor do campo `Status` do JSON do status de conclusão for `SUCCEEDED`, você poderá obter os resultados da solicitação de análise de vídeo ao chamar `get-label-detection`. Ao chamar `GetLabelDetection`, você pode classificar e agregar os resultados retornados usando os argumentos `AggregateBy` e `SortBy`.

Os procedimentos a seguir não incluem código para pesquisar a fila do Amazon SQS. Além disso, eles não incluem código para analisar o JSON retornado da fila do Amazon SQS. Para obter um exemplo em Java, consulte Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK).

Pré-requisitos

Para executar esse procedimento, você precisa ter o AWS CLI instalado. Para obter mais informações, consulte Comece a usar o Amazon Rekognition. A conta da AWS que você usa deve ter permissões de acesso à API do Amazon Rekognition. Para obter mais informações, acesse Ações definidas pelo Amazon Rekognition.

Para configurar o Amazon Rekognition Video e fazer o upload de um vídeo

1. Configure o acesso do usuário ao Amazon Rekognition Video e configure o acesso do Amazon Rekognition Video ao Amazon SNS. Para obter mais informações, consulte Configuração do Amazon Rekognition Video.
2. Faça upload de um arquivo de vídeo no formato MOV ou MPEG-4 no bucket do S3. Ao desenvolver e testar, sugerimos o uso de vídeos curtos com até 30 segundos de duração.

   Para obter instruções, consulte Como fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

Para detectar rótulos em um vídeo

1. Execute o AWS CLI comando a seguir para começar a detectar rótulos em um vídeo.

```
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"]}}
```

   Atualize os seguintes valores:

   - Mude `amzn-s3-demo-bucket` e `videofile` para o nome do bucket do Amazon S3 e o nome do arquivo que você especificou na etapa 2.
   - Altere `us-east-1` para a região da AWS que você está usando.
   - Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

- Mude `TopicARN` para o ARN do tópico do Amazon SNS que você criou na etapa 3 do Configuração do Amazon Rekognition Video.
- Mude `RoleARN` para o ARN do perfil de serviço do IAM que você criou na etapa 7 do Configuração do Amazon Rekognition Video.
- Se necessário, você pode especificar o `endpoint-url`. A CLI da AWS deve determinar automaticamente a URL correta do endpoint com base na região fornecida. No entanto, se você estiver usando um endpoint da sua VPC privada, talvez seja necessário especificar o `endpoint-url`. O recurso AWS Service Endpoints lista a sintaxe para especificar URLs de endpoints e os nomes e códigos de cada região.
- Você também pode incluir critérios de filtragem no parâmetro de configurações. Por exemplo, você pode usar o `LabelsInclusionFilter` ou o `LabelsExclusionFilter` com uma lista de valores desejados.

Se você estiver acessando a CLI em um dispositivo Windows, use aspas duplas em vez de aspas simples e escape das aspas duplas internas com barra invertida (ou seja, \) para resolver quaisquer erros de analisador que você possa encontrar. Para ver um exemplo, veja abaixo:

```
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. Anote o valor do `JobId` na resposta. A resposta é semelhante ao seguinte exemplo JSON.

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

3. Escreva um código para pesquisar na fila do Amazon SQS o status de conclusão JSON (usando receive-message).
4. Escreva o código para extrair o campo `Status` do JSON do status de conclusão.
5. Se o valor de `Status` for`SUCCEEDED`, execute o AWS CLI comando a seguir para mostrar os resultados da detecção do rótulo.

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

Atualize os seguintes valores:

- Altere o `JobId` para que corresponda ao identificador de trabalho que você anotou na etapa 2.
- Altere `Endpoint` e `us-east-1` para o endpoint e a região da AWS que você está usando.

Os resultados são semelhantes ao seguinte exemplo JSON:

```
{
    "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"
            }
```

```
        }, .......
```

Referência: Notificação de resultados de análise de vídeo

O Amazon Rekognition publica os resultados de uma solicitação de análise do Amazon Rekognition Video, incluindo o status de conclusão, em um tópico do Amazon Simple Notification Service (Amazon SNS). Para receber a notificação de um tópico do Amazon SNS, use uma fila ou uma função do Amazon Simple Queue Service. AWS Lambda Para obter mais informações, consulte the section called “Chamando as operações de vídeo do Amazon Rekognition Video”. Para obter um exemplo, consulte Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK).

A carga está no seguinte formato JSON:

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

Nome	Descrição
JobId	O identificador do trabalho. Corresponde a um identificador de trabalho que é retornado de uma operação `Start`, como StartPersonTracking.
Status	O status do trabalho. Os valores válidos são SUCCEEDED, FAILED ou ERROR.
API	A operação Amazon Rekognition Video usada para analisar o vídeo de entrada.

Nome	Descrição
JobTag	Identificador para o trabalho. Especifique `JobTag` em uma chamada para a operação Start, como StartLabelDetection.
Timestamp	O time stamp Unix de quando o trabalho foi concluído.
Vídeo	Detalhes sobre o vídeo que foi processado. Inclui o nome do arquivo e o bucket do Amazon S3 em que o arquivo está armazenado.

Veja a seguir um exemplo de uma notificação bem-sucedida enviada para um tópico do 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"
  }
}
```

Vídeo sobre solução de problemas do Amazon Rekognition

Veja a seguir informações de solução de problemas para trabalhar com o Amazon Rekognition Video e vídeos armazenados.

Nunca recebo o status de conclusão enviado para o tópico do Amazon SNS

O Amazon Rekognition Video publica informações de status em um tópico do Amazon SNS quando a análise do vídeo é concluída. Normalmente, você recebe a mensagem de status de conclusão ao se inscrever no tópico com uma fila do Amazon SQS ou uma função do Lambda. Para ajudar na sua investigação, assine o tópico do Amazon SNS por e-mail para receber as mensagens enviadas para

o tópico do Amazon SNS em sua caixa de entrada de e-mail. Para obter mais informações, consulte Assinatura de um tópico do Amazon SNS.

Se você não receber a mensagem em seu aplicativo, considere o seguinte:

- Verifique se a análise foi concluída. Verifique o valor de `JobStatus` na resposta da operação Get (`GetLabelDetection`, por exemplo). Se o valor for `IN_PROGRESS`, a análise não foi concluída e o status de conclusão ainda não foi publicado no tópico do Amazon SNS.
- Verifique se você tem um perfil de serviço do IAM que dá ao Amazon Rekognition Video permissões para publicar em seus tópicos do Amazon SNS. Para obter mais informações, consulte Configuração do Amazon Rekognition Video.
- Confirme se o perfil de serviço do IAM que você está usando pode ser publicada no tópico do Amazon SNS usando credenciais de função e se as permissões da suo perfil de serviço têm como escopo seguro os recursos que você está usando. Execute as seguintes etapas:
  - Obtenha o nome de recurso da Amazon (ARN) do usuário:

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

  - Adicione o ARN do usuário à relação de confiança da função. Para obter mais informações, consulte Modificar uma função. O exemplo de política de confiança a seguir especifica as credenciais da função do usuário e restringe as permissões do perfil de serviço apenas aos recursos que você está usando (para obter mais informações sobre como limitar com segurança o escopo das permissões de um perfil de serviço, consulte Prevenção do problema do substituto confuso entre serviços):

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

- Assuma a função: `aws sts assume-role --role-arn arn:Role ARN --role-session-name SessionName --profile RekognitionUser`
- Publique no tópico do Amazon SNS: `aws sns publish --topic-arn arn:Topic ARN --message "Hello World!" --region us-east-1 --profile RekognitionUser`

Se o comando AWS CLI funcionar, você receberá a mensagem (na sua caixa de entrada de e-mail, caso tenha se inscrito no tópico por e-mail). Se você não receber a mensagem:

- Verifique se você configurou o Amazon Rekognition Video. Para obter mais informações, consulte Configuração do Amazon Rekognition Video.
- Consulte as outras dicas de resolução de problemas para essa questão.

- Verifique se você está usando o tópico correto do Amazon SNS:
  - Se você usar um perfil de serviço IAM para conceder ao Amazon Rekognition Video acesso a um único tópico do Amazon SNS, verifique se concedeu permissões ao tópico correto do Amazon SNS. Para obter mais informações, consulte Conceder acesso a um tópico existente do Amazon SNS.
  - Se você usa uma função de serviço do IAM para dar ao Amazon Rekognition Video acesso a vários tópicos do SNS, verifique se você está usando o tópico correto e se o nome do tópico está prefixado com. AmazonRekognition Para obter mais informações, consulte Conceder acesso a vários tópicos do Amazon SNS.
  - Se você usa uma AWS Lambda função, confirme se sua função Lambda está inscrita no tópico correto do Amazon SNS. Para obter mais informações, consulte Fanout para funções do Lambda.
- Se você inscrever uma fila do Amazon SQS em seu tópico do Amazon SNS, confirme se seu tópico do Amazon SNS tem permissões para enviar mensagens para a fila do Amazon SQS. Para obter mais informações, consulte Dar permissão ao tópico do Amazon SNS para enviar mensagens para a fila do Amazon SQS.

Preciso de ajuda adicional para solucionar problemas no tópico do Amazon SNS

Você pode usar AWS X-Ray com o Amazon SNS para rastrear e analisar as mensagens que trafegam pelo seu aplicativo. Para obter mais informações, consulte Amazon SNS e. AWS X-Ray

Para obter ajuda adicional, você pode postar sua pergunta no fórum do Amazon Rekognition ou considerar a possibilidade de se inscrever no suporte técnico da AWS.

Trabalhando com eventos de streaming de vídeo

Você pode usar o Amazon Rekognition Video para detectar e reconhecer faces ou detectar objetos em streaming de vídeo. O Amazon Rekognition Video usa o Amazon Kinesis Video Streams para receber e processar um fluxo de vídeo. Você cria um processador de stream com parâmetros que mostram o que você deseja que o processador de stream detecte no stream de vídeo. O Rekognition envia resultados de detecção de rótulos de eventos de streaming de vídeo como notificações do Amazon SNS e do Amazon S3. O Rekognition exibe os resultados da pesquisa facial em um fluxo de dados do Kinesis.

Os processadores de fluxo de pesquisa facial são usados no `FaceSearchSettings` para pesquisar faces de uma coleção. Para obter mais informações sobre como implementar processadores de stream de pesquisa facial para analisar faces em streaming de vídeo, consulte the section called “Pesquisando faces em uma coleção em um vídeo de streaming”.

Os processadores de stream de detecção de etiquetas usam `ConnectedHomeSettings` para pesquisar pessoas, pacotes e animais de estimação em eventos de streaming de vídeo. Para obter mais informações sobre como implementar processadores de fluxo de detecção de etiquetas, consulte the section called “Detectar rótulos em eventos de streaming de vídeo”.

Visão geral das operações do processador de stream do Amazon Rekognition Video

Você começa a analisar um streaming de vídeo iniciando um processador de streaming de vídeo Amazon Rekognition e transmitindo vídeo para o Amazon Rekognition Video. Um processador de stream do Amazon Rekognition Video permite que você inicie, interrompa e gerencie processadores de stream. Crie um processador de fluxo chamando CreateStreamProcessor. Os parâmetros de solicitação para criar um processador de stream de busca facial incluem os Amazon Resource Names (ARNs) para o stream de vídeo do Kinesis, o stream de dados do Kinesis e o identificador da coleção usada para reconhecer rostos no streaming de vídeo. Os parâmetros de solicitação para

criar um processador de stream de monitoramento de segurança incluem os Amazon Resource Names (ARNs) para o stream de vídeo do Kinesis e o tópico do Amazon SNS, os tipos de objetos que você deseja detectar no stream de vídeo e as informações de um bucket do Amazon S3 para os resultados de saída. Você também inclui um nome que você especifica para o processador de stream.

Comece o processamento de um vídeo chamando a operação StartStreamProcessor. Para obter informações de status para um processador de fluxo, chame DescribeStreamProcessor. Outras operações que você pode chamar são TagResourcemarcar um processador de fluxo e DeleteStreamProcessorexcluir um processador de fluxo. Se você estiver usando um processador de stream de busca facial, você também pode usar StopStreamProcessorpara parar um processador de stream. Para obter uma lista de processadores de fluxo em sua conta, chame ListStreamProcessors.

Depois que o processador de stream começar a funcionar, você transmite o vídeo para o Amazon Rekognition Video por meio do stream de vídeo do Kinesis que você especificou `CreateStreamProcessor`. Você pode usar a operação do SDK do Kinesis Video PutMediaStreams para entregar vídeo ao stream de vídeo do Kinesis. Para ver um exemplo, consulte Exemplo de PutMedia API.

Para obter informações sobre como seu aplicativo pode consumir os resultados da análise do Amazon Rekognition Video a partir de um processador de stream de pesquisa facial, consulte Lendo os resultados da análise de streaming de vídeo.

Adicionar tags a um novo processador de stream

Você pode identificar, organizar, pesquisar e filtrar os processadores de stream do Amazon Rekognition usando tags. Cada tag é um rótulo que consiste em um valor e uma chave definida pelo usuário.

Você pode adicionar tags a um processador de stream ao criá-lo usando a operação `CreateStreamProcessor`. Especifique uma ou mais tags no parâmetro de entrada da matriz `Tags`. Veja a seguir um exemplo de JSON para a solicitação `CreateStreamProcessor` com tags.

```
{
       "Name": "streamProcessorForCam",
       "Input": {
              "KinesisVideoStream": {
                     "Arn": "arn:aws:kinesisvideo:us-east-1:nnnnnnnnnnnn:stream/
inputVideo"
```

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

Adicionar tags a um processador de stream existente

Você pode identificar, organizar, pesquisar e filtrar os processadores de stream do Amazon Rekognition usando tags. Cada tag é um rótulo que consiste em um valor e uma chave definida pelo usuário.

Para adicionar uma ou mais tags a um processador de fluxo existente, use a operação `TagResource`. Especifique o Amazon Resource Name (ARN) do processador de fluxo (`ResourceArn`) e as tags (`Tags`) que você deseja adicionar. O exemplo a seguir mostra como adicionar duas tags.

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

> **Note**
>
> Se você não souber o Amazon Resource Name do processador de fluxo, poderá usar a operação `DescribeStreamProcessor`.

Listar tags em um processador de stream

Você pode identificar, organizar, pesquisar e filtrar os processadores de stream do Amazon Rekognition usando tags. Cada tag é um rótulo que consiste em um valor e uma chave definida pelo usuário.

Para listar as tags anexadas a um processador de stream, use a operação `ListTagsForResource` e especifique o ARN do stream processor (`ResourceArn`). A resposta é um mapa de chaves e valores de tags que são anexados ao processador de fluxo especificado.

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

A saída exibe uma lista de tags anexadas ao processador de stream:

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

Excluir tags de um processador de stream

Você pode identificar, organizar, pesquisar e filtrar os processadores de stream do Amazon Rekognition usando tags. Cada tag é um rótulo que consiste em um valor e uma chave definida pelo usuário.

Para remover uma ou mais tags de um processador de stream, use a operação `UntagResource`. Especifique o ARN do modelo (`ResourceArn`) e das chaves de tag (`Tag-Keys`) que você deseja remover.

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

Como alternativa, você pode especificar chaves de tag neste formato:

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

Gerenciamento de erros

Esta seção descreve erros de runtime e como lidar com eles. Ele também descreve mensagens de erro e códigos específicos do Amazon Rekognition.

Tópicos

- Componentes de erros
- Mensagens e códigos de erro
- Tratamento de erros na aplicação

Componentes de erros

Quando seu programa envia uma solicitação, o Amazon Rekognition tenta processá-la. Se a solicitação for bem-sucedida, o Amazon Rekognition retornará um código HTTP de status de sucesso (`200 OK`), junto com os resultados da operação solicitada.

Se a solicitação não for bem-sucedida, o Amazon Rekognition retornará um erro. Cada erro tem três componentes:

- Um código de status HTTP (como `400`).
- Um nome de exceção (como `InvalidS3ObjectException`).
- Uma mensagem de erro (como `Unable to get object metadata from S3. Check object key, region and/or access permissions.`).

A AWS SDKs cuida da propagação de erros para seu aplicativo, para que você possa tomar as medidas apropriadas. Por exemplo, em um programa Java, você pode escrever a lógica `try-catch` para lidar com um `ResourceNotFoundException`.

Se você não estiver usando um SDK da AWS, precisará analisar o conteúdo da resposta de baixo nível do Amazon Rekognition. Veja a seguir um exemplo dessa resposta:

```
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."}
```

Mensagens e códigos de erro

A seguir está uma lista de exceções que o Amazon Rekognition retorna, agrupadas por código de status HTTP. Se OK para tentar novamente? for Sim, você poderá enviar a mesma solicitação novamente. Se OK to retry? (OK tentar novamente?) for No (Não), você deverá corrigir o problema no lado do cliente antes de enviar uma nova solicitação.

Código de status HTTP 400

Um código de status HTTP `400` indica um problema com sua solicitação. Alguns exemplos de problemas são falha de autenticação, parâmetros necessários que estão ausentes ou excedem a throughput provisionada de uma operação. Será necessário corrigir o problema no aplicativo antes de enviar a solicitação novamente.

AccessDeniedException

Mensagem: Ocorreu um erro (AccessDeniedException) ao chamar a operação: Usuário: não está autorizado a realizar: no recurso:. <Operation><User ARN><Operation><Resource ARN>

Você não está autorizado a executar a ação. Use o nome de recurso da Amazon (ARN) de um usuário autorizado ou a função do IAM para executar a operação.

OK para tentar novamente? Não

GroupFacesInProgressException

Mensagem: Falha ao agendar GroupFaces o trabalho. Existe um grupo existente para trabalhar nesta coleção.

Tentar novamente a operação depois que a tarefa existente for concluída.

OK para tentar novamente? Não

IdempotentParameterMismatchException

Mensagem: O ClientRequestToken: que você forneceu já está em uso. <Token>

Um parâmetro ClientRequestToken de entrada foi reutilizado com uma operação, mas pelo menos um dos outros parâmetros de entrada é diferente da chamada anterior para a operação.

OK para tentar novamente? Não

ImageTooLargeException

Mensagem: O tamanho da imagem é muito grande.

O tamanho da imagem de entrada excede o limite permitido. Se você estiver ligando DetectProtectiveEquipment, o tamanho ou a resolução da imagem excede o limite permitido. Para obter mais informações, consulte Diretrizes e cotas no Amazon Rekognition.

OK para tentar novamente? Não

InvalidImageFormatException

Mensagem: A solicitação tem um formato de imagem inválido.

O formato da imagem fornecida não tem suporte. Use um formato de imagem com suporte (.JPEG e .PNG). Para obter mais informações, consulte Diretrizes e cotas no Amazon Rekognition.

OK para tentar novamente? Não

InvalidPaginationTokenException

Mensagens

- Token inválido
- Token de paginação inválido

O token de paginação na solicitação não é válido. O token pode ter expirado.

OK para tentar novamente? Não

InvalidParameterException

Mensagem: A solicitação tem parâmetros inválidos.

Um parâmetro de entrada violou uma restrição. Valide seus parâmetros antes de chamar a operação de API novamente.

OK para tentar novamente? Não

Inválido S3 ObjectException

Mensagens:

- A solicitação tem um objeto do S3 inválido.
- Não é possível obter metadados do objeto do S3. Verifique a chave do objeto, a região e/ou as permissões de acesso.

O Amazon Rekognition não consegue acessar o objeto S3 que foi especificado na solicitação. Para obter mais informações, consulte Configurar o acesso ao S3: gerenciamento de acesso do AWS S3. Para obter informações sobre a solução de problemas, consulte Solução de problemas do Amazon S3.

OK para tentar novamente? Não

LimitExceededException

Mensagens:

- Limite do processador de fluxo excedido para a conta, limite — <Limite atual>
- <Número de trabalhos abertos> trabalhos abertos para o usuário <ARN do usuário> Limite máximo: <Limite máximo>

Um limite do serviço Amazon Rekognition foi excedido. Por exemplo, se você iniciar muitas tarefas do Amazon Rekognition Video simultaneamente, chamadas para iniciar operações, como `StartLabelDetection`, gerarão uma exceção `LimitExceededException` (código de status HTTP): 400) até que o número de trabalhos em execução simultânea fique abaixo do limite do serviço Amazon Rekognition.

OK para tentar novamente? Não

ProvisionedThroughputExceededException

Mensagens:

- Taxa provisionada excedida.
- Limite de download do S3 excedido.

O número de solicitações excedeu o limite da throughput. Para obter mais informações, consulte Limites do Amazon Rekognition Service.

Para solicitar um aumento de limite, siga as instruções em the section called “Crie um caso para alterar as cotas de TPS”.

OK para tentar novamente? Sim

ResourceAlreadyExistsException

Mensagem: O ID da coleção: <ID da coleção> já existe.

Uma coleção com o ID especificado já existe.

OK para tentar novamente? Não

ResourceInUseException

Mensagens:

- O nome do processador de fluxo já está em uso.
- O recurso especificado está em uso.
- Processador não disponível para interromper fluxo.
- Não é possível excluir o processador de fluxo.

Tente novamente quando o recurso estiver disponível.

OK para tentar novamente? Não

ResourceNotFoundException

Mensagem: várias mensagens dependendo da chamada de API.

O recurso especificado não existe.

OK para tentar novamente? Não

ThrottlingException

Mensagem: Diminua o ritmo; súbito aumento na taxa de solicitações.

O aumento da taxa de solicitações está muito rápido. Diminua o ritmo e aumente gradualmente a taxa de solicitações. Recomendamos que você recue exponencialmente e tente novamente. Por padrão, eles AWS SDKs usam lógica de repetição automática e recuo exponencial. Para obter mais informações, consulte Novas tentativas e recuo exponencial na AWS e Recuo exponencial e variação.

OK para tentar novamente? Sim

VideoTooLargeException

Mensagem: O tamanho do vídeo em bytes: <Tamanho do vídeo> é maior que o limite máximo de: <Tamanho máximo> bytes.

O tamanho do arquivo ou a duração da mídia fornecida é muito grande. Para obter mais informações, consulte Diretrizes e cotas no Amazon Rekognition.

OK para tentar novamente? Não

Código de status HTTP 5xx

Um código de status HTTP `5xx` indica um problema que deve ser resolvido pela AWS. Isso pode ser um erro temporário. Se for, você pode tentar fazer a solicitação novamente até que ela tenha êxito. Caso contrário, acesse o Painel de Status de Serviços da AWS para ver se há problemas operacionais com o serviço.

InternalServerError (HTTP 500)

Mensagem: Erro interno do servidor

O Amazon Rekognition teve um problema de serviço. Tente fazer a chamada novamente. Recue exponencialmente e tente novamente. Por padrão, eles AWS SDKs usam lógica de repetição automática e recuo exponencial. Para obter mais informações, consulte Novas tentativas e recuo exponencial na AWS e Recuo exponencial e variação.

OK para tentar novamente? Sim

ThrottlingException (HTTP 500)

Mensagem: Serviço indisponível

O Amazon Rekognition está temporariamente indisponível para processar a solicitação. Tente fazer a chamada novamente. Recomendamos que você recue exponencialmente e tente novamente. Por padrão, eles AWS SDKs usam lógica de repetição automática e recuo exponencial. Para obter mais informações, consulte Novas tentativas e recuo exponencial na AWS e Recuo exponencial e variação.

OK para tentar novamente? Sim

Tratamento de erros na aplicação

Para que seu aplicativo seja executado consistentemente, é necessário adicionar uma lógica para detectar erros e responder a eles. As abordagens comuns incluem o uso de blocos `try-catch` ou de uma instrução `if-then`.

A AWS SDKs realiza suas próprias tentativas e verificações de erros. Se você encontrar um erro ao usar uma das AWS SDKs, o código e a descrição do erro podem ajudá-lo a solucioná-lo.

Você também deve ver um `Request ID` na resposta. O `Request ID` pode ser útil se você precisa trabalhar com o AWS Support para diagnosticar um problema.

O trecho de código Java a seguir tenta detectar objetos em uma imagem e faz um tratamento de erro rudimentar. (Nesse caso, ele simplesmente informa ao usuário que houve falha na solicitação.)

```
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());
}
```

Neste trecho de código, o construtor `try-catch` lida com dois tipos diferentes de exceções:

- `AmazonRekognitionException` — Essa exceção ocorre se a solicitação do cliente foi transmitida corretamente para o Amazon Rekognition, mas o Amazon Rekognition não conseguiu processar a solicitação e, em vez disso, retornou uma resposta de erro.
- `AmazonClientException` — Essa exceção ocorre se o cliente não conseguiu obter uma resposta de um serviço ou se o cliente não conseguiu analisar a resposta de um serviço.

Usar o Amazon Rekognition como um serviço autorizado pelo FedRAMP

O programa de conformidade do AWS FedRAMP inclui o Amazon Rekognition como um serviço autorizado pelo FedRAMP. Se você for um cliente federal ou comercial, poderá usar o serviço para processar e armazenar cargas de trabalho confidenciais nas regiões Leste dos AWS EUA e Oeste dos EUA, com dados até o nível de impacto moderado. Você pode usar o serviço para cargas de trabalho confidenciais no limite de autorização da região AWS GovCloud (EUA), com dados até o nível de alto impacto. Para obter mais informações sobre conformidade com FedRAMP, consulte Conformidade com FedRAMP AWS.

Para ser compatível com o FedRAMP, você pode usar um endpoint FIPS (Federal Information Processing Standard - Norma federal de processamento de informações). Isso lhe dá acesso a módulos criptográficos 140-2 FIPS validados quando estiver trabalhando com informações confidenciais. Para obter mais informações sobre os endpoints do FIPS, consulte Visão geral da publicação 140-2 da FIPS.

Você pode usar o AWS Command Line Interface (AWS CLI) ou um dos AWS SDKs para especificar o endpoint usado pelo Amazon Rekognition.

Para conhecer os endpoints que podem ser usados com o Amazon Rekognition, consulte Regiões e endpoints do Amazon Rekognition.

A seguir, exemplos do tópico Listar coleções no Guia do desenvolvedor do Amazon Rekognition. Eles são modificados para especificar a região e o endpoint FIPS por meio dos quais o Amazon Rekognition é acessado.

Java

Para Java, use o método `withEndpointConfiguration` ao criar o cliente Amazon Rekognition. Este exemplo mostra as coleções em que você deve usar o endpoint FIPS na Região Leste dos EUA (Norte da Virgínia):

```
//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

Para o AWS CLI, use o `--endpoint-url` argumento para especificar o endpoint pelo qual o Amazon Rekognition é acessado. Este exemplo mostra as coleções em que você deve usar o endpoint FIPS na região Leste dos EUA (Ohio):

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

Python

Para o Python, use o argumento `endpoint_url` na função boto3.client. Defina-o para o endpoint que você desejar especificar. Este exemplo mostra as coleções em que você deve usar o endpoint FIPS na Região Oeste dos EUA (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()
```

Melhores práticas para sensores, imagens de entrada e vídeos

Esta seção contém informações sobre as melhores práticas para o uso do Amazon Rekognition. Essas práticas recomendadas ajudarão você a obter o melhor desempenho das operações invocadas. Se você está com dificuldades para obter os resultados esperados de uma operação, siga as práticas recomendadas documentadas aqui.

Para obter informações sobre a latência das operações de imagem, consulte o seguinte:

- the section called “Latência de operação do Amazon Rekognition Image”

As operações de comparação facial e pesquisa de faces exigem que você siga as práticas recomendadas específicas para encontrar faces em uma imagem. Esses requisitos também foram documentados no seguinte link:

- the section called “Recomendações para imagens de entrada de comparação facial”
- the section called “Recomendações para pesquisar faces em uma coleção”

As seções a seguir abordam como configurar sua câmera para cada tipo de mídia que o Amazon Rekognition consegue analisar:

- the section called “Recomendações para a configuração da câmera (imagem e vídeo)”
- the section called “Recomendações para a configuração da câmera (vídeo armazenado e streaming)”
- the section called “Recomendações para a configuração da câmera (streaming de vídeo)”

As operações de Vivacidade da face também têm práticas recomendadas próprias que devem ser seguidas para obter o melhor desempenho da ferramenta de verificação de vivacidade:

- the section called “Recomendações para o uso do Face Liveness”

Latência de operação do Amazon Rekognition Image

Para garantir a menor latência possível para as operações do Amazon Rekognition Image, considere o seguinte:

- A região do bucket do Amazon S3 que contém as imagens deve corresponder à região que você usa para operações da API Amazon Rekognition Image.
- Chamar uma operação do Amazon Rekognition Image com bytes de imagem é mais rápido do que carregar a imagem em um bucket do Amazon S3 e, em seguida, fazer referência à imagem carregada em uma operação do Amazon Rekognition Image. Considere essa abordagem se estiver fazendo upload de imagens para o Amazon Rekognition Image para processamento quase em tempo real. Por exemplo, imagens carregadas de uma câmera IP ou imagens carregadas por meio de um portal da web.
- Se a imagem já estiver em um bucket do Amazon S3, fazer referência a ela em uma operação do Amazon Rekognition Image provavelmente será mais rápido do que passar os bytes da imagem para a operação.

Recomendações para imagens de entrada de comparação facial

Os modelos usados para operações de comparação facial são projetados para trabalhar com uma grande variedade de poses, expressões faciais, faixas etárias, rotações, condições de iluminação e tamanhos. Recomendamos que você use as diretrizes a seguir ao escolher fotos de referência para CompareFaces ou para adicionar faces a uma coleção usando IndexFaces.

Recomendações gerais para entrada de imagens para operações faciais

- Use imagens claras e nítidas. Evite o máximo possível usar imagens que possam estar desfocadas devido à movimentação da câmera e da pessoa presente. DetectFaces pode ser usado para determinar o brilho e a nitidez de uma face.
- Para fins de detecção de olhares, é recomendável fazer o upload da imagem original no tamanho e qualidade originais.
- Use uma imagem com uma face que esteja dentro do intervalo recomendado de ângulos. O ângulo de inclinação deve ser menor que 30 graus para baixo e menos de 45 graus para cima. O ângulo de guinada deve ser menor que 45 graus em qualquer direção. Não há nenhuma restrição no ângulo de rotação lateral.
- Use uma imagem de uma face com ambos os olhos abertos e visíveis.

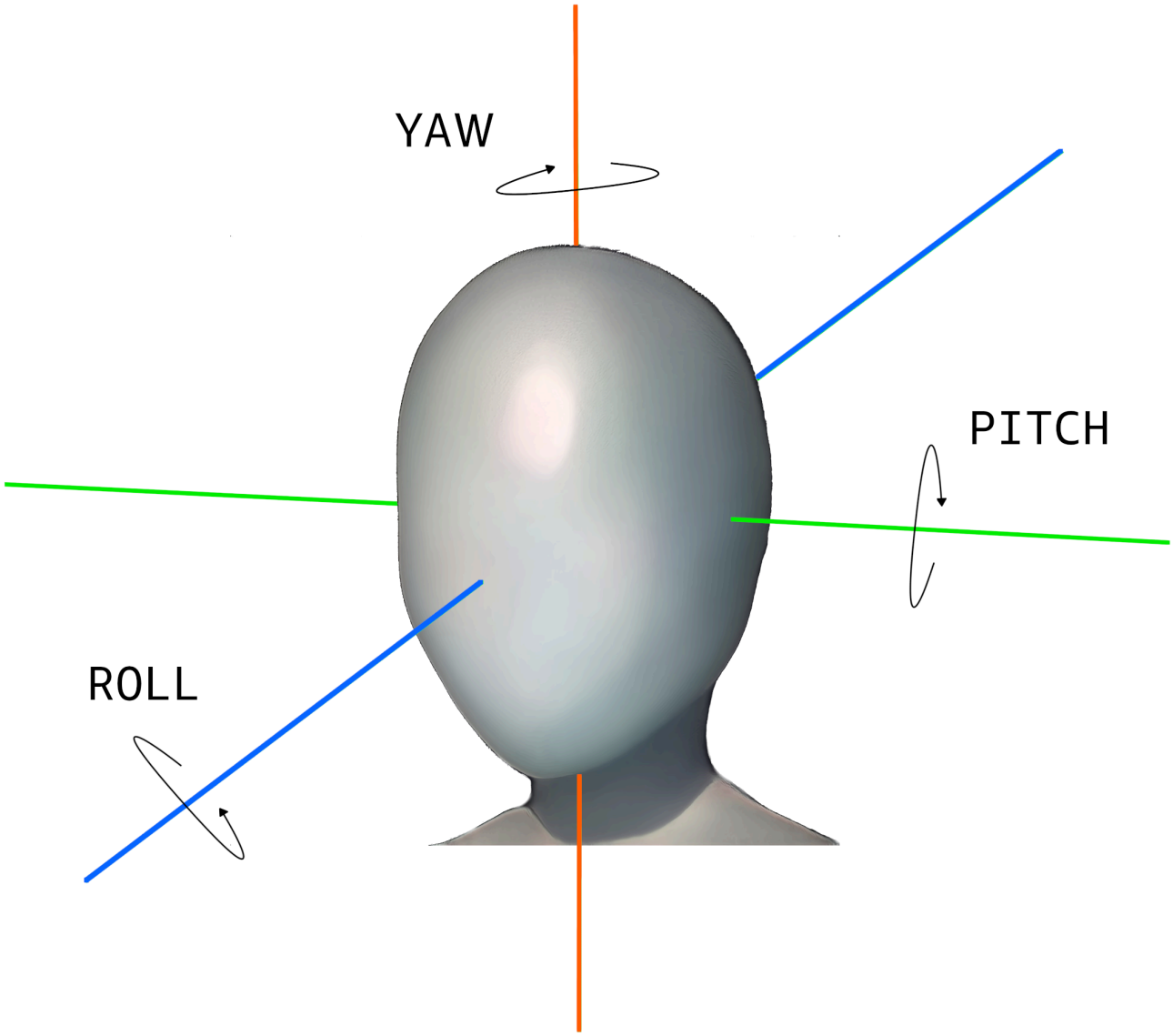
- Use uma imagem que não esteja obscurecida nem recortada de forma justa. A imagem deve conter toda a cabeça e ombros da pessoa. Ela não deve estar recortada na caixa delimitadora da face.
- Evite itens que bloqueiam a face, como testeiras e máscaras.
- Use uma imagem de face que ocupe uma grande parte da imagem. As imagens em que a face ocupa a maior parte do espaço da imagem apresentam maior precisão.
- Certifique-se de que as imagens sejam grandes o suficiente quanto à resolução. O Amazon Rekognition pode reconhecer faces tão pequenas quanto 50 x 50 pixels em resoluções de imagem de até 1920 x 1080. Imagens de resolução mais alta precisam de um tamanho mínimo maior da face. As faces maiores do que o tamanho mínimo fornecem um conjunto de resultados de comparação facial mais preciso.
- Use imagens coloridas.
- Use imagens com iluminação homogênea sobre a face, em vez de formas de iluminação variadas, como sombras.
- Use imagens com contraste suficiente com o plano de fundo. Um plano de fundo monocromático de alto contraste funciona bem.
- Use imagens de faces com expressões faciais neutras, com a boca fechada e pouco ou nenhum sorriso para aplicativos que exigem alta precisão.

Recomendações para pesquisar faces em uma coleção

- Ao pesquisar faces em uma coleção, certifique-se de que as imagens faciais recentes sejam indexadas.
- Ao criar uma coleção usando `IndexFaces`, use imagens de várias faces de um indivíduo com diferentes ângulos de inclinação e guinada (dentro do intervalo de ângulos recomendado). Recomendamos que pelo menos cinco imagens da pessoa sejam indexadas: de frente, rosto virado para a esquerda com uma inclinação de 45 graus ou menos, rosto virado para a direita com uma inclinação de 45 graus ou menos, rosto inclinado para baixo com uma inclinação de 30 graus ou menos e rosto inclinado para cima com uma inclinação de 45 graus ou menos. Se você deseja monitorar se essas instâncias de faces pertencem à mesma pessoa, considere o uso do atributo de ID da imagem externa, se há apenas uma face na imagem que está sendo indexada. Por exemplo, cinco imagens de John Doe podem ser rastreadas na coleção com a imagem IDs externa como. `John_Doe_1.jpg, … John_Doe_5.jpg`

Recomendações para a configuração da câmera (imagem e vídeo)

As recomendações a seguir são adicionais a Recomendações para imagens de entrada de comparação facial.



- Resolução da imagem: não há requisitos mínimos para a resolução da imagem, desde que a resolução da face seja de 50 x 50 pixels para imagens com uma resolução total de até 1920 x 1080. Imagens de resolução mais alta precisam de um tamanho mínimo maior da face.

> **Note**
>
> A recomendação anterior é baseada na resolução nativa da câmera. A geração de uma imagem de alta resolução a partir de uma imagem de baixa resolução não produz os resultados necessários para a pesquisa de faces (devido aos artefatos gerados pela expansão da imagem).

- Ângulo da câmera: há três medidas para o ângulo da câmera, como inclinação, rotação e guinada.
  - Inclinação: recomendamos um ângulo de inclinação de menos de 30 graus quando a câmera está voltada para baixo e menos de 45 graus quando a câmera está voltada para cima.
  - Rotação lateral: não há um requisito mínimo para esse parâmetro. O Amazon Rekognition pode lidar com qualquer quantidade de rolagem.
  - Guinada: recomendamos um ângulo de guinada de menos de 45 graus em qualquer direção.

  O ângulo da face ao longo de qualquer eixo que é capturado pela câmera é uma combinação do ângulo da câmera voltada para a cena e do ângulo da cabeça da pessoa que está na cena. Por exemplo, se a câmera está inclinada para baixo em 30 graus e a pessoa também está com a cabeça inclinada para baixo em 30 graus, a inclinação real da face vista pela câmera é 60 graus. Nesse caso, o Amazon Rekognition não seria capaz de reconhecer a face. Recomendamos configurar a câmera de forma que os ângulos da câmera se baseiem na suposição de que as pessoas, em geral, estejam olhando para a câmera, com a inclinação total (combinação de face e câmera) de 30 graus ou menos.
- Zoom da câmera: a resolução mínima recomendada para a face de 50 x 50 pixels deve direcionar essa configuração da câmera. Recomendamos que você use a configuração de zoom de uma câmera de forma que cada face obtenha uma resolução não inferior a 50 x 50 pixels.
- Altura da câmera: a inclinação recomendada para a câmera deve direcionar esse parâmetro.

Recomendações para a configuração da câmera (vídeo armazenado e streaming)

As recomendações a seguir são adicionais a Recomendações para a configuração da câmera (imagem e vídeo).

- O codec deve ser codificado pelo h.264.
- A taxa de quadros recomendada é de 30 fps. (Ela não deve ser inferior a 5 fps.)

- A taxa de bits recomendada do codificador é 3 Mbps. (Ela não deve ser inferior a 1,5 Mbps.)
- Taxa de quadros vs. Resolução de quadros: se a taxa de bits do codificador for uma restrição, recomendamos dar preferência a uma resolução de quadros mais alta em vez de uma taxa de quadros mais alta para obter melhores resultados de busca de faces. Isso garante que o Amazon Rekognition obtenha o quadro de melhor qualidade dentro da taxa de bits alocada. No entanto, há uma desvantagem nisso. Devido à baixa taxa de quadros, a câmera perde movimento rápido em uma cena. É importante compreender o dilema entre esses dois parâmetros para uma determinada configuração. Por exemplo, se a taxa de bits máxima possível for 1,5 Mbps, uma câmera poderá capturar 1080p a 5 fps ou 720p a 15 fps. A escolha entre os dois vai depender do aplicativo, desde que a resolução de faces recomendada de 50 x 50 pixels seja atendida.

Recomendações para a configuração da câmera (streaming de vídeo)

A recomendação a seguir é adicional a Recomendações para a configuração da câmera (vídeo armazenado e streaming).

Uma restrição adicional com aplicativos de streaming é a largura de banda da Internet. Para vídeo ao vivo, o Amazon Rekognition aceita apenas o Amazon Kinesis Video Streams como entrada. Você deve compreender a dependência entre a taxa de bits do codificador e a largura de banda da rede disponível. A largura de banda disponível deve ser compatível, no mínimo, com a mesma taxa de bits que a câmera está usando para codificar o streaming ao vivo. Isso garante que qualquer imagem capturada pela câmera seja retransmitida por meio do Amazon Kinesis Video Streams. Se a largura de banda disponível for menor que a taxa de bits do codificador, o Amazon Kinesis Video Streams descartará bits com base na largura de banda da rede. Isso resulta em baixa qualidade de vídeo.

Uma configuração típica de streaming envolve a conexão de várias câmeras a um hub de rede que retransmite os streamings. Nesse caso, a largura de banda deve acomodar a soma acumulada dos streamings provenientes de todas as câmeras conectadas ao hub. Por exemplo, se o hub estiver conectado a cinco câmeras com codificação a 1,5 Mbps, a largura de banda da rede disponível deverá ser pelo menos 7,5 Mbps. Para garantir que nenhum pacote seja descartado, considere manter a largura de banda da rede maior que 7,5 MBPS para acomodar as variações decorrentes de conexões perdidas entre uma câmera e o hub. O valor real dependerá da confiabilidade da rede interna.

Recomendações para o uso do Face Liveness

Recomendamos as seguintes melhores práticas ao usar o Rekognition Face Liveness:

- Os usuários devem concluir a verificação Face Liveness em ambientes que não sejam muito escuros ou muito claros e tenham iluminação bastante uniforme.
- Os usuários devem aumentar o brilho da tela até o nível máximo ao fazer verificações em navegadores da web. O Mobile Native SDKs ajusta o brilho da tela automaticamente.
- Escolha um limite de pontuação de confiança que reflita a natureza do seu caso de uso. Para casos de uso com maiores preocupações de segurança, use um limite alto.
- Execute regularmente verificações humanas nas imagens de auditoria para garantir que os ataques falsos sejam mitigados com o limite de confiança que você definiu.
- Ofereça um caminho alternativo de verificação da vivacidade facial para seus usuários se eles forem fotossensíveis ou não quiserem verificar a vivacidade facial usando o Rekognition.
- Não envie nem exiba a pontuação de verificação de atividade no aplicativo do usuário. Envie apenas um sinal de aprovação ou reprovação.
- Permitir apenas cinco verificações de vivacidade com falha em três minutos em um único dispositivo. Depois de cinco falhas, o tempo limite do usuário é de 30 a 60 minutos. Se o padrão for visto de 3 a 5 vezes repetidamente, impeça o dispositivo do usuário de fazer chamadas adicionais.
- Implemente a tela de preparação em seu fluxo de trabalho para que os usuários possam passar pelas verificações de vivacidade facial com mais facilidade.
- Você é responsável por fornecer avisos de privacidade legalmente adequados e obter o consentimento necessário de seus Usuários Finais para o processamento, armazenamento, uso e transferência de conteúdo pela Face Liveness.

Detectando objetos e conceitos

Esta seção fornece informações para detectar rótulos em imagens e vídeos com o Amazon Rekognition Image e o Amazon Rekognition Video.

Um rótulo ou tag é um objeto ou conceito (incluindo cenas e ações) encontrado em uma imagem ou vídeo com base em seu conteúdo. Por exemplo, uma imagem de pessoas em uma praia tropical pode conter rótulos como Palmeira (objeto), Praia (cena), Corrida (ação) e Ao ar livre (conceito).

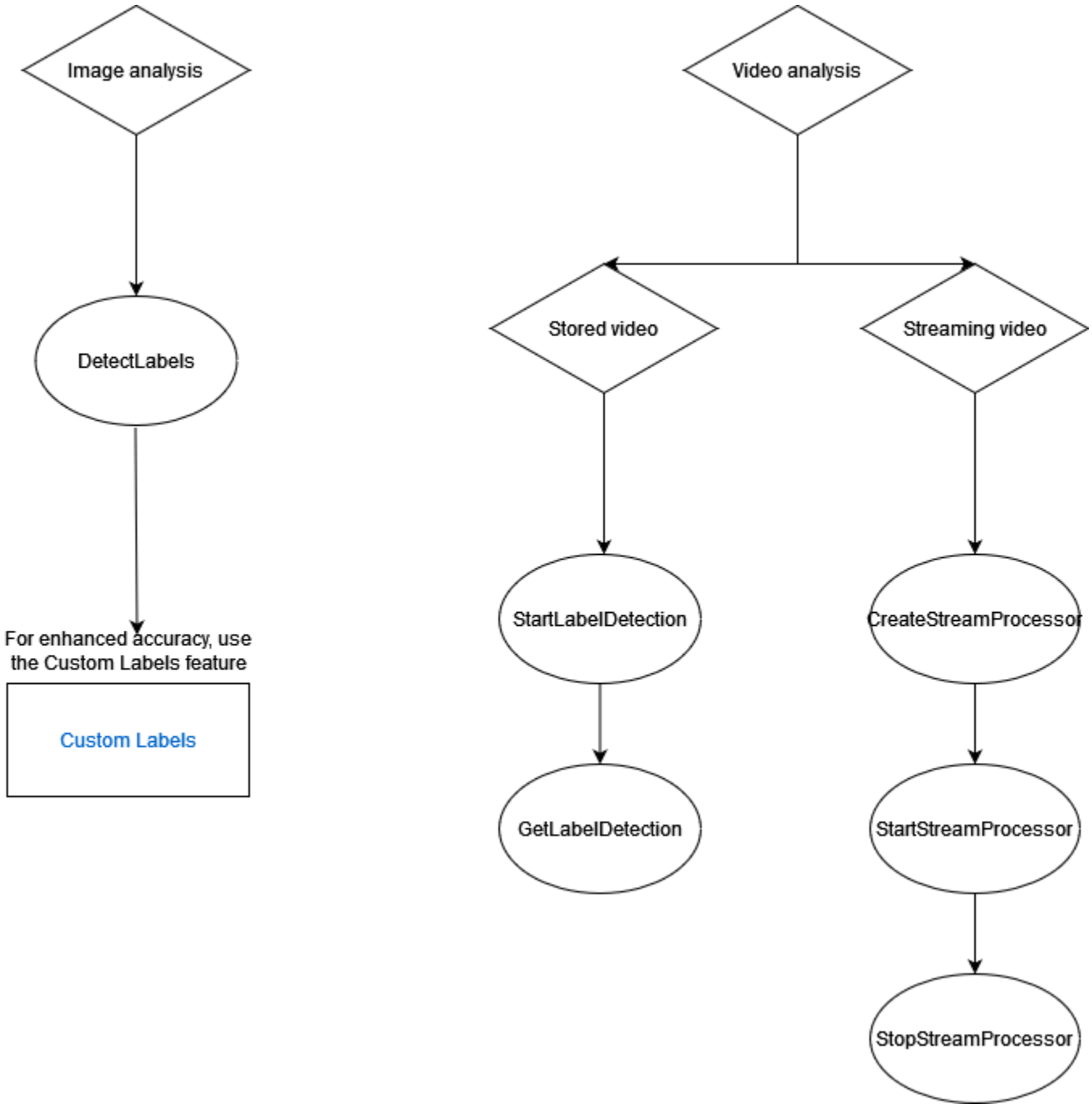
Rótulos compatíveis com as operações de detecção de rótulos do Rekognition

- Para baixar a lista mais recente de rótulos e caixas delimitadoras de objetos compatíveis com o Amazon Rekognition, clique aqui.
- Para baixar a lista anterior de rótulos e caixas delimitadoras de objetos, clique aqui.

> **Note**
>
> O Amazon Rekognition faz previsões binárias de gênero (homem, mulher, menina etc.) com base na aparência física de uma pessoa em uma imagem específica. Esse tipo de previsão não foi projetado para categorizar a identidade de gênero de uma pessoa, e você não deve usar o Amazon Rekognition para fazer essa determinação. Por exemplo, um ator masculino usando uma peruca de cabelos compridos e brincos para um papel pode ser considerado feminino.
> Usar o Amazon Rekognition para fazer previsões binárias de gênero é mais adequado para casos de uso em que estatísticas agregadas de distribuição por gênero precisam ser analisadas sem identificar usuários específicos. Por exemplo, a porcentagem de usuários que são mulheres em comparação com homens em uma plataforma de mídia social.
> Não recomendamos o uso de previsões binárias de gênero para tomar decisões que afetam os direitos, a privacidade ou o acesso de um indivíduo aos serviços.

O Amazon Rekognition devolve etiquetas em inglês. Você pode usar o Amazon Translate para traduzir rótulos em inglês para outros idiomas.

O diagrama a seguir mostra a ordem das operações de chamada, dependendo de suas metas de uso das operações do Amazon Rekognition Image ou do Amazon Rekognition Video:



Objetos de resposta de rótulos

Caixas delimitadoras

O Amazon Rekognition Image e o Amazon Rekognition Video podem devolver a caixa delimitadora para rótulos de objetos comuns, como carros, móveis, roupas ou animais de estimação. As informações da caixa delimitadora não são retornadas para rótulos de objetos menos comuns.

Você pode usar caixas delimitadoras para encontrar as localizações exatas de objetos em uma imagem, contar instâncias de objetos detectados ou para medir o tamanho de um objeto usando as dimensões da caixa delimitadora.

Por exemplo, na imagem a seguir, o Amazon Rekognition Image é capaz de detectar a presença de uma pessoa, um skate, carros estacionados e outras informações. O Amazon Rekognition Image também retorna a caixa delimitadora de uma pessoa detectada e de outros objetos detectados, como carros e rodas.



Escore de confiança

O Amazon Rekognition Video e o Amazon Rekognition Image fornecem uma pontuação percentual da confiança que o Amazon Rekognition tem na precisão de cada rótulo detectado.

Pais

O Amazon Rekognition Image e o Amazon Rekognition Video usam uma taxonomia hierárquica de rótulos ancestrais para categorizá-los. Por exemplo, uma pessoa caminhando em uma rua pode

ser detectada como um Pedestre. O rótulo pai de um Pedestre é Pessoa. Ambos os rótulos são retornados na resposta. Todos os rótulos ancestrais são retornados e cada rótulo contém uma lista do pai e de outros rótulos ancestrais. Por exemplo, rótulos de avós e bisavós, se existirem. Você pode usar rótulos pai para criar grupos de rótulos relacionados e permitir a consulta de rótulos semelhantes em uma ou mais imagens. Por exemplo, uma consulta para todos os Veículos pode retornar um carro de uma imagem e uma moto de outra.

Categorias

O Amazon Rekognition Image e o Amazon Rekognition Video retornam informações sobre categorias de etiquetas. Os rótulos fazem parte de categorias que agrupam rótulos individuais com base em funções e contextos comuns, como "Veículos e automóveis" e "Alimentos e bebidas". Uma categoria de rótulo pode ser uma subcategoria de uma categoria principal.

Aliases

Além de devolver etiquetas, o Amazon Rekognition Image e o Amazon Rekognition Video retornam quaisquer aliases associados à etiqueta. Os aliases são rótulos com o mesmo significado ou rótulos que são visualmente intercambiáveis com o rótulo principal retornado. Por exemplo, "Cell Phone" é um alias de "Mobile Phone".

Nas versões anteriores, o Amazon Rekognition Image retornava aliases como "Celular" na mesma lista de nomes de rótulos primários que continha "Dispositivo móvel". Agora, o Amazon Rekognition Image retorna "Celular" em um campo chamado "aliases" e "Dispositivo móvel" na lista de nomes de etiquetas primárias. Se seu aplicativo depende das estruturas retornadas por uma versão anterior do Rekognition, talvez seja necessário transformar a resposta atual retornada pelas operações de detecção de rótulos de imagem ou vídeo na estrutura de resposta anterior, na qual todos os rótulos e aliases são retornados como rótulos primários.

Se você precisar transformar a resposta atual da DetectLabels API (para detecção de rótulos em imagens) na estrutura de resposta anterior, veja o exemplo de código emTransformando a resposta DetectLabels.

Se você precisar transformar a resposta atual da GetLabelDetection API (para detecção de rótulos em vídeos armazenados) na estrutura de resposta anterior, veja o exemplo de código emTransformando a resposta GetLabelDetection .

Propriedades da imagem

O Amazon Rekognition Image retorna informações sobre a qualidade da imagem (nitidez, brilho e contraste) de toda a imagem. A nitidez e o brilho também são retornados para o primeiro plano e o plano de fundo da imagem. As propriedades da imagem também podem ser usadas para detectar cores dominantes de toda a imagem, primeiro plano, fundo e objetos com caixas delimitadoras.



Veja a seguir um exemplo dos ImageProperties dados contidos na resposta de uma DetectLabels operação para a imagem em andamento:

Image Properties	Dominant Colors Examples and Pixel Percentage	Image Quality
Entire Image	Hex code #808080, RGB (128, 128, 128), 15.72	Brightness: 76.08 Sharpness: 89.72 Contrast: 88.42
	Hex code #000000, RGB (0, 0, 0), 15.10	
	Hex code #696969, RGB (105, 105, 105), 14.02	
	Hex code #8fbc8f, RGB (143, 188, 143), 12.70	
	Hex code #5f9ea0, RGB (95, 158, 160), 11.92	
Foreground	Hex code #8fbc8f, RGB (143, 188, 143), 30.18	Brightness: 79.48 Sharpness: 93.47
	Hex code #5f9ea0, RGB (95, 158, 160), 24.29	
	Hex code #000000, RGB (0, 0, 0), 12.02	
	Hex code #2f4f4f, RGB (47, 79, 79), 9.20	
	Hex code #696969, RGB (105, 105, 105), 8.95	
Background	Hex code #808080, RGB (128, 128, 128), 21.16	Brightness: 74.42 Sharpness: 87.84
	Hex code #2f4f4f, RGB (47, 79, 79), 14.61	
	Hex code #000000, RGB (0, 0, 0), 14.23	
	Hex code #696969, RGB (105, 105, 105), 13.19	
	Hex code #ffebcd, RGB (255, 235, 205), 12.80	
Car (example of objects with bounding boxes)	Hex code #5f9ea0, RGB (95, 158, 160), 29.18	Not applicable
	Hex code #8fbc8f, RGB (143, 188, 143), 14.39	
	Hex code #000000, RGB (0, 0, 0), 11.76	
	Hex code #808080, RGB (128, 128, 128), 11.38	
	Hex code #2f4f4f, RGB (47, 79, 79), 9.44	

As propriedades da imagem não estão disponíveis para o Amazon Rekognition Video.

Versão do modelo

Tanto o Amazon Rekognition Image quanto o Amazon Rekognition Video retornam a versão do modelo de detecção de rótulos usado para detectar rótulos em uma imagem ou vídeo armazenado.

Filtros de inclusão e exclusão

Você pode filtrar os resultados retornados pelas operações de detecção de rótulos do Amazon Rekognition Image e do Amazon Rekognition Video. Filtre os resultados fornecendo critérios de filtragem para rótulos e categorias. Os filtros de etiquetas podem ser inclusivos ou exclusivos.

Consulte Detectar rótulos em uma imagem para obter mais informações sobre a filtragem dos resultados obtidos com `DetectLabels`.

Consulte Detectando rótulos em um vídeo para obter mais informações sobre a filtragem dos resultados obtidos por `GetLabelDetection`.

Classificação e agregação de resultados

Os resultados obtidos de determinadas operações do Amazon Rekognition Video podem ser classificados e agregados de acordo com registros de data e hora e segmentos de vídeo. Ao recuperar os resultados de um trabalho de Detecção de Rótulos ou Moderação de Conteúdo, com `GetLabelDetection` ou `GetContentModeration` respectivamente, você pode usar os argumentos `AggregateBy` e `SortBy` para especificar como deseja que seus resultados sejam retornados. Você pode usar `SortBy` com `TIMESTAMP` ou `NAME` (nomes de rótulos) e usar `TIMESTAMPS` ou `SEGMENTS` com o AggregateBy argumento.

Detectar rótulos em uma imagem

Você pode usar a DetectLabelsoperação para detectar rótulos (objetos e conceitos) em uma imagem e recuperar informações sobre as propriedades de uma imagem. As propriedades da imagem incluem atributos como a cor do primeiro plano e do plano de fundo e a nitidez, o brilho e o contraste da imagem. Você pode recuperar apenas os rótulos em uma imagem, apenas as propriedades da imagem ou ambas. Para obter um exemplo, consulte Analisando imagens armazenadas em um bucket do Amazon S3.

Os exemplos a seguir usam various AWS SDKs e the AWS CLI to call`DetectLabels`. Para obter informações sobre a resposta da operação `DetectLabels`, consulte DetectLabels resposta.

Para detectar rótulos em uma imagem

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Faça upload de uma imagem que contenha um ou mais objetos, como árvores, casas e barcos, para seu bucket do S3. A imagem deve estar no formato .jpg ou .png.

   Para obter instruções, consulte Como fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

3. Use os exemplos a seguir para chamar a operação `DetectLabels`.

   Java

   Este exemplo exibe uma lista de rótulos que foram detectados na imagem de entrada. Substitua os valores de `bucket` e `photo` pelos nomes do bucket do Amazon S3 e da imagem usados na etapa 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

Esse exemplo exibe a saída JSON da operação da CLI `detect-labels`. Substitua os valores de `bucket` e `photo` pelos nomes do bucket do Amazon S3 e da imagem usados na etapa 2. Substitua o valor de `profile-name` com o nome do seu perfil de desenvolvedor.

```
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 você estiver acessando a CLI em um dispositivo Windows, use aspas duplas em vez de aspas simples e escape das aspas duplas internas com barra invertida (ou seja, \) para resolver quaisquer erros de analisador que você possa encontrar. Para obter um exemplo, veja o seguinte:

```
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

Este exemplo exibe os rótulos que foram detectados na imagem de entrada. Na função `main`, substitua os valores de `bucket` e `photo` pelos nomes do bucket e da imagem do Amazon S3 que você usou na Etapa 2. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Este exemplo exibe uma lista de rótulos que foram detectados na imagem de entrada. Substitua os valores de `bucket` e `photo` pelos nomes do bucket do Amazon S3 e da imagem usados na etapa 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

Este exemplo exibe uma lista de rótulos que foram detectados na imagem de entrada. Substitua os valores de `bucket` e `photo` pelos nomes do bucket do Amazon S3 e da imagem usados na etapa 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

Este exemplo exibe uma lista de rótulos que foram detectados na imagem de entrada. Substitua os valores de `bucket` e `photo` pelos nomes do bucket do Amazon S3 e da imagem usados na etapa 2. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

Se você estiver usando TypeScript definições, talvez seja necessário usar `import AWS from 'aws-sdk'` em vez de`const AWS = require('aws-sdk')`, para executar o programa com o Node.js. Você pode consultar o AWS SDK para Javascript para obter mais

detalhes. Dependendo de como você configurou suas configurações, talvez você também precise especificar sua região com `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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
//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 solicitação de operação

A entrada de `DetectLabel` é uma imagem. Neste exemplo de entrada JSON, a imagem de origem é carregada de um bucket do Amazon S3. `MaxLabels` é o número máximo de rótulos a serem retornados na resposta. `MinConfidence` é a confiança mínima que o Amazon Rekognition Image deve ter na precisão da etiqueta detectada para que ela seja retornada na resposta.

Os recursos permitem que você especifique um ou mais recursos da imagem que você deseja retornar, permitindo que você selecione `GENERAL_LABELS` e `IMAGE_PROPERTIES`. A inclusão de `GENERAL_LABELS` retornará os rótulos detectados na imagem de entrada, enquanto a inclusão de `IMAGE_PROPERTIES` permitirá que você acesse a cor e a qualidade da imagem.

As configurações permitem filtrar os itens devolvidos tanto para os recursos `GENERAL_LABELS` quanto para os recursos `IMAGE_PROPERTIES`. Para etiquetas, você pode usar filtros inclusivos e exclusivos. Você também pode filtrar por rótulo específico, rótulos individuais ou por categoria de rótulo:

- LabelInclusionFilters - Permite que você especifique quais rótulos você deseja incluir na resposta.
- LabelExclusionFilters - Permite que você especifique quais rótulos você deseja excluir da resposta.

- LabelCategoryInclusionFilters - Permite que você especifique quais categorias de etiquetas você deseja incluir na resposta.
- LabelCategoryExclusionFilters - Permite que você especifique quais categorias de rótulos você deseja excluir da resposta.

Você também pode combinar filtros inclusivos e exclusivos de acordo com suas necessidades, excluindo alguns rótulos ou categorias e incluindo outros.

`IMAGE_PROPERTIES` referem-se às cores dominantes e aos atributos de qualidade de uma imagem, como nitidez, brilho e contraste. Ao detectar `IMAGE_PROPERTIES`, você pode especificar o número máximo de cores dominantes a serem retornadas (o padrão é 10) usando o parâmetro `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 resposta

A resposta de `DetectLabels` é uma matriz de rótulos detectados na imagem e o nível de confiança em que foram detectadas.

Esta é uma resposta de exemplo de `DetectLabels`. O exemplo de resposta abaixo contém vários atributos retornados para GENERAL_LABELS, incluindo:

- Nome: o nome do rótulo detectado. Neste exemplo, a operação detectou um objeto com o rótulo Celular.
- Confiança: cada rótulo tem um nível de confiança associado. Neste exemplo, a confiança no rótulo foi de 99,36%.
- Pais: os rótulos ancestrais de um rótulo detectado. Neste exemplo, o rótulo Celular tem um rótulo principal chamado Telefone.
- Aliases: informações sobre possíveis aliases para o rótulo. Neste exemplo, a etiqueta Celular tem um possível alias de Celular.
- Categorias: a categoria de etiqueta à qual a etiqueta detectada pertence. Neste exemplo, é Tecnologia e Computação.

A resposta para rótulos de objetos comuns inclui as informações da caixa delimitadora para a localização do rótulo na imagem de entrada. Por exemplo, o rótulo Pessoa tem uma matriz de instâncias que contém duas caixas delimitadoras. Essas são as localizações de duas pessoas detectadas na imagem.

A resposta também inclui atributos relacionados a IMAGE_PROPERTIES. Os atributos apresentados pelo recurso IMAGE_PROPERTIES são:

- Qualidade: informações sobre nitidez, brilho e contraste da imagem de entrada, pontuadas entre 0 e 100. A qualidade é relatada para toda a imagem e para o plano de fundo e primeiro plano da imagem, se disponível. No entanto, o contraste é relatado apenas para a imagem inteira, enquanto a nitidez e o brilho também são relatados para fundo e primeiro plano.
- Cor dominante: uma matriz das cores dominantes na imagem. Cada cor dominante é descrita com um nome de cor simplificado, uma paleta de cores CSS, valores RGB e um código hexadecimal.
- Primeiro plano: informações sobre as cores, nitidez e brilho dominantes do primeiro plano da imagem de entrada.
- Plano de fundo: informações sobre as cores, nitidez e brilho dominantes do fundo da imagem de entrada.

Quando GENERAL_LABELS e IMAGE_PROPERTIES são usados juntos como parâmetros de entrada, o Amazon Rekognition Image também retornará as cores dominantes dos objetos com caixas delimitadoras.

O campo `LabelModelVersion` contém o número da versão do modelo de detecção usado por `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"
}
```

Transformando a resposta DetectLabels

Ao usar a DetectLabels API, talvez você precise que a estrutura de resposta imite a estrutura de resposta da API mais antiga, na qual tanto os rótulos principais quanto os aliases estavam contidos na mesma lista.

Veja a seguir um exemplo da resposta atual da API de DetectLabels:

```
"Labels": [
        {
            "Name": "Mobile Phone",
            "Confidence": 99.99717712402344,
            "Instances": [],
            "Parents": [
                {
                "Name": "Phone"
                }
             ],
            "Aliases": [
                {
                "Name": "Cell Phone"
                }
             ]
        }
 ]
```

O exemplo a seguir mostra a resposta anterior da DetectLabelsAPI:

```
"Labels": [
        {
            "Name": "Mobile Phone",
            "Confidence": 99.99717712402344,
            "Instances": [],
            "Parents": [
                {
                "Name": "Phone"
                }
             ]
         },
         {
            "Name": "Cell Phone",
            "Confidence": 99.99717712402344,
            "Instances": [],
            "Parents": [
                {
                "Name": "Phone"
                }
             ]
         },
]
```

Se necessário, você pode transformar a resposta atual para seguir o formato da resposta mais antiga. Você pode usar o código de exemplo a seguir para transformar a resposta mais recente da API na estrutura de resposta da API anterior:

Python

O exemplo de código a seguir demonstra como transformar a resposta atual da DetectLabels API. No exemplo de código abaixo, você pode substituir o valor *EXAMPLE_INFERENCE_OUTPUT* de pela saída de uma DetectLabels operação que você executou.

```
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)
```

Abaixo está um exemplo da resposta transformada:

```
#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"
            ]
        }
    ]
}
```


Detectando rótulos em um vídeo

O Amazon Rekognition Video pode detectar rótulos (objetos e conceitos) e a hora em que um rótulo é detectado em um vídeo. Para obter um exemplo de código do SDK, consulte Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK). Para obter um AWS CLI exemplo, consulteAnalisando um vídeo com o AWS Command Line Interface.

A detecção de rótulos do Amazon Rekognition Video é uma operação assíncrona. Para iniciar a detecção de rótulos em um vídeo, ligue StartLabelDetection.

O Amazon Rekognition Video publica o status de conclusão da análise de vídeo em um tópico do Amazon Simple Notification Service. Se a análise do vídeo for bem-sucedida, chame GetLabelDetection para obter os rótulos detectados. Para obter informações sobre como chamar

as operações da API de análise de vídeo, consulte Chamando as operações de vídeo do Amazon Rekognition Video.

StartLabelDetectionSolicitação

O exemplo a seguir é uma solicitação para a operação `StartLabelDetection`. Você fornece à operação `StartLabelDetection` um vídeo armazenado em um bucket do Amazon S3. No exemplo de solicitação JSON, o bucket do Amazon S3 e o nome do vídeo são especificados com `MinConfidence`, `Features`, `Settings`, e `NotificationChannel`.

`MinConfidence` é a confiança mínima que o Amazon Rekognition Video deve ter na precisão do rótulo detectado, ou de uma caixa delimitadora de instância (se detectada), para que ele seja retornado na resposta.

Com `Features`, você pode especificar que deseja que GENERAL_LABELS seja retornado como parte da resposta.

Com `Settings`, você pode filtrar os itens retornados para GENERAL_LABELS. Para etiquetas, você pode usar filtros inclusivos e exclusivos. Você também pode filtrar por rótulo específico, rótulos individuais ou por categoria de rótulo:

- `LabelInclusionFilters`: usado para especificar quais rótulos você deseja incluir na resposta
- `LabelExclusionFilters`: usado para especificar quais rótulos você deseja excluir da resposta.
- `LabelCategoryInclusionFilters`: usado para especificar quais categorias de rótulos você deseja incluir na resposta.
- `LabelCategoryExclusionFilters`: usado para especificar quais categorias de rótulos você deseja excluir da resposta.

Você também pode combinar filtros inclusivos e exclusivos de acordo com suas necessidades, excluindo alguns rótulos ou categorias e incluindo outros.

`NotificationChannel` é o ARN do tópico do Amazon SNS no qual você deseja que o Amazon Rekognition Video publique o status de conclusão da operação de detecção de rótulos. Se você estiver usando a política de permissões `AmazonRekognitionServiceRole`, o tópico do Amazon SNS deverá ter um nome de tópico que comece com Rekognition.

Veja a seguir um exemplo de solicitação `StartLabelDetection` no formato JSON, incluindo filtros:

```
{
    "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 Resposta da operação

O `GetLabelDetection` retorna uma matriz (`Labels`) que contém informações sobre os rótulos detectados no vídeo. A matriz pode ser classificada por hora ou pelo rótulo detectado ao especificar o parâmetro `SortBy`. Você também pode selecionar como os itens de resposta são agregados usando o parâmetro `AggregateBy`.

O exemplo a seguir é a resposta JSON do `GetLabelDetection`. Na resposta, observe o seguinte:

- Ordem de classificação — A matriz de etiquetas retornadas é classificada por hora. Para classificar por rótulo, especifique `NAME` no parâmetro de entrada `SortBy` para `GetLabelDetection`. Se o rótulo aparecer várias vezes no vídeo, haverá várias instâncias do elemento (LabelDetection). A ordem de classificação padrão é `TIMESTAMP`, enquanto a ordem de classificação secundária é `NAME`.

- Informações do rótulo: o elemento da `LabelDetection` matriz contém um objeto (Label) que, por sua vez, contém o nome do rótulo e a confiança que o Amazon Rekognition tem na precisão do rótulo detectado. Um objeto `Label` também inclui uma taxonomia hierárquica de rótulos e informações de caixa delimitadora para rótulos comuns. `Timestamp` é a hora em que o rótulo foi detectado, definido como o número de milissegundos decorridos desde o início do vídeo.

  As informações sobre quaisquer categorias ou aliases associados a uma etiqueta também são retornadas. Para resultados agregados por `SEGMENTS` do vídeo, as estruturas `StartTimestampMillis`, `EndTimestampMillis` e `DurationMillis` são retornadas, definindo a hora de início, a hora de término e a duração de um segmento, respectivamente.
- Agregação: specifica como os resultados são agregados quando retornados. O padrão é agregar por `TIMESTAMPS`. Você também pode optar por agregar por `SEGMENTS`, o que agrega os resultados em uma janela de tempo. Se a agregação for feita por `SEGMENTS`, as informações sobre instâncias detectadas com caixas delimitadoras não serão retornadas. Somente rótulos detectados durante os segmentos são retornados.
- Informações de paginação – O exemplo mostra uma página de informações de detecção de rótulo. Você pode especificar quantos objetos `LabelDetection` retornar no parâmetro de entrada `MaxResults` para `GetLabelDetection`. Se existirem mais resultados além de `MaxResults`, o `GetLabelDetection` retornará um token (`NextToken`) usado para obter a próxima página de resultados. Para obter mais informações, consulte Obter os resultados da análise do Amazon Rekognition Video.
- Informações de vídeo – a resposta inclui informações sobre o formato do vídeo (`VideoMetadata`) em cada página de informações retornada pelo `GetLabelDetection`.

Veja a seguir um exemplo de GetLabelDetection resposta no formato JSON com agregação por 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
    }
}
```

Veja a seguir um exemplo de GetLabelDetection resposta no formato JSON com agregação por SEGMENTOS:

```
{
    "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
    }
}
```

Transformando a resposta GetLabelDetection

Ao recuperar resultados com a operação da GetLabelDetection API, talvez você precise que a estrutura de resposta imite a estrutura de resposta da API mais antiga, na qual tanto os rótulos primários quanto os aliases estavam contidos na mesma lista.

O exemplo de resposta JSON encontrado na seção anterior exibe a forma atual da resposta da API de. GetLabelDetection

O exemplo a seguir mostra a resposta anterior da 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 necessário, você pode transformar a resposta atual para seguir o formato da resposta mais antiga. Você pode usar o código de exemplo a seguir para transformar a resposta mais recente da API na estrutura de resposta da API anterior:

```
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
```

Detectar rótulos em eventos de streaming de vídeo

Você pode usar o Amazon Rekognition Video para detectar rótulos em streaming de vídeo. Para fazer isso, você cria um processador de streaming (CreateStreamProcessor) para iniciar e gerenciar a análise do streaming de vídeo.

O Amazon Rekognition Video usa o Amazon Kinesis Video Streams para receber e processar um fluxo de vídeo. Ao criar o processador de streaming, você escolhe o que deseja que o processador de stream detecte. Você pode escolher pessoas, pacotes e animais de estimação, ou pessoas e pacotes. Os resultados da análise são enviados para seu bucket do Amazon S3 e nas notificações do Amazon SNS. Observe que o Amazon Rekognition Video detecta a presença de uma pessoa no vídeo, mas não detecta se a pessoa é uma pessoa específica. Para pesquisar um rosto de uma coleção em um vídeo de streaming, consulte the section called “Pesquisando faces em uma coleção em um vídeo de streaming”.

Para usar o Amazon Rekognition Video com streaming de vídeo, seu aplicativo exige o seguinte:

- Um stream de vídeo do Kinesis para enviar streaming de vídeo para o Amazon Rekognition Video. Para obter mais informações, consulte o Guia do desenvolvedor do Amazon Kinesis Video Streams.

- Um processador de stream do Amazon Rekognition Video para gerenciar a análise do streaming de vídeo. Para obter mais informações, consulte Visão geral das operações do processador de stream do Amazon Rekognition Video.
- Um bucket do Amazon S3. O Amazon Rekognition Video publica a saída da sessão no bucket do S3. A saída inclui o quadro da imagem em que uma pessoa ou objeto de interesse foi detectado pela primeira vez. Você deve ser o proprietário do bucket do S3.
- Um tópico do Amazon SNS para o qual o Amazon Rekognition Video publica alertas inteligentes e um resumo. end-of-session

Tópicos

- Configurando seus recursos do Amazon Rekognition Video e do Amazon Kinesis
- Chamar operações de detecção de rótulo para eventos de streaming de vídeo

Configurando seus recursos do Amazon Rekognition Video e do Amazon Kinesis

Os procedimentos a seguir descrevem as etapas a serem seguidas para provisionar o fluxo de vídeo do Kinesis e outros recursos usados para detectar rótulos em um streaming de vídeo.

Pré-requisitos

Para executar esse procedimento, AWS SDK para Java deve estar instalado. Para obter mais informações, consulte Comece a usar o Amazon Rekognition. A conta da AWS que você usa exige permissões de acesso à API do Amazon Rekognition. Para obter mais informações, consulte Ações definidas pelo Amazon Rekognition no Guia do usuário do IAM.

Para detectar rótulos em um stream de vídeo (AWS SDK)

1. Crie um bucket do Amazon S3. Anote o nome do bucket e os prefixos de chave que você queira usar. Você usa essas informações posteriormente.
2. Criar um tópico do Amazon SNS. Você pode usá-lo para receber notificações quando um objeto de interesse for detectado pela primeira vez no stream de vídeo. Anote o nome de recurso da Amazon (ARN) para o tópico. Para obter mais informações, consulte Criar um tópico do Amazon SNS no guia do desenvolvedor do Amazon SNS.
3. Assine um endpoint no tópico do Amazon SNS. Para obter mais informações, consulte Assinatura de um tópico do Amazon SNS no guia do desenvolvedor do Amazon SNS.

4. Crie um stream de vídeo do Kinesis e anote o nome de recurso da Amazon (ARN) do stream.
5. Se você ainda não o fez, crie um perfil de serviço do IAM para dar ao Amazon Rekognition Video acesse aos seus streams de vídeo do Kinesis, ao seu bucket do S3 e ao seu tópico do Amazon SNS. Para obter mais informações, consulte Fornecendo acesso para processadores de fluxo de detecção de rótulos.

Em seguida, você pode criar o processador de fluxo de detecção de rótulos e iniciar o processador de fluxo usando o nome de processador de fluxo que você escolheu.

> **Note**
>
> Inicie o processador de stream somente depois de verificar se você pode ingerir mídia no stream de vídeo do Kinesis.

Orientação e configuração da câmera

Os eventos de streaming de vídeo do Amazon Rekognition Video podem oferecer suporte a todas as câmeras compatíveis com o Kinesis Video Streams. Para obter melhores resultados, recomendamos colocar a câmera entre 0 e 45 graus do chão. A câmera precisa estar na posição vertical canônica. Por exemplo, se houver uma pessoa na moldura, a pessoa deve estar orientada verticalmente e a cabeça da pessoa deve estar mais alta na moldura do que os pés.

Fornecendo acesso para processadores de fluxo de detecção de rótulos

Você usa uma função de serviço AWS Identity and Access Management (IAM) para dar ao Amazon Rekognition Video acesso de leitura aos streams de vídeo do Kinesis. Para fazer isso, use os perfis do IAM para dar ao Amazon Rekognition Video acesso ao seu bucket do Amazon S3 e a um tópico do Amazon SNS.

Você pode criar uma política de permissões que permita ao Amazon Rekognition Video acessar um tópico existente do Amazon SNS, um bucket do Amazon S3 e um stream de vídeo do Kinesis. Para obter um step-by-step procedimento usando o AWS CLI, consultethe section called “AWS CLI comandos para configurar uma função IAM de detecção de rótulos”.

Para dar ao Amazon Rekognition Video acesso a recursos para detecção de rótulos

1. Crie uma nova política de permissões com o editor de políticas IAM JSON e use a política a seguir. Substitua `kvs-stream-name` pelo nome do stream de vídeo do Kinesis, `topicarn` pelo

nome do recurso da Amazon (ARN) do tópico do Amazon SNS que você deseja usar e `bucket-name` pelo nome do bucket do 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. Crie um perfil de serviço do IAM ou atualize um perfil de serviço do IAM existente. Use as informações a seguir para criar o perfil de serviço do IAM:

   1. Escolha Rekognition como o nome de serviço.
   2. Escolha Rekognition para o caso de uso da função de serviço.
   3. Anexe a política de permissões que você criou na etapa 1.

3. Anote o ARN da função de serviço. Você precisa dele para criar o processador de stream antes de realizar operações de análise de vídeo.

4. (Opcional) Se você usar sua própria AWS KMS chave para criptografar dados enviados ao seu bucket do S3, deverá adicionar a seguinte declaração com a função do IAM. (Esse é o perfil do IAM criado para a política de chave, que corresponde à chave gerenciada pelo cliente que você deseja usar.)

```
            {
                        "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 comandos para configurar uma função IAM de detecção de rótulos

Se você ainda não o fez, configure e configure o AWS CLI com suas credenciais.

Insira os comandos a seguir no AWS CLI para configurar uma função do IAM com as permissões necessárias para a detecção de rótulos.

1. `export IAM_ROLE_NAME=labels-test-role`
2. `export AWS_REGION=us-east-1`

3. Crie um arquivo de política de relacionamento de confiança (por exemplo, assume-role-rekognition .json) com o conteúdo a seguir.

```
{
  "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 o nome do tópico do SNS com o qual você deseja receber notificações não começar com o prefixo AmazonRekognition "", adicione a seguinte política:

   `aws iam attach-role-policy --role-name $IAM_ROLE_NAME --policy-arn "arn:aws:iam::aws:policy/AmazonSNSFullAccess" --region $AWS_REGION`
7. Se você usa sua própria chave do AWS KMS para criptografar dados enviados para seu bucket do Amazon S3, atualize a política de chave da chave gerenciada pelo cliente que você deseja usar.

   a. Crie um arquivo kms_key_policy.json que contenha o seguinte conteúdo:

```
{
"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`. Substitua KMS_KEY_ID pelo ID da chave KMS que você criou.

c. `aws kms put-key-policy --policy-name default --key-id $KMS_KEY_ID --policy file://path-to-kms-key-policy.json`

Chamar operações de detecção de rótulo para eventos de streaming de vídeo

O Amazon Rekognition Video pode detectar pessoas ou objetos relevantes em um streaming de vídeo e notificar você quando eles forem detectados. Ao criar um processador de stream de detecção de etiquetas, escolha quais etiquetas você deseja que o Amazon Rekognition Video detecte. Podem ser pessoas, pacotes e animais de estimação, ou pessoas, pacotes e animais de estimação. Escolha somente os rótulos específicos que você deseja detectar. Dessa forma, os únicos rótulos relevantes criam notificações. Você pode configurar opções para determinar quando armazenar informações de vídeo e, em seguida, fazer um processamento adicional com base nos rótulos detectados no quadro.

Depois de configurar seus recursos, o processo para detectar rótulos em um streaming de vídeo é o seguinte:

1. Crie o processador de stream
2. Inicie o processador de streaming
3. Se um objeto de interesse for detectado, você receberá uma notificação do Amazon SNS para a primeira ocorrência de cada objeto de interesse.
4. O processador de stream é interrompido quando o tempo especificado em `MaxDurationInSeconds` é concluído.
5. Você recebe uma notificação final do Amazon SNS com um resumo do evento.
6. O Amazon Rekognition Video publica um resumo detalhado da sessão em seu bucket do S3.

Tópicos

- Criação do processador de stream de detecção de rótulos do Amazon Rekognition Video
- Iniciando o processador de fluxo de detecção de rótulos do Amazon Rekognition Video
- Analisando os resultados da detecção de etiquetas

Criação do processador de stream de detecção de rótulos do Amazon Rekognition Video

Antes de poder analisar um streaming de vídeo, você cria um processador de streaming de vídeo Amazon Rekognition Video (). CreateStreamProcessor

Se você quiser criar um processador de fluxo para detectar rótulos de interesse e pessoas, forneça como entrada um fluxo de vídeo do Kinesis (`Input`), informações de bucket do Amazon S3 (`Output`) e um ARN de tópico do Amazon SNS (`StreamProcessorNotificationChannel`). Você também pode fornecer um ID de chave KMS para criptografar os dados enviados para seu bucket do S3. Você especifica o que deseja detectar `Settings`, como pessoas, pacotes e pessoas, ou animais de estimação, pessoas e pacotes. Você também pode especificar em que parte do quadro você deseja que o Amazon Rekognition monitore com `RegionsOfInterest`. Veja a seguir um exemplo de JSON para a solicitação `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
        }
      ]
    }
  ]
}
```

Observe que você pode alterar o valor `MinConfidence` ao especificar o `ConnectedHomeSettings` para o processador de stream. `MinConfidence` é um valor numérico que varia de 0 a 100 que indica a certeza do algoritmo sobre suas previsões. Por exemplo, uma notificação `person` com um valor de confiança de 90 significa que o algoritmo tem certeza absoluta de que a pessoa está presente no vídeo. Um valor de confiança de 10 indica que pode haver uma pessoa. Você pode definir o valor `MinConfidence` desejado de sua escolha entre 0 e 100, dependendo da frequência com que deseja ser notificado. Por exemplo, se você quiser ser notificado somente quando o Rekognition tiver certeza absoluta de que há um pacote no quadro do vídeo, você pode definir de `MinConfidence` para 90.

Por padrão, `MinConfidence` está definido como 50. Se quiser otimizar o algoritmo para maior precisão, você pode definir `MinConfidence` como maior que 50. Em seguida, você recebe menos notificações, mas cada notificação é mais confiável. Se você quiser otimizar o algoritmo para uma maior recuperação, defina `MinConfidence` como menor que 50 para receber mais notificações.

Iniciando o processador de fluxo de detecção de rótulos do Amazon Rekognition Video

Inicie a análise de streaming de vídeo chamando StartStreamProcessor com o nome do processador de fluxo que você especificou em `CreateStreamProcessor`. Ao executar a operação `StartStreamProcessor` em um processador de fluxo de detecção de etiquetas, você insere as informações de início e término para determinar o tempo de processamento.

Quando você inicia o processador de stream, o estado do processador de stream de detecção de etiquetas muda das seguintes formas:

1. Quando você liga `StartStreamProcessor`, o estado do processador de fluxo de detecção de rótulos vai de `STOPPED` ou `FAILED` para `STARTING`.
2. Enquanto o processador de fluxo de detecção de etiquetas é executado, ele permanece em `STARTING`.
3. Quando o processador de fluxo de detecção de etiquetas termina de funcionar, o estado se torna `STOPPED` ou `FAILED`.

O `StartSelector` especifica o ponto de partida no stream do Kinesis para iniciar o processamento. Você pode usar o carimbo de data/hora do produtor KVS ou o número do fragmento KVS. Para obter mais informações, consulte Fragmento.

> **Note**
>
> Se você usar o carimbo de data/hora do KVS Producer, deverá inserir a hora em milissegundos.

O `StopSelector` especifica quando parar de processar o fluxo. Você pode especificar um tempo máximo para processar o vídeo. O padrão é uma duração máxima de 10 segundos. Observe que o tempo real de processamento pode ser um pouco maior do que a duração máxima, dependendo do tamanho dos fragmentos individuais do KVS. Se a duração máxima tiver sido atingida ou excedida no final de um fragmento, o tempo de processamento será interrompido.

Veja a seguir um exemplo de JSON para a solicitação `StartStreamProcessor`.

```
{
   "Name": "string",
   "StartSelector": {
     "KVSStreamStartSelector": {
         "KVSProducerTimestamp": 1655930623123
      },
        "StopSelector": {
            "MaxDurationInSeconds": 11
      }
   }
}
```

Se o processador de stream for iniciado com sucesso, uma resposta HTTP 200 será retornada. Um corpo JSON vazio está incluído.

Analisando os resultados da detecção de etiquetas

Há três maneiras pelas quais o Amazon Rekognition Video publica notificações de um processador de fluxo de detecção de etiquetas: notificações do Amazon SNS para eventos de detecção de objetos, uma notificação end-of-session do Amazon SNS para um resumo e um relatório detalhado do bucket do Amazon S3.

- Notificações do Amazon SNS para eventos de detecção de objetos.

Se rótulos forem detectados no stream de vídeo, você receberá notificações do Amazon SNS para eventos de detecção de objetos. O Amazon Rekognition publica uma notificação na primeira vez em que uma pessoa ou objeto de interesse é detectado no stream de vídeo. As notificações incluem informações como o tipo de rótulo detectado, a confiança e um link para a imagem do herói. Eles também incluem uma imagem recortada da pessoa ou objeto detectado e um carimbo de data e hora da detecção. A notificação tem o seguinte formato:

```
{"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
    }
}
```

- Resumo do Amazon SNS. end-of-session

  Você também recebe uma notificação do Amazon SNS quando a sessão de processamento do stream é concluída. Essa notificação lista os metadados da sessão. Isso inclui detalhes como a duração do fluxo que foi processado. A notificação tem o seguinte formato:

```
{"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
    }
}
```

- Relatório do bucket do Amazon S3.

  O Amazon Rekognition Video publica resultados de inferência detalhados de uma operação de análise de vídeo no bucket do Amazon S3 que é fornecido na operação `CreateStreamProcessor`. Esses resultados incluem molduras de imagem em que um objeto de interesse ou pessoa foi detectado pela primeira vez.

Os quadros estão disponíveis no S3 no seguinte caminho: ObjectKeyPrefix// StreamProcessorNameSessionId/*service_determined_unique_path*. Nesse caminho, LabelKeyPrefixé um argumento opcional fornecido pelo cliente, StreamProcessorNameé o nome do recurso do processador de stream e SessionIdé uma ID exclusiva para a sessão de processamento de stream. Substitua-os de acordo com sua situação.

Detectar rótulos personalizados

Os Amazon Rekognition Custom Labels podem identificar objetos e cenas em imagens que são específicas para suas necessidades comerciais, como logotipos ou peças de máquinas de engenharia. Para obter mais informações, consulte O que são Amazon Rekognition Custom Labels? no Guia do desenvolvedor do Amazon Rekognition Custom Labels.

Detectando e analisanda faces

O Amazon Rekognition fornece informações que APIs você pode usar para detectar e analisar rostos em imagens e vídeos. Esta seção apresenta uma visão geral dos pacotes de detecção facial dos conceitos de detecção facial e comparação de faces. Essas operações incluem funcionalidades como detectar pontos de referência faciais, analisar emoções e comparar rostos.

O Amazon Rekognition pode identificar pontos de referência faciais (por exemplo, posição dos olhos), detectar emoções (por exemplo, felicidade ou tristeza) e outros atributos (por exemplo, presença de óculos, oclusão facial). Quando um rosto é detectado, o sistema analisa os atributos faciais e retorna uma pontuação de confiança para cada atributo.



Esta seção traz exemplos de operações de imagem e vídeo.

Para obter mais informações sobre o uso das operações de imagem do Rekognition, consulte Como trabalhar com imagens.

Para obter mais informações sobre o uso das operações de vídeo do Rekognition, consulte Trabalhar com operações de análise de vídeo armazenado.

Observe que essas operações não são operações de armazenamento. Você pode usar as operações de armazenamento e coleções de Face para salvar os metadados faciais de faces detectadas em uma imagem. Posteriormente, você pode pesquisar faces armazenadas em imagens e vídeos. Por exemplo, isso permite pesquisar uma pessoa específica em um vídeo. Para obter mais informações, consulte Pesquisa de faces em uma coleção.

Para obter mais informações, consulte a seção Faces do Amazon FAQs Rekognition.

> **Note**
>
> Os modelos de detecção facial usados pelo Amazon Rekognition Image e pelo Amazon Rekognition Video não suportam a detecção de faces em personagens animados ou de desenhos animados ou entidades não humanas. Se você quiser detectar personagens de desenhos animados em imagens ou vídeos, recomendamos o uso de Amazon Rekognition Custom Labels. Para obter mais informações, consulte o Guia do desenvolvedor de Amazon Rekognition Custom Labels.

Tópicos

- Visão geral da detecção facial e comparação de faces
- Diretrizes sobre atributos faciais
- Detectanda faces em uma imagem
- Comparanda faces em imagens
- Detectanda faces em um vídeo armazenado

Visão geral da detecção facial e comparação de faces

O Amazon Rekognition fornece aos usuários acesso a dois aplicativos principais de machine learning para imagens contendo rostos: detecção facial e comparação facial. Eles potencializam atributos cruciais, como análise facial e verificação de identidade, tornando-os vitais para vários aplicativos, desde segurança até organização de fotos pessoais.

Detecção facial

Um sistema de detecção facial responde à pergunta: “Há uma face nesta foto?” Os principais aspectos da detecção facial incluem:

- Localização e orientação: determina a presença, a localização, a escala e a orientação das faces em imagens ou quadros de vídeo.
- Atributos faciais: detecta rostos independentemente de atributos como sexo, idade ou pelos faciais.
- Informações adicionais: dão detalhes sobre oclusão facial e direção do olhar.

Comparação facial

Um sistema de comparação facial foca a pergunta: “A face em uma imagem corresponde a uma face em outra imagem?” As funcionalidades do sistema de comparação facial incluem:

- Previsões de correspondência facial: compara um rosto em uma imagem com um rosto em um banco de dados fornecido para prever correspondências.
- Manipulação de atributos faciais: manipula atributos para comparar faces, independentemente da expressão, dos pelos faciais e da idade.

Pontuações de confiança e detecções não realizadas

Os sistemas de detecção e comparação facial utilizam pontuações de confiança. Uma pontuação de confiança indica a probabilidade de previsões, como a presença de uma face ou uma correspondência entre faces. Pontuações mais altas indicam maior probabilidade. Por exemplo, 90% de confiança sugere uma probabilidade maior de detecção ou correspondência correta do que 60%.

Se um sistema de detecção facial não detectar adequadamente um rosto ou fornecer uma previsão de baixa confiança para um rosto real, isso é um 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 perdido.

Da mesma forma, um sistema de comparação facial pode não corresponder a duas faces que pertencem à mesma pessoa (detection/false negative) or may incorrectly predict that two faces from different people are the same person (false alarm/falsepositivo perdido).

Design da aplicação e definição de limites

- Você pode definir um limite que especifique o nível mínimo de confiança necessário para retornar um resultado. Escolher os limites de confiança adequados é essencial para o design da aplicação e para a tomada de decisões com base nas saídas do sistema.
- Os limites devem refletir seu caso de uso. Alguns exemplos de casos de uso e limites de confiança:

- • Aplicativos de fotos: um limite mais baixo (por exemplo, 80%) pode ser suficiente para identificar familiares em fotos.
  - Cenários de alto risco: em casos de uso em que o risco de detecção não realizada ou alarme falso é maior, como aplicações de segurança, o sistema deve usar um nível de confiança mais alto. Nesses casos, um limite mais alto (por exemplo, 99%) é recomendado para combinações faciais precisas.

Para obter mais informações sobre configuração e entender os limites de confiança, consulte Pesquisa de faces em uma coleção.

Diretrizes sobre atributos faciais

Aqui estão detalhes sobre como o Amazon Rekognition processa e retorna atributos faciais.

- FaceDetail Objeto: Para cada face detectada, um FaceDetail objeto é retornado. FaceDetail Ele contém dados sobre pontos turísticos faciais, qualidade, pose e muito mais.
- Previsões de atributos: atributos como emoção, sexo, idade e outros são previstos. Um nível de confiança é atribuído para cada previsão, e as previsões são retornadas com a respectiva pontuação de confiança. Um limite de confiança de 99% é recomendado para casos de uso sensíveis. Para estimativa de idade, o ponto médio da faixa etária prevista oferece a melhor aproximação.

Observe que as previsões de gênero e emoção são baseadas na aparência física e não devem ser usadas para determinar a identidade real de gênero ou o estado emocional. Uma previsão binária de gênero (masculino/feminino) é baseada na aparência física de uma face em uma imagem específica. Isso não indica a identidade de gênero de uma pessoa, e você não deve usar o Rekognition para fazer essa determinação. Não recomendamos o uso de previsões binárias de gênero para tomar decisões que afetam os direitos, a privacidade ou o acesso de um indivíduo aos serviços. De modo similar, uma previsão de emoção não indica o estado emocional interno real de uma pessoa, e você não deve usar o Rekognition para fazer essa determinação. Uma pessoa fingindo estar com o rosto feliz em uma foto pode parecer feliz, mas não estar sentindo felicidade.

Aplicação e casos de uso

Aqui estão algumas aplicações práticas e casos de uso desses atributos:

- Aplicações: atributos como sorriso, pose e nitidez podem ser utilizados para selecionar fotos de perfil ou estimar dados demográficos de modo anônimo.
- Casos de uso comuns: aplicações de redes sociais e estimativa demográfica em eventos ou lojas de varejo são exemplos típicos.

Para obter informações mais detalhadas sobre cada atributo, consulte FaceDetail.

Detectanda faces em uma imagem

O Amazon Rekognition Image DetectFacesfornece a operação que procura as principais características faciais, como olhos, nariz e boca, para detectar rostos em uma imagem de entrada. O Amazon Rekognition Image detecta as 100 maiores faces em uma imagem.

Você pode fornecer a imagem de entrada como uma matriz de bytes de imagem (bytes de imagem codificados em base64) ou especificar um objeto Amazon S3. Neste procedimento, você carrega uma imagem (JPEG ou PNG) no bucket do S3 e especifica o nome da chave do objeto.

Como detectar faces em uma imagem

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Carregue uma imagem (que contenha uma ou mais faces) no bucket do S3.

   Para obter instruções, consulte Como fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

3. Use os exemplos a seguir para chamar `DetectFaces`.

   Java

   Este exemplo exibe a faixa etária estimada para faces detectadas e lista o JSON de todos os atributos faciais detectados. Altere o valor de `photo` para o nome de arquivo da imagem. Altere o valor de `amzn-s3-demo-bucket` para o bucket do Amazon S3 em que a imagem está armazenada.

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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
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

Este exemplo exibe a saída JSON da `detect-faces` AWS CLI operação. Substitua `file` pelo nome de um arquivo de imagem. Substitua `amzn-s3-demo-bucket` pelo nome do bucket do Amazon S3 que contém o arquivo de imagem.

```
aws rekognition detect-faces --image "{"S3Object":{"Bucket":"amzn-s3-demo-
bucket,"Name":"image-name"}}"\
```

```
                                --attributes "ALL" --profile profile-name --region
 region-name
```

Se você estiver acessando a CLI em um dispositivo Windows, use aspas duplas em vez de aspas simples e escape das aspas duplas internas com barra invertida (ou seja, \) para resolver quaisquer erros de analisador que você possa encontrar. Para obter um exemplo, veja o seguinte:

```
aws rekognition detect-faces --image "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --attributes "ALL"
--profile profile-name --region region-name
```

Python

Este exemplo exibe a faixa etária estimada e outros atributos das faces detectados e lista o JSON para todos os atributos faciais detectados. Altere o valor de `photo` para o nome de arquivo da imagem. Altere o valor de `amzn-s3-demo-bucket` para o bucket do Amazon S3 em que a imagem está armazenada. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
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

Este exemplo exibe a faixa etária estimada para faces detectadas e lista o JSON de todos os atributos faciais detectados. Altere o valor de `photo` para o nome de arquivo da imagem. Altere o valor de `amzn-s3-demo-bucket` para o bucket do Amazon S3 em que a imagem está armazenada.

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

Este exemplo exibe a faixa etária estimada para faces detectadas e lista vários atributos faciais. Altere o valor de `photo` para o nome de arquivo da imagem. Altere o valor de `amzn-s3-demo-bucket` para o bucket do Amazon S3 em que a imagem está armazenada.

```
# 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

Este exemplo exibe a faixa etária estimada para faces detectadas e lista vários atributos faciais. Altere o valor de `photo` para o nome de arquivo da imagem. Altere o valor de `amzn-s3-demo-bucket` para o bucket do Amazon S3 em que a imagem está armazenada.

Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

Se você estiver usando TypeScript definições, talvez seja necessário usar `import AWS from 'aws-sdk'` em vez de`const AWS = require('aws-sdk')`, para executar o programa com o Node.js. Você pode consultar o AWS SDK para Javascript para obter mais detalhes. Dependendo de como você configurou suas configurações, talvez você também precise especificar sua região com `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 solicitação de operação

A entrada de `DetectFaces` é uma imagem. Neste exemplo, a imagem é carregada de um bucket do Amazon S3. O parâmetro `Attributes` especifica que todos os atributos faciais devem ser retornados. Para obter mais informações, consulte Como trabalhar com imagens.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "input.jpg"
        }
    },
    "Attributes": [
        "ALL"
    ]
}
```

DetectFaces resposta da operação

`DetectFaces` retorna as seguintes informações para cada fase detectada:

- Caixa delimitadora – As coordenadas da caixa delimitadora que circunda a face.
- Confiança – O nível de confiança da caixa delimitadora que contém uma face.
- Pontos de referência faciais — Uma variedade de pontos de referência faciais. Para cada ponto de referência (como olho esquerdo, olho direito e boca), a resposta fornece as coordenadas x e y.
- Atributos faciais — Um conjunto de atributos faciais, como se a face está ocluído, retornado como um objeto `FaceDetail`. O conjunto inclui: barba AgeRange, emoções, óculos EyeDirection, sexo, bigode EyesOpen FaceOccluded MouthOpen, sorriso e óculos de sol. Para cada atributo, a resposta fornece um valor. O valor pode ser de diferentes tipos, como um tipo booleano (se uma pessoa está usando óculos escuros), uma cadeia de caracteres (se a pessoa é homem ou mulher) ou um valor de grau angular (para inclinação/arrasto das direções do olhar). Além disso, para a maioria dos atributos, a resposta também dá confiança no valor detectado para o atributo. Observe que, embora EyeDirection os atributos FaceOccluded e sejam compatíveis com o uso`DetectFaces`, eles não são compatíveis com a análise de vídeos com `StartFaceDetection` `GetFaceDetection` e.

- Qualidade – Descreve o brilho e a nitidez da face. Para obter mais informações sobre como garantir a melhor detecção de faces possível, consulte Recomendações para imagens de entrada de comparação facial.
- Pose – Descreve a rotação da face dentro da imagem.

A solicitação pode representar uma série de atributos faciais que você deseja que sejam retornados. Um subconjunto de atributos faciais `DEFAULT` - `BoundingBox`, `Confidence`, `Pose` `Quality` e `Landmarks`, sempre será retornado. Você pode solicitar o retorno de atributos faciais específicos (além da lista padrão) usando `["DEFAULT", "FACE_OCCLUDED", "EYE_DIRECTION"]` ou apenas um atributo, como `["FACE_OCCLUDED"]`. Você pode solicitar todos os atributos faciais usando `["ALL"]`. Solicitar mais atributos pode aumentar o tempo de resposta.

A seguir está uma resposta de exemplo de uma chamada de 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
  }
}
```


Observe o seguinte:

- Os dados `Pose` descrevem a rotação da face detectada. Você pode usar a combinação dos dados `BoundingBox` e `Pose` para desenhar a caixa delimitadora em torno de faces exibidas pelo aplicativo.
- O `Quality` descreve o brilho e a nitidez da face. Você pode achar útil comparar faces em imagens diferentes e encontrar a melhor.
- A resposta anterior mostra todos os `landmarks` faciais que o serviço pode detectar, todos os atributos faciais e as emoções. Para ter todos esses na resposta, você deve especificar o parâmetro `attributes` com valor `ALL`. Por padrão, a API `DetectFaces` retorna apenas os cinco atributos faciais a seguir: `BoundingBox`, `Confidence`, `Pose`, `Quality` e `landmarks`. Os pontos de referência padrão retornados são: `eyeLeft`, `eyeRight`, `nose`, `mouthLeft` e `mouthRight`.

Comparanda faces em imagens

Com o Rekognition, você pode comparar faces entre duas imagens usando a operação. CompareFaces Esse atributo é útil para aplicações como verificação de identidade ou correspondência de fotos.

CompareFaces compara uma face na imagem de origem com cada face na imagem de destino. As imagens são CompareFaces passadas para:

- Representação de uma imagem codificada em base64.
- Objetos do Amazon S3.

Detecção facial versus comparação facial

A comparação facial é diferente da detecção facial. A detecção facial (que usa DetectFaces) identifica apenas a presença e a localização de rostos em uma imagem ou vídeo. Por outro lado, a comparação facial envolve comparar uma face detectada em uma imagem de origem com faces em uma imagem de destino para a descoberta de correspondências.

Limites de similaridade

Use o parâmetro `similarityThreshold` para definir o nível mínimo de confiança para que as correspondências sejam incluídas na resposta. Por padrão, apenas faces com uma pontuação de similaridade maior ou igual a 80% são retornadas na resposta.

> **Note**
>
> O `CompareFaces` usa algoritmos de machine learning (ML), que são probabilísticos. Um falso negativo é uma previsão incorreta de que uma face na imagem alvo tem uma pontuação de confiança de similaridade baixa quando comparada à face na imagem de origem. Para reduzir a probabilidade de falsos negativos, recomendamos que você compare a imagem de destino com várias imagens de origem. Se você planeja usar `CompareFaces` para tomar uma decisão que afete os direitos, a privacidade ou o acesso a serviços de um indivíduo, recomendamos que você passe o resultado a um humano para análise e validação adicional antes de agir.

Os exemplos de código a seguir demonstram como usar as CompareFaces operações para vários AWS SDKs. No AWS CLI exemplo, você carrega duas imagens JPEG no seu bucket do Amazon S3 e especifica o nome da chave do objeto. Nos outros exemplos, você carrega dois arquivos do sistema de arquivos local e os insere como matrizes de bytes de imagens.

Para comparar faces

1. Se ainda não tiver feito isso:

a. Crie ou `AmazonRekognitionFullAccess` atualize um usuário com permissões `AmazonS3ReadOnlyAccess` (somente AWS CLI por exemplo). Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use o código de exemplo a seguir para chamar a operação `CompareFaces`.

Java

Este exemplo exibe informações sobre faces correspondentes em imagens de origem e destino carregadas do sistema de arquivos local.

Substitua os valores de `sourceImage` e `targetImage` pelo caminho e pelo nome de arquivo das imagens de origem e destino.

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

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.CompareFacesMatch;
import com.amazonaws.services.rekognition.model.CompareFacesRequest;
import com.amazonaws.services.rekognition.model.CompareFacesResult;
import com.amazonaws.services.rekognition.model.ComparedFace;
import java.util.List;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import com.amazonaws.util.IOUtils;

public class CompareFaces {

   public static void main(String[] args) throws Exception{
       Float similarityThreshold = 70F;
       String sourceImage = "source.jpg";
       String targetImage = "target.jpg";
```

```
      ByteBuffer sourceImageBytes=null;
      ByteBuffer targetImageBytes=null;

      AmazonRekognition rekognitionClient =
AmazonRekognitionClientBuilder.defaultClient();

      //Load source and target images and create input parameters
      try (InputStream inputStream = new FileInputStream(new
File(sourceImage))) {
         sourceImageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
      }
      catch(Exception e)
      {
          System.out.println("Failed to load source image " + sourceImage);
          System.exit(1);
      }
      try (InputStream inputStream = new FileInputStream(new
File(targetImage))) {
          targetImageBytes =
ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
      }
      catch(Exception e)
      {
          System.out.println("Failed to load target images: " + targetImage);
          System.exit(1);
      }

      Image source=new Image()
           .withBytes(sourceImageBytes);
      Image target=new Image()
           .withBytes(targetImageBytes);

      CompareFacesRequest request = new CompareFacesRequest()
              .withSourceImage(source)
              .withTargetImage(target)
              .withSimilarityThreshold(similarityThreshold);

      // Call operation
      CompareFacesResult
compareFacesResult=rekognitionClient.compareFaces(request);


      // Display results
```

```
        List <CompareFacesMatch> faceDetails =
 compareFacesResult.getFaceMatches();
        for (CompareFacesMatch match: faceDetails){
          ComparedFace face= match.getFace();
          BoundingBox position = face.getBoundingBox();
          System.out.println("Face at " + position.getLeft().toString()
                + " " + position.getTop()
                + " matches with " + match.getSimilarity().toString()
                + "% confidence.");

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

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

Java V2

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
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

Este exemplo exibe a saída JSON da `compare-faces` AWS CLI operação.

Substitua `amzn-s3-demo-bucket` pelo nome do bucket do Amazon S3 que contém as imagens de origem e de destino. Substitua `source.jpg` e `target.jpg` pelos nomes de arquivo das imagens de origem e destino.

```
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 você estiver acessando a CLI em um dispositivo Windows, use aspas duplas em vez de aspas simples e escape das aspas duplas internas com barra invertida (ou seja, \) para resolver quaisquer erros de analisador que você possa encontrar. Para obter um exemplo, veja o seguinte:

```
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

Este exemplo exibe informações sobre faces correspondentes em imagens de origem e destino carregadas do sistema de arquivos local.

Substitua os valores de `source_file` e `target_file` pelo caminho e pelo nome de arquivo das imagens de origem e destino. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Este exemplo exibe informações sobre faces correspondentes em imagens de origem e destino carregadas do sistema de arquivos local.

Substitua os valores de `sourceImage` e `targetImage` pelo caminho e pelo nome de arquivo das imagens de origem e destino.

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

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

public class CompareFaces
{
    public static void Example()
    {
        float similarityThreshold = 70F;
        String sourceImage = "source.jpg";
        String targetImage = "target.jpg";

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        Amazon.Rekognition.Model.Image imageSource = new
 Amazon.Rekognition.Model.Image();
        try
        {
            using (FileStream fs = new FileStream(sourceImage, FileMode.Open,
 FileAccess.Read))
            {
                byte[] data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageSource.Bytes = new MemoryStream(data);
            }
        }
        catch (Exception)
        {
```

```
            Console.WriteLine("Failed to load source image: " + sourceImage);
            return;
        }

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

        CompareFacesRequest compareFacesRequest = new CompareFacesRequest()
        {
            SourceImage = imageSource,
            TargetImage = imageTarget,
            SimilarityThreshold = similarityThreshold
        };

        // Call operation
        CompareFacesResponse compareFacesResponse =
rekognitionClient.CompareFaces(compareFacesRequest);

        // Display results
        foreach(CompareFacesMatch match in compareFacesResponse.FaceMatches)
        {
            ComparedFace face = match.Face;
            BoundingBox position = face.BoundingBox;
            Console.WriteLine("Face at " + position.Left
                  + " " + position.Top
                  + " matches with " + match.Similarity
                  + "% confidence.");
        }
```

```
        Console.WriteLine("There was " +
 compareFacesResponse.UnmatchedFaces.Count + " face(s) that did not match");

    }
}
```

Ruby

Este exemplo exibe informações sobre faces correspondentes em imagens de origem e destino carregadas do sistema de arquivos local.

Substitua os valores de `photo_source` e `photo_target` pelo caminho e pelo nome de arquivo das imagens de origem e destino.

```
 # Add to your Gemfile
  # gem 'aws-sdk-rekognition'
  require 'aws-sdk-rekognition'
  credentials = Aws::Credentials.new(
     ENV['AWS_ACCESS_KEY_ID'],
     ENV['AWS_SECRET_ACCESS_KEY']
  )
  bucket        = 'bucket' # the bucketname without s3://
  photo_source  = 'source.jpg'
  photo_target  = 'target.jpg'
  client   = Aws::Rekognition::Client.new credentials: credentials
  attrs = {
    source_image: {
      s3_object: {
        bucket: bucket,
        name: photo_source
      },
    },
    target_image: {
      s3_object: {
        bucket: bucket,
        name: photo_target
      },
    },
    similarity_threshold: 70
  }
  response = client.compare_faces attrs
  response.face_matches.each do |face_match|
```

```
    position   = face_match.face.bounding_box
    similarity = face_match.similarity
    puts "The face at: #{position.left}, #{position.top} matches with
 #{similarity} % confidence"
  end
```

Node.js

Este exemplo exibe informações sobre faces correspondentes em imagens de origem e destino carregadas do sistema de arquivos local.

Substitua os valores de `photo_source` e `photo_target` pelo caminho e pelo nome de arquivo das imagens de origem e destino. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
// 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 solicitação de operação

A entrada de `CompareFaces` é uma imagem. Neste exemplo, as imagens de origem e destino carregadas do sistema de arquivos local. O parâmetro de entrada `SimilarityThreshold` especifica o mínimo de confiança que comparou faces que precisa corresponder para ser incluído na resposta. Para obter mais informações, consulte Como trabalhar com imagens.

```
{
    "SourceImage": {
        "Bytes": "/9j/4AAQSk2Q==..."
    },
    "TargetImage": {
        "Bytes": "/9j/4O1Q==..."
    },
    "SimilarityThreshold": 70
}
```

CompareFaces resposta da operação

A resposta inclui:

- Uma matriz de correspondências de faces: uma lista de faces correspondentes com pontuações de similaridade e metadados para cada face correspondente. Se várias faces coincidirem, a matriz `faceMatches`

  incluirá todas as combinações de faces.
- Detalhes da correspondência facial: cada face combinada também fornece uma caixa delimitadora, valor de confiança, locais de pontos de referência e pontuação de similaridade.

- Uma lista de faces incomparáveis: a resposta também inclui faces da imagem de destino que não correspondem à face da imagem de origem. Inclui uma caixa delimitadora para cada rosto incomparável.
- Informações da face de origem: inclui informações sobre a face da imagem de origem que foi usada na comparação, incluindo a caixa delimitadora e o valor de confiança.

O exemplo mostra que uma correspondência de face foi encontrada na imagem de destino. Para essa correspondência facial, ele fornece uma caixa delimitadora e um valor de confiança (o nível de confiança que o Amazon Rekognition tem de que a caixa delimitadora contém uma face). A pontuação de similaridade de 99,99 indica a semelhança das faces. O exemplo também mostra uma face que o Amazon Rekognition encontrou na imagem de destino que não correspondia à face que foi analisada na imagem de origem.


```
{
    "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
    }
}
```


Detectanda faces em um vídeo armazenado

O Amazon Rekognition Video pode detectar faces em vídeos armazenados em um bucket do Amazon S3 e fornecer informações como:

- As horas em que as faces são detectadas em um vídeo.
- O local das faces no quadro do vídeo na hora em que foram detectadas.
- Pontos de referência faciais, como a posição do olho esquerdo.

- Atributos adicionais, conforme explicado na página the section called “Diretrizes sobre atributos faciais”.

A detecção facial do Amazon Rekognition Video em vídeos armazenados é uma operação assíncrona. Para iniciar a detecção de rostos em vídeos, ligue StartFaceDetection. O Amazon Rekognition Video publica o status de conclusão da análise de vídeo em um tópico do Amazon Simple Notification Service (Amazon SNS). Se a análise do vídeo for bem-sucedida, você poderá chamar GetFaceDetection para obter os resultados da análise do vídeo. Para obter mais informações sobre como iniciar uma análise de vídeo e obter os resultados, consulte Chamando as operações de vídeo do Amazon Rekognition Video.

Esse procedimento expande o código em Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK), que usa uma fila do Amazon Simple Queue Service (Amazon SQS) para obter o status de conclusão de uma solicitação de análise de vídeo.

Para detectar faces em um vídeo armazenado em um bucket do Amazon S3 (SDK)

1. Execute Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK).
2. Adicione o código a seguir à classe `VideoDetect` criada por você na etapa 1.

   AWS CLI

   - No exemplo de código a seguir, altere `amzn-s3-demo-bucket` e `video-name` para o nome do bucket e do arquivo do Amazon S3 que você especificou na etapa 2.
   - Altere `region-name` para a região da AWS que você está usando. Substitua o valor de `profile_name` com o nome do seu perfil de desenvolvedor.
   - Mude `TopicARN` para o ARN do tópico do Amazon SNS que você criou na etapa 3 do Configuração do Amazon Rekognition Video.
   - Mude `RoleARN` para o ARN do perfil de serviço do IAM que você criou na etapa 7 do Configuração do Amazon Rekognition Video.

```
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 você estiver acessando a CLI em um dispositivo Windows, use aspas duplas em vez de aspas simples e escape das aspas duplas internas com barra invertida (ou seja, \) para resolver quaisquer erros de analisador que você possa encontrar. Para obter um exemplo, veja o seguinte:

```
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
```

Depois de executar a operação `StartFaceDetection` e obter o número de identificação do trabalho, execute a operação `GetFaceDetection` a seguir e forneça o número de identificação do trabalho:

```
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);


}
```

Na função `main`, substitua as linhas:

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

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

por:

```
        StartFaceDetection(amzn-s3-demo-bucket, video);

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

Java V2

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
//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
```

Na função `main`, substitua as linhas:

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

por:

```
    analyzer.StartFaceDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetFaceDetectionResults()
```

> **Note**
>
> Se você já tiver executado um exemplo de vídeo diferente de Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK), o nome da função a ser substituída será outro.

3. Execute o código. As informações sobre as faces detectadas no vídeo são mostradas.

GetFaceDetection resposta da operação

O `GetFaceDetection` retorna uma matriz (`Faces`) que contém informações sobre as faces detectadas no vídeo. Existe um elemento da matriz, FaceDetection, para cada vez que uma face é detectada no vídeo. Os elementos da matriz são retornados classificados por hora, em milissegundos desde o início do vídeo.

O exemplo a seguir é a resposta parcial do JSON em `GetFaceDetection`. Na resposta, observe o seguinte:

- Caixa delimitadora – As coordenadas da caixa delimitadora que circunda a face.
- Confiança – O nível de confiança da caixa delimitadora que contém uma face.
- Pontos de referência faciais — Uma variedade de pontos de referência faciais. Para cada ponto de referência (como olho esquerdo, olho direito e boca), a resposta fornece as coordenadas `x` e `y`.
- Atributos faciais — Um conjunto de atributos faciais AgeRange, que inclui: barba, emoções, óculos, sexo, bigode EyesOpen, sorriso MouthOpen e óculos de sol. O valor pode ser de tipos diferentes, como um tipo booliano (se a pessoa está usando óculos de sol) ou uma sequência (se a pessoa é do sexo masculino ou feminino). Além disso, para a maioria dos atributos, a resposta também dá confiança no valor detectado para o atributo. Observe que, embora EyeDirection os atributos FaceOccluded e sejam compatíveis com o uso`DetectFaces`, eles não são compatíveis com a análise de vídeos com `StartFaceDetection` `GetFaceDetection` e.
- Carimbo de data e hora — A hora em que a face foi detectado no vídeo.
- Informações de paginação — O exemplo mostra uma página de informações de detecção facial. Você pode especificar quantos elementos de pessoas retornar no parâmetro de entrada `MaxResults` para `GetFaceDetection`. Se existirem mais resultados além de `MaxResults`, o `GetFaceDetection` retornará um token (`NextToken`) usado para obter a próxima página de resultados. Para obter mais informações, consulte Obter os resultados da análise do Amazon Rekognition Video.
- Informações de vídeo – A resposta inclui informações sobre o formato do vídeo (`VideoMetadata`) em cada página de informações retornada pelo `GetFaceDetection`.
- Qualidade – Descreve o brilho e a nitidez da face.
- Pose — Descreve a rotação da face.

```
{
    "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
    }
}
```

Pesquisa de faces em uma coleção

O Amazon Rekognition permite que você use uma face de entrada para pesquisar correspondências em uma coleção de faces armazenadas. Você começa armazenando informações sobre faces detectadas em contêineres do lado do servidor chamados "coleções". As coleções armazenam faces e usuários individuais (várias faces da mesma pessoa). Faces individuais são armazenadas como vetores faciais, uma representação matemática da face (não uma imagem real da face). Imagens diferentes da mesma pessoa podem ser usadas para criar e armazenar vários vetores faciais na mesma coleção. Em seguida, você pode agregar vários vetores de face da mesma pessoa para criar um vetor de usuário. Os vetores do usuário podem oferecer maior precisão na busca facial com representações mais robustas, contendo vários graus de iluminação, nitidez, pose, aparência etc.

Depois de criar uma coleção, você pode usar uma face de entrada para pesquisar vetores de usuário ou vetores de face correspondentes em uma coleção. Pesquisar em vetores de usuário pode melhorar significativamente a precisão em comparação com a pesquisa em vetores faciais individuais. Você pode usar faces detectadas em imagens, vídeos armazenados e vídeos em streaming para pesquisar em vetores faciais armazenados. Você pode usar faces detectadas em imagens para pesquisar em vetores de usuário armazenados.

Para armazenar informações faciais, você precisará fazer o seguinte:

1. Criar uma coleção - Para armazenar informações faciais, você deve primeiro criar (CreateCollection) uma coleção facial em uma das AWS regiões da sua conta. Você especifica essa coleção de faces ao chamar a operação `IndexFaces`.
2. Index Faces - A IndexFacesoperação detecta face (s) em uma imagem, extrai e armazena o (s) vetor (s) facial (s) na coleção. Você pode usar essa operação para detectar faces em uma imagem e manter informações sobre traços faciais detectados em uma coleção. Esse é um exemplo de operação de API com base em armazenamento porque o serviço armazena as informações do vetor facial no servidor.

Para criar um usuário e associar vários vetores faciais a um usuário, você precisará fazer o seguinte:

1. Criar um usuário - Você deve primeiro criar um usuário com CreateUser. Você pode melhorar a precisão da correspondência facial agregando vários vetores faciais da mesma pessoa em um vetor de usuário. Você pode associar até 100 vetores de face a um vetor de usuário.

2. Associar faces - Depois de criar o usuário, você pode adicionar vetores de face existentes a esse usuário com a AssociateFacesoperação. Os vetores de face devem residir na mesma coleção de um vetor de usuário para serem associados a esse vetor de usuário.

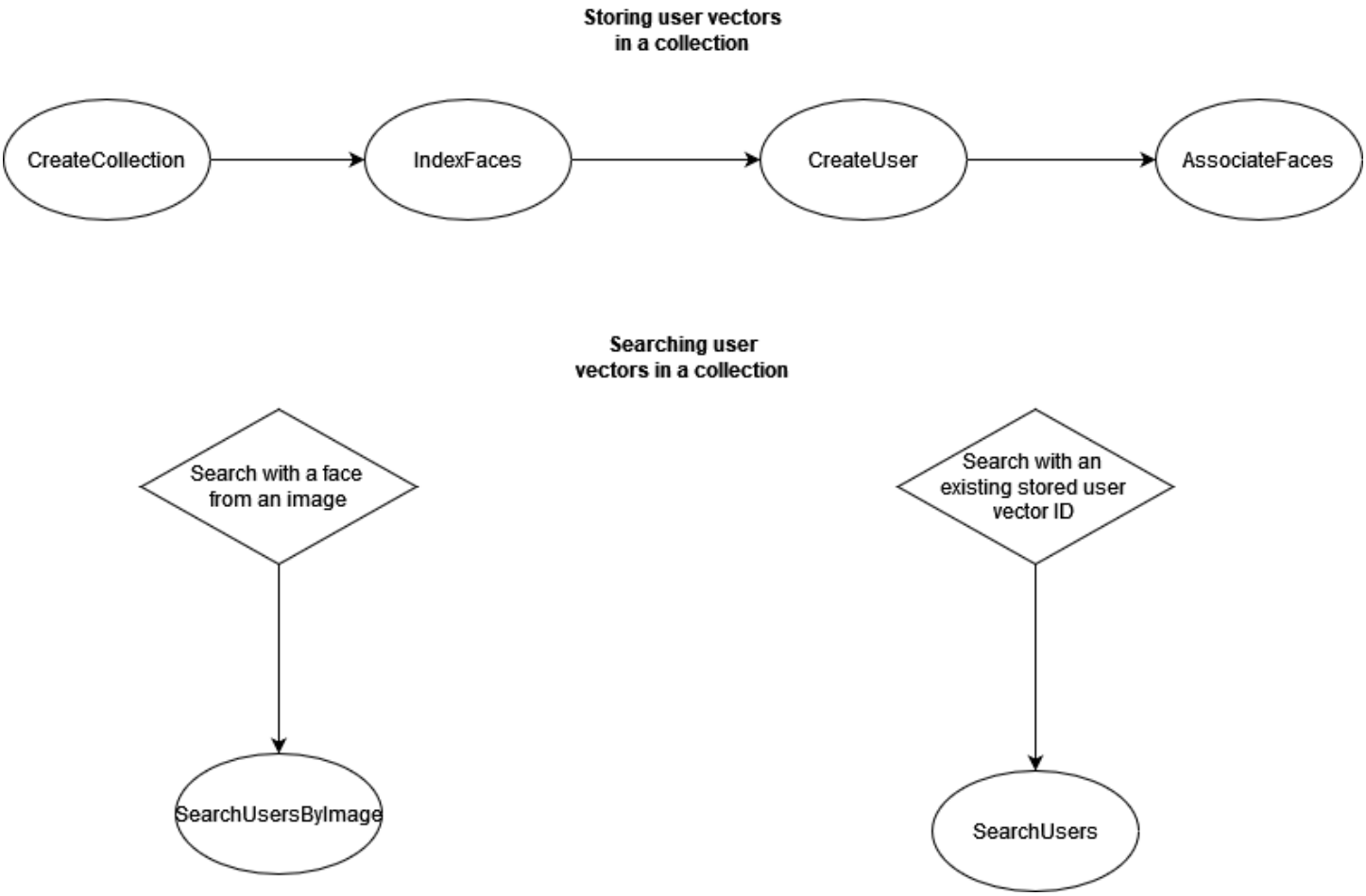
Depois de criar uma coleção e armazenar vetores faciais e de usuários, você pode usar as seguintes operações para pesquisar correspondências de faces:

- SearchFacesByImage- Para pesquisar rostos individuais armazenados com um rosto de uma imagem.
- SearchFaces- Para pesquisar rostos individuais armazenados com um ID facial fornecido.
- SearchUsers- Para pesquisar usuários armazenados com um ID facial ou ID de usuário fornecido.
- SearchUsersByImage- Para pesquisar usuários armazenados com um rosto de uma imagem.
- StartFaceSearch- Para pesquisar rostos em um vídeo armazenado.
- CreateStreamProcessor- Para pesquisar rostos em um streaming de vídeo.

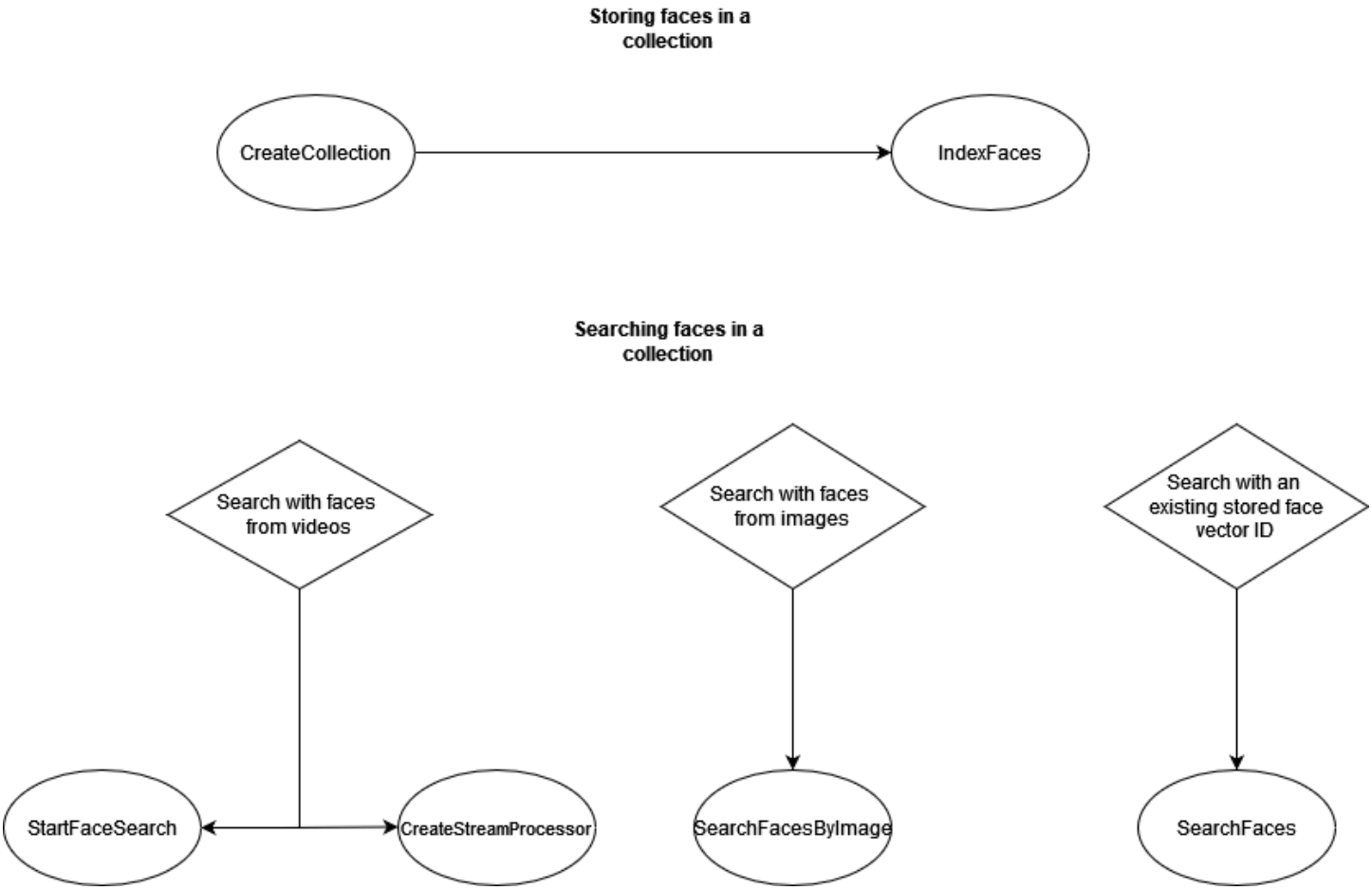
> **Note**
>
> As coleções armazenam vetores faciais, que são representações matemáticas de faces. As coleções não armazenam imagens de faces.

Os diagramas a seguir mostram a ordem das operações de chamada com base em suas metas de uso de coleções:

Para obter a máxima precisão de correspondência com os vetores do usuário:



Para uma correspondência de alta precisão com vetores faciais individuais:



Você pode usar coleções em vários cenários. Por exemplo, você pode criar uma coleção de rostos que armazene rostos detectados a partir de imagens digitalizadas de crachás de funcionários e emitidas IDs pelo governo usando as operações `IndexFaces` e. `AssociateFaces` Quando um funcionário entra no prédio, uma imagem da face do funcionário é capturada e enviada para a operação `SearchUsersByImage`. Se a correspondência de face produzir uma pontuação de similaridade suficientemente alta (por exemplo, 99%), você poderá autenticar o funcionário.

Gerenciar coleções de faces, faces e usuários

A coleção facial é o recurso usado pelo Amazon Rekognition para conter informações relacionadas a faces e usuários. Você usa várias operações da API do Rekognition para gerenciar a coleção em si e todas as faces ou vetores faciais armazenados na coleção.

Gerenciar uma coleção

A coleção de faces é o principal recurso do Amazon Rekognition, e cada coleção de faces criada tem um nome de recurso da Amazon (ARN) exclusivo. Você cria cada coleção de rostos em uma

AWS região específica da sua conta. Quando uma coleção é criada, ela é associada à versão mais recente do modelo de detecção de faces. Para obter mais informações, consulte Noções básicas do versionamento de modelo.

Você pode realizar as seguintes operações de gerenciamento em uma coleção:

- Crie uma coleção com CreateCollection. Para obter mais informações, consulte Criar uma coleção.
- Liste as coleções disponíveis com ListCollections. Para obter mais informações, consulte Listar coleções.
- Descreva uma coleção com DescribeCollection. Para obter mais informações, consulte Descrever uma coleção.
- Exclua uma coleção com DeleteCollection. Para obter mais informações, consulte Excluir uma coleção.

Gerenciar faces em uma coleção

Depois de criar uma coleção de faces, você pode armazenar faces nela. O Amazon Rekognition fornece as seguintes operações para gerenciar faces em uma coleção:

- A operação IndexFaces detecta faces na imagem de entrada (JPEG ou PNG) e a adiciona à coleção de faces especificada. Um ID exclusivo é retornado para cada face detectada na imagem. Depois de manter faces, você poderá pesquisar a coleção de faces em busca de correspondências de face. Para obter mais informações, consulte Adicionar faces a uma coleção.
- A operação ListFaces lista as faces em uma coleção. Para obter mais informações, consulte Adicionar faces a uma coleção.
- A operação DeleteFaces exclui faces de uma coleção. Para obter mais informações, consulte Excluir faces de uma coleção.

Gerenciar usuários em uma coleção

Depois de armazenar vários vetores faciais da mesma pessoa, você pode melhorar a precisão associando todos esses vetores faciais em um vetor de usuário. Você pode usar as seguintes operações para gerenciar seus usuários:

- CreateUser- A operação cria um novo usuário em uma coleção com um ID de usuário exclusivo fornecido.

- AssociateUsers- Adicione de 1 a 100 faces exclusivas IDs a um ID de usuário. Depois de associar pelo menos um ID facial a um usuário, você pode pesquisar correspondências com esse usuário em sua coleção.
- ListUsers- Lista os usuários em uma coleção.
- DeleteUsers- Exclui um usuário de uma coleção com o ID de usuário fornecido.
- DisassociateFaces- Remove um ou mais rostos IDs de um usuário.

Usar limites de similaridade para associar e combinar rostos

Limites de similaridade são usados para associar e combinar faces. O que segue é uma orientação para o uso de limites de similaridade em ambos os casos de uso.

Usando limites de similaridade para associar faces

Ao associar rostos usando a AssociateFacesoperação, é importante garantir que os rostos associados a um usuário sejam todos da mesma pessoa. Para ajudar, o parâmetro `UserMatchThreshold` especifica a confiança mínima de correspondência do usuário necessária para que a nova face seja associada a um `UserID` que já contenha pelo menos uma `FaceID`. Isso ajuda a garantir que `FaceIds` eles estejam associados ao `UserID` à direita . O valor varia de 0 a 100 e o valor padrão é 75.

Usar limites de similaridade para combinar faces

Nós permitimos que você controle os resultados de todas as operações de pesquisa (CompareFacesSearchFaces, SearchFacesByImage, SearchUsers,, SearchUsersByImage) fornecendo um limite de similaridade como parâmetro de entrada.

`FaceMatchThreshold`, é o atributo de entrada do limite de similaridade para `SearchFaces` e `SearchFacesByImage`, e controla quantos resultados são retornados com base na semelhança com a face que está sendo correspondida. O atributo de limite de similaridade para `SearchUsers` e `SearchUsersByImage` é `UserMatchThreshold`, e ele controla quantos resultados são retornados com base na semelhança com o vetor do usuário que está sendo correspondido. O atributo de limite é `SimilarityThreshold` para `CompareFaces`.

Respostas com um valor de atributo de resposta de `Similarity` que é menor do que o limite não são retornados. Esse limite é importante para calibrar para seu caso de uso, uma vez que ele pode

determinar quantos falsos positivos são incluídos nos resultados de correspondência. Isso controla a recuperação dos resultados da pesquisa — quanto menor o limite, maior a recuperação.

Todos os sistemas de aprendizagem de máquina são probabilísticos. Você deve usar seu julgamento para definir o limite de semelhança correto, dependendo de seu caso de uso. Por exemplo, se você quer criar um aplicativo para identificar fotos semelhantes procurando familiares, você pode escolher um limite inferior (por exemplo, 80%). Por outro lado, para vários casos de uso de aplicação da lei, recomendamos usar um alto valor limite de 99% ou superior para reduzir erros de identificação acidental.

Além de `FaceMatchThreshold` e `UserMatchThreshold`, você pode usar o atributo de resposta `Similarity` como um meio de reduzir erros de identificação acidentais. Por exemplo, você pode optar por usar um limite inferior (como 80%) para retornar mais resultados. Em seguida, você pode usar o atributo de resposta Similaridade (porcentagem de similaridade) para restringir a escolha e filtrar as respostas corretas em seu aplicativo. Mais uma vez, usar uma semelhança maior (por exemplo, 99% e acima) reduz o risco de erros de identificação.

Orientação para indexar faces em cenários comuns

Veja a seguir as orientações de uso do `IndexFaces` em cenários comuns.

Aplicações críticas ou de segurança pública

- Chame o IndexFaces com imagens que contêm apenas uma face cada e associe o ID de face retornado com o identificador para a pessoa presente da imagem.
- Você pode usar o DetectFaces antes da indexação para verificar se há somente uma face na imagem. Se mais de uma face for detectada, envie novamente a imagem, após revisá-la, e deixe apenas uma face presente. Isso evita a indexação não intencional de várias faces e sua associação com a mesma pessoa.

Aplicativos de compartilhamento de fotos e mídias sociais

- Você deve usar o `IndexFaces` sem restrições em imagens que contenham várias faces, em casos de uso como álbuns de família. Nesses casos, você precisa identificar cada pessoa em cada foto e usar essas informações para agrupar fotos pelas pessoas presentes.

Uso geral

- Indexe várias imagens diferentes da mesma pessoa, especialmente com atributos faciais diferentes (poses faciais, pelos faciais etc.), crie um usuário e associe as diferentes faces a esse usuário para melhorar a qualidade da correspondência.
- Inclua um processo de revisão para que as correspondências malsucedidas possam ser indexadas com o identificador da face correta para melhorar a capacidade de correspondência das faces subsequentes.
- Para obter informações sobre a qualidade da imagem, consulte Recomendações para imagens de entrada de comparação facial.

Pesquisa de faces e usuários em uma coleção

Depois de criar uma coleção de faces e armazenar vetores faciais e/ou vetores de usuário, você pode pesquisar correspondências de faces em uma coleção de faces. Com o Amazon Rekognition, você pode pesquisar faces em uma coleção que corresponda a:

- Um ID de face fornecido (SearchFaces). Para obter mais informações, consulte Procurando uma face com um ID facial.
- O maior face em uma imagem fornecida (SearchFacesByImage). Para obter mais informações, consulte Procurando um rosto com uma imagem.
- Faces em um vídeo armazenado. Para obter mais informações, consulte Pesquisanda faces em vídeos armazenados.
- Faces em um streaming de vídeo. Para obter mais informações, consulte Trabalhando com eventos de streaming de vídeo.

Você pode usar a operação `CompareFaces` para comparar uma face em uma imagem de origem com faces na imagem de destino. O escopo dessa comparação está limitado às faces detectadas na imagem de destino. Para obter mais informações, consulte Comparar faces em imagens.

As várias operações de pesquisa vistas na lista a seguir comparam uma face (identificada por uma imagem de `FaceId` ou por uma imagem de entrada) com todas as faces armazenadas em uma determinada coleção de faces:

- SearchFaces
- SearchFacesByImage

- SearchUsers
- SearchUsersByImage

Casos de uso que envolvem segurança pública

Além das recomendações listadas em Melhores práticas para sensores, imagens de entrada e vídeos e Orientação para indexar faces em cenários comuns, você deve usar as melhores práticas a seguir ao implantar sistemas de detecção e comparação facial e em casos de uso que envolvem a segurança pública. Em primeiro lugar, você deve usar limites de confiança de 99% ou superior para reduzir erros e falsos positivos. Em segundo lugar, você deve envolver revisores humanos para verificar os resultados recebidos de um sistema de detecção ou comparação facial, e não deve tomar decisões com base na saída do sistema sem uma revisão humana adicional. Os sistemas de detecção e comparação facial devem servir como uma ferramenta para ajudar a restringir o campo e permitir que pessoas revisem e analisem as opções rapidamente. Em terceiro lugar, recomendamos que você deve ser transparente sobre o uso de sistemas de detecção e comparação facial nesses casos de uso, inclusive, sempre que possível, informando os usuários finais e pessoas afetadas sobre o uso desses sistemas, obtendo seu consentimento para a utilização e fornecendo um mecanismo em que os usuários finais e pessoas afetadas possam fornecer comentários para melhorar o sistema.

Se você for uma agência de aplicação da lei que usa o recurso de comparação facial do Amazon Rekognition em conexão com investigações criminais, deverá seguir os requisitos listados na seção Termos de serviço da AWS. Eles incluem o seguinte.

- Coloque pessoas devidamente treinadas para examinar todas as decisões relativas a ações que possam afetar as liberdades civis de uma pessoa ou direitos humanos equivalentes.
- Treine o pessoal sobre a utilização responsável dos sistemas de reconhecimento facial.
- Divulgue publicamente o seu uso de sistemas de reconhecimento facial.
- Não use o Amazon Rekognition para vigilância contínua de uma pessoa sem análise independente ou circunstâncias exigentes.

Em todos os casos, as correspondências da comparação facial devem ser visualizadas no contexto de outras provas convincentes, e não devem ser usadas como o único determinante para tomar alguma medida. No entanto, se a comparação facial for usada para non-law-enforcement cenários (por exemplo, para desbloquear um telefone ou autenticar a identidade de um funcionário para

acessar um prédio comercial privado e seguro), essas decisões não exigiriam uma auditoria manual porque não afetariam as liberdades civis ou os direitos humanos equivalentes de uma pessoa.

Se está planejando usar um sistema de detecção ou comparação facial para casos de uso que envolvam a segurança pública, você deve empregar as práticas recomendadas mencionadas anteriormente. Além disso, você deve consultar os recursos publicados sobre o uso da comparação facial. Isso inclui o documento Face Recognition Policy Development Template For Use In Criminal Intelligence and Investigative Activities publicado pelo Bureau of Justice Assistance do Departamento de Justiça. O modelo oferece vários recursos relacionados a comparação facial e biometria e foi criado para fornecer às agências de segurança pública e departamentos de polícia uma estrutura para o desenvolvimento de políticas de comparação facial que estejam em conformidade com as leis aplicáveis, reduzam os riscos à privacidade e estabeleçam as formas de responsabilidade e supervisão da entidade. Alguns materiais adicionais incluem Best Privacy Practices for Commercial Use of Facial Recognition da National Telecommunications and Information Administration (Administração Nacional de Informação e Telecomunicações) e Best Practices for Common Uses of Facial Recognition pelo pessoal da Federal Trade Commission (Comissão Federal de Comércio). Outros materiais podem ser desenvolvidos e publicados no futuro, e você deve se informar continuamente sobre esse tópico importante.

Lembre-se que você deve estar em conformidade com todas as leis aplicáveis ao usar os serviços da AWS, e não deve usar qualquer serviço da AWS de forma a violar direitos ou prejudicar outras pessoas. Isso significa que você não deve usar os serviços da AWS em casos de uso de segurança pública de uma forma que discrimine ilegalmente uma pessoa ou viole seus direitos individuais, sua privacidade ou suas liberdades civis. Você deve obter orientação jurídica apropriada, conforme necessário, para analisar todos os requisitos legais ou dúvidas sobre seu caso de uso.

Usando o Amazon Rekognition para ajudar na segurança pública

O Amazon Rekognition pode ajudar em cenários de segurança pública e aplicação da lei, como encontrar crianças perdidas, combater o tráfico de pessoas ou prevenir crimes. Em situações legais e de segurança pública, considere o seguinte:

- Use o Amazon Rekognition como a primeira etapa para encontrar possíveis correspondências. As respostas das operações faciais do Amazon Rekognition permitem que você obtenha rapidamente um conjunto de possíveis correspondências para análise posterior.
- Não use as respostas do Amazon Rekognition para tomar decisões autônomas em cenários que exijam análise por um ser humano. Se você é uma agência policial que usa o Amazon Rekognition para ajudar na identificação de uma pessoa em conexão com uma investigação criminal, e ações

serão tomadas com base na identificação que possa afetar as liberdades civis ou direitos humanos equivalentes dessa pessoa, a decisão de agir deve ser tomada por uma pessoa devidamente treinada com base no exame independente das evidências de identificação.

- Use um limite de similaridade de 99% para cenários onde correspondências de similaridade de face altamente precisos são necessários. Um exemplo disso é autenticar o acesso a um edifício.
- Quando os direitos civis forem uma preocupação, como os casos de uso envolvendo a aplicação da lei, use os limites de confiança de 99% ou acima e empregue a revisão humana das previsões de comparação facial para garantir que os direitos civis de uma pessoa não sejam violados.
- Use um limite de similaridade inferior a 99% para cenários que se beneficiam de um conjunto maior de correspondências em potencial. Um exemplo disso é encontrar desaparecidos. Se necessário, você pode usar o atributo de resposta de similaridade de correspondências para determinar quão similar as correspondências em potencial são para a pessoa que você deseja reconhecer.
- Tenha um plano para falsos positivos de correspondências de faces que são retornados pelo Amazon Rekognition. Por exemplo, melhorar a correspondência usando várias imagens da mesma pessoa quando você cria o índice com a operação IndexFaces. Para obter mais informações, consulte Orientação para indexar faces em cenários comuns.

Em outros casos de uso (como mídias sociais), recomendamos que você use seu bom senso para avaliar se os resultados do Amazon Rekognition precisam de análise humana. Além disso, dependendo dos requisitos de seu aplicativo, o limite de similaridade pode ser menor.

Criar uma coleção

Você pode usar a operação CreateCollection para criar uma coleção.

Para obter mais informações, consulte Gerenciar uma coleção.

Para criar uma coleção (SDK)

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use os exemplos a seguir para chamar a operação `CreateCollection`.

Java

O exemplo a seguir cria uma coleção e exibe o Amazon Resource Name (ARN – Nome de recurso da Amazon).

Altere o valor de `collectionId` para o nome da coleção que você deseja criar.

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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
//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

Esse AWS CLI comando exibe a saída JSON para a operação da `create-collection` CLI.

Substitua o valor de `collection-id` pelo nome da coleção que você deseja criar.

Substitua o valor de `profile_name` com o nome do seu perfil de desenvolvedor.

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

Python

O exemplo a seguir cria uma coleção e exibe o Amazon Resource Name (ARN – Nome de recurso da Amazon).

Altere o valor de `collection_id` para o nome da coleção que você deseja criar. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

O exemplo a seguir cria uma coleção e exibe o Amazon Resource Name (ARN – Nome de recurso da Amazon).

Altere o valor de `collectionId` para o nome da coleção que você deseja criar.

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

No exemplo a seguir, substitua o valor de `region` pelo nome da região associada à sua conta e substitua o valor de `collectionName` com o nome desejado da sua coleção.

Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
//Copyright 2018 Amazon.com, Inc. or its 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 solicitação de operação

A entrada para `CreationCollection` é o nome da coleção que você deseja criar.

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

CreateCollection resposta da operação

O Amazon Rekognition cria a coleção e retorna o nome de recurso da Amazon (ARN) da coleção recém-criada.

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

Aplicação de tags nas coleções

Você pode identificar, organizar, pesquisar e filtrar coleções do Amazon Rekognition usando tags. Cada tag é um rótulo que consiste em um valor e uma chave definida pelo usuário.

Você também pode usar tags para controlar o acesso a uma coleção usando o Identity and Access Management (IAM). Para obter mais informações, consulte Controle do acesso aos AWS recursos usando tags de recursos.

Tópicos

- Adicionar tags a uma nova coleção
- Adicionar tags a uma coleção existente
- Listar tags em uma coleção
- Excluir tags de uma coleção

Adicionar tags a uma nova coleção

Você pode adicionar tags a uma coleção ao criá-la usando a operação `CreateCollection`. Especifique uma ou mais tags no parâmetro `Tags` de entrada da matriz.

AWS CLI

Substitua o valor de `profile_name` com o nome do seu perfil de desenvolvedor.

```
aws rekognition create-collection --collection-id "collection-name" --tags
 "{"key1":"value1","key2":"value2"}" --profile profile-name
```

Para dispositivos Windows:

```
aws rekognition create-collection --collection-id "collection-name" --tags
 "{\"key1\":\"value1\",\"key2\":\"value2\"}" --profile profile-name
```

Python

Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
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()
```

Adicionar tags a uma coleção existente

Para adicionar uma ou mais tags a uma coleção existente, use a operação `TagResource`. Especifique o nome do recurso da Amazon (ARN) (`ResourceArn`) e as tags (`Tags`) da coleção que você deseja adicionar. O exemplo a seguir mostra como adicionar duas tags.

AWS CLI

Substitua o valor de `profile_name` com o nome do seu perfil de desenvolvedor.

```
aws rekognition tag-resource --resource-arn collection-arn --tags
 "{"key1":"value1","key2":"value2"}" --profile profile-name
```

Para dispositivos Windows:

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

Python

Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
# Copyright 2018 Amazon.com, Inc. or its 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 você não souber o nome do recurso da Amazon da coleção, poderá usar a operação `DescribeCollection`.

Listar tags em uma coleção

Para listar as tags anexadas a uma coleção, use a operação `ListTagsForResource` e especifique o ARN da coleção (`ResourceArn`). A resposta é um mapa de chaves e valores de tags anexados à coleção especificada.

AWS CLI

Substitua o valor de `profile_name` com o nome do seu perfil de desenvolvedor.

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

Python

Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
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()
```

A saída exibe uma lista de tags anexadas à coleção:

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

Excluir tags de uma coleção

Para remover uma ou mais tags de uma coleção, use a operação `UntagResource`. Especifique o ARN do modelo (`ResourceArn`) e das chaves de tag (`Tag-Keys`) que você deseja remover.

AWS CLI

Substitua o valor de `profile_name` com o nome do seu perfil de desenvolvedor.

```
aws rekognition untag-resource --resource-arn resource-arn --profile profile-name --
tag-keys "key1" "key2"
```

Como alternativa, você pode especificar chaves de tag neste formato:

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

Python

Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
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()
```

Listar coleções

Você pode usar a operação ListCollections para listar as coleções na região que está usando.

Para obter mais informações, consulte Gerenciar uma coleção.

Para listar coleções (SDK)

1. Se ainda não tiver feito isso:
   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.
   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.
2. Use os exemplos a seguir para chamar a operação `ListCollections`.

   Java

   O exemplo a seguir lista as coleções na região atual.

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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
//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

Esse AWS CLI comando exibe a saída JSON para a operação da `list-collections` CLI. Substitua o valor de `profile_name` com o nome do seu perfil de desenvolvedor.

```
aws rekognition list-collections --profile profile-name
```

Python

O exemplo a seguir lista as coleções na região atual.

Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

O exemplo a seguir lista as coleções na região atual.

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

Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
//Copyright 2018 Amazon.com, Inc. or its 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 solicitação de operação

A entrada para `ListCollections` é o número máximo de coleções a serem retornadas.

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

```
}
```

Se a resposta tiver mais coleções do que as solicitadas por `MaxResults`, será retornado um token que poderá ser usado para obter o próximo conjunto de resultados, em uma chamada subsequente para `ListCollections`. Por exemplo:

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

ListCollections resposta da operação

O Amazon Rekognition retorna uma matriz de coleções (`CollectionIds`). Uma matriz separada (`FaceModelVersions`) fornece a versão do modelo de face usado para analisar faces em cada coleção. Por exemplo, na resposta JSON a seguir, a coleção `MyCollection` analisa faces usando a versão 2.0 do modelo de face. A coleção `AnotherCollection` usa a versão 3.0 do modelo de face. Para obter mais informações, consulte Noções básicas do versionamento de modelo.

`NextToken` é o token usado para obter o próximo conjunto de resultados, em uma chamada subsequente para `ListCollections`.

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

Descrever uma coleção

Você pode usar a operação DescribeCollection para obter as seguintes informações sobre uma coleção:

- O número de faces que são indexados na coleção.
- A versão do modelo que está sendo usado com a coleção. Para obter mais informações, consulte the section called “Noções básicas do versionamento de modelo”.
- O nome de recurso da Amazon (ARN) da coleção.
- A data e a hora da criação da coleção.

Para descrever uma coleção (SDK)

1. Se ainda não tiver feito isso:
   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.
   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.
2. Use os exemplos a seguir para chamar a operação `DescribeCollection`.

   Java

   Este exemplo descreve uma coleção.

   Altere o valor de `collectionId` para o ID da coleção desejada.

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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
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

Esse AWS CLI comando exibe a saída JSON para a operação da `describe-collection` CLI. Altere o valor de `collection-id` para o ID da coleção desejada. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
aws rekognition describe-collection --collection-id collection-name --profile
 profile-name
```

Python

Este exemplo descreve uma coleção.

Altere o valor de `collection_id` para o ID da coleção desejada. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Este exemplo descreve uma coleção.

Altere o valor de `collectionId` para o ID da coleção desejada.

```
//Copyright 2018 Amazon.com, Inc. or its 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 solicitação de operação

A entrada para `DescribeCollection` é o ID da coleção desejada, conforme mostrado no exemplo JSON a seguir.

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

DescribeCollectionresposta da operação

A resposta inclui:

- O número de faces que são indexados na coleção, `FaceCount`.
- A versão do modelo facial que está sendo usada com a coleção, `FaceModelVersion`. Para obter mais informações, consulte the section called “Noções básicas do versionamento de modelo”.
- A coleção do nome de recurso da Amazon, `CollectionARN`.
- A data e a hora da criação da coleção, `CreationTimestamp`. O valor de `CreationTimestamp` é o número de milissegundos desde o horário Unix até a criação da coleção. O horário epoch Unix é 00:00:00 UTC (Tempo Universal Coordenado), quinta-feira, 1° de janeiro de 1970. Para obter mais informações, consulte Hora Unix.

```
{
    "CollectionARN": "arn:aws:rekognition:us-east-1:nnnnnnnnnnn:collection/
MyCollection",
    "CreationTimestamp": 1.533422155042E9,
    "FaceCount": 200,
    "UserCount" : 20,
    "FaceModelVersion": "1.0"
}
```

Excluir uma coleção

Você pode usar a operação DeleteCollection para excluir uma coleção.

Para obter mais informações, consulte Gerenciar uma coleção.

Para excluir uma coleção (SDK)

1. Se ainda não tiver feito isso:
   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.
   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.
2. Use os exemplos a seguir para chamar a operação `DeleteCollection`.

Java

Este exemplo exclui uma coleção.

Altere o valor `collectionId` para a coleção que você deseja excluir.

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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
// 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

Esse AWS CLI comando exibe a saída JSON para a operação da `delete-collection` CLI. Substitua o valor de `collection-id` pelo nome da coleção que você deseja excluir. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
aws rekognition delete-collection --collection-id collection-name --profile
 profile-name
```

Python

Este exemplo exclui uma coleção.

Altere o valor `collection_id` para a coleção que você deseja excluir. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Este exemplo exclui uma coleção.

Altere o valor `collectionId` para a coleção que você deseja excluir.

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

Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
//Copyright 2018 Amazon.com, Inc. or its 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 solicitação de operação

A entrada para `DeleteCollection` é o ID da coleção a ser excluída, conforme mostrado no exemplo JSON a seguir.

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

DeleteCollection resposta da operação

A resposta `DeleteCollection` contém um código de status HTTP que indica o sucesso ou falha da operação. `200` é retornado se a coleção for excluída com êxito.

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

Adicionar faces a uma coleção

Você pode usar a operação IndexFaces para detectar faces em uma imagem e adicioná-las a uma coleção. Para cada face detectada, o Amazon Rekognition extrai características faciais e armazena as informações das características em um banco de dados. Além disso, o comando armazena metadados para cada face detectada na coleção de faces especificada. O Amazon Rekognition não armazena os bytes reais da imagem.

Para obter informações sobre como fornecer faces adequadas para a indexação, consulte Recomendações para imagens de entrada de comparação facial.

Para cada face, a operação `IndexFaces` mantém as seguintes informações:

- Características faciais multidimensionais: `IndexFaces` usa a análise facial para extrair informações multidimensionais sobre as características faciais e armazena as informações na coleção facial. Você não pode acessar essas informações diretamente. No entanto, o Amazon Rekognition usa essas informações ao pesquisar correspondências faciais em uma coleção de faces.

- Metadados — Os metadados de cada face incluem uma caixa delimitadora, um nível de confiança (que a caixa delimitadora contém uma face), atribuído pelo IDs Amazon Rekognition (ID facial e ID

da imagem) e uma ID de imagem externa (se você a forneceu) na solicitação. Essas informações são retornadas para você em resposta para a chamada à API `IndexFaces`. Para obter um exemplo, consulte o elemento `face` na resposta de exemplo a seguir.

O serviço retorna esses metadados em resposta para as seguintes chamadas à API:

- `ListFaces`
- Pesquisar operações de faces – As respostas para SearchFaces e SearchFacesByImage retornam a confiança na correspondência de cada face, além desses metadados da face correspondente.

O número de faces indexadas por `IndexFaces` depende da versão do modelo de detecção de faces associado à coleção de entrada. Para obter mais informações, consulte Noções básicas do versionamento de modelo.

As informações sobre faces indexadas são retornadas em uma matriz de objetos FaceRecord.

Você pode desejar associar faces indexadas com a imagem em que foram detectadas. Por exemplo, você pode desejar manter um índice de cliente de imagens e faces nas imagens. Para associar faces a uma imagem, especifique um ID de imagem no parâmetro da solicitação `ExternalImageId`. O ID da imagem pode ser o nome do arquivo ou outro ID que você criar.

Além das informações anteriores que a API mantém na coleção de faces, a API também retorna detalhes da face que não são mantidos na coleção. (Consulte o elemento `faceDetail` na resposta de exemplo a seguir.)

> **Note**
>
> O `DetectFaces` retorna as mesmas informações, portanto, você não precisa chamar `DetectFaces` e `IndexFaces` para a mesma imagem.

Filtrar faces

A IndexFaces operação permite filtrar as faces indexadas a partir de uma imagem. Com `IndexFaces`, você pode especificar um número máximo de faces a serem indexadas ou pode escolher apenas faces detectadas com um índice de alta qualidade.

Você pode especificar o número máximo de faces que são indexadas por `IndexFaces` usando o parâmetro de entrada `MaxFaces`. Isso é útil quando você deseja indexar as maiores faces em uma imagem e não deseja indexar as faces menores, como as faces de pessoas em segundo plano.

Por padrão, `IndexFaces` escolhe um requisito de qualidade que é usado para filtrar faces. É possível usar o parâmetro de entrada `QualityFilter` para definir explicitamente o requisito de qualidade. Os valores são:

- `AUTO` — O Amazon Rekognition escolhe a barra de qualidade usada para filtrar as faces (valor padrão).
- `LOW` — Todas, exceto as faces de menor qualidade, são indexadas.
- `MEDIUM`
- `HIGH` — Somente as faces da mais alta qualidade são indexadas.
- `NONE` – nenhuma face é filtrada com base na qualidade.

`IndexFaces` filtra faces pelos seguintes motivos:

- A face é muito pequena em comparação com as dimensões da imagem.
- A face está muito embaçada.
- A imagem é muito escura.
- A face tem uma pose radical.
- A face não tem detalhes suficientes para ser adequada à pesquisa de faces.

> **Note**
>
> Para usar a filtragem de qualidade, é necessário ter uma coleção associada à versão 3, ou posterior, do modelo de faces. Para obter a versão do modelo de face associado a uma coleção, chame DescribeCollection.

As informações sobre faces que não estão indexadas por `IndexFaces` são retornadas em uma matriz de objetos do UnindexedFace. A matriz de `Reasons` contém uma lista dos motivos pelos quais uma face não é indexada. Por exemplo, um valor de `EXCEEDS_MAX_FACES` é uma face que não foi indexada porque o número de faces especificado pelo `MaxFaces` já foi detectado.

Para obter mais informações, consulte Gerenciar faces em uma coleção.

Para adicionar faces a uma coleção (SDK)

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Faça upload de uma imagem (contendo uma ou mais faces) para seu bucket do Amazon S3.

   Para obter instruções, consulte Como fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

3. Use os exemplos a seguir para chamar a operação `IndexFaces`.

   Java

   Este exemplo exibe os identificadores para faces adicionadas à coleção.

   Altere o valor de `collectionId` para o nome da coleção a que você deseja adicionar uma face. Substitua os valores de `bucket` e `photo` pelos nomes do bucket do Amazon S3 e da imagem usados na etapa 2. O parâmetro `.withMaxFaces(1)` restringe o número de faces indexadas a 1. Remova ou altere seu valor para atender às suas necessidades.

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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
//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

Esse AWS CLI comando exibe a saída JSON para a operação da `index-faces` CLI.

Substitua o valor de `collection-id` pelo nome da coleção em que você deseja que a face seja armazenada. Substitua os valores de `Bucket` e `Name` pelo bucket do Amazon S3 e pelo arquivo de imagem que você usou na etapa 2. O parâmetro `max-faces` restringe o número de faces indexadas a 1. Remova ou altere seu valor para atender às suas necessidades. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
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 você estiver acessando a CLI em um dispositivo Windows, use aspas duplas em vez de aspas simples e escape das aspas duplas internas com barra invertida (ou seja, \) para resolver quaisquer erros de analisador que você possa encontrar. Para obter um exemplo, veja o seguinte:

```
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

Este exemplo exibe os identificadores para faces adicionadas à coleção.

Altere o valor de `collectionId` para o nome da coleção a que você deseja adicionar uma face. Substitua os valores de `bucket` e `photo` pelos nomes do bucket do Amazon S3 e da imagem usados na etapa 2. O parâmetro de entrada `MaxFaces` restringe o número de faces indexadas a 1. Remova ou altere seu valor para atender às suas necessidades. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Este exemplo exibe os identificadores para faces adicionadas à coleção.

Altere o valor de `collectionId` para o nome da coleção a que você deseja adicionar uma face. Substitua os valores de `bucket` e `photo` pelos nomes do bucket do Amazon S3 e da imagem usados na etapa 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 solicitação de operação

A entrada para `IndexFaces` é a imagem a ser indexada e a coleção à qual adicionar a face ou as faces.

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "ExternalImageId": "input.jpg",
    "DetectionAttributes": [
        "DEFAULT"
    ],
    "MaxFaces": 1,
    "QualityFilter": "AUTO"
}
```

IndexFaces resposta da operação

`IndexFaces` retorna informações sobre as faces que foram detectadas na imagem. Por exemplo, a resposta JSON a seguir inclui os atributos de detecção padrão para faces detectadas na imagem de entrada. O exemplo também mostra faces não indexadas porque o valor `MaxFaces` do parâmetro de entrada foi excedido — a matriz `Reasons` contém EXCEEDS_MAX_FACES. Se uma face não for indexada por motivos de qualidade, `Reasons` conterá valores como LOW_SHARPNESS ou LOW_BRIGHTNESS. Para obter mais informações, consulte 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"
            ]
        }
    ]
}
```


Para obter todas as informações faciais, especifique 'ALL' para o parâmetro de solicitação `DetectionAttributes`. Por exemplo, na resposta do exemplo a seguir, observe as informações adicionais no elemento `faceDetail`, que não é mantido no servidor:

- 25 pontos de referência faciais (em comparação com apenas cinco no exemplo anterior)
- Dez atributos faciais (óculos, barba, oclusão, direção do olhar e assim por diante)
- Emoções (consulte o elemento `emotion`)

O elemento `face` fornece metadados que são mantidos no servidor.

`FaceModelVersion` é a versão do modelo de face associado à coleção. Para obter mais informações, consulte Noções básicas do versionamento de modelo.

`OrientationCorrection` é a orientação estimada da imagem. As informações de correção de orientação não serão retornadas se você estiver usando uma versão do modelo de detecção facial que seja superior à versão 3. Para obter mais informações, consulte Obter a orientação e as coordenadas da caixa delimitadora da imagem.

O exemplo de resposta a seguir mostra o JSON retornado ao especificar ["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": []
}
```

Listanda faces e usuários associados em uma coleção

Você pode usar a ListFacesoperação para listar faces e seus usuários associados em uma coleção.

Para obter mais informações, consulte Gerenciar faces em uma coleção.

Para listar faces em uma coleção (SDK)

1. Se ainda não tiver feito isso:
   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.
   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.
2. Use os exemplos a seguir para chamar a operação `ListFaces`.

   Java

   Este exemplo exibe uma lista de faces em uma coleção.

   Altere o valor de `collectionId` para a coleção desejada.

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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
// 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

Esse AWS CLI comando exibe a saída JSON para a operação da `list-faces` CLI. Substitua o valor de `collection-id` pelo nome da coleção que você deseja listar. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
aws rekognition list-faces --collection-id "collection-id"  --profile profile-
name
```

Python

Este exemplo exibe uma lista de faces em uma coleção.

Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Este exemplo exibe uma lista de faces em uma coleção.

Altere o valor de `collectionId` para a coleção desejada.

```
//Copyright 2018 Amazon.com, Inc. or its 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 solicitação de operação

A entrada para `ListFaces` é o ID da coleção para a qual você deseja listar faces. `MaxResultsé` o número máximo de faces a serem retornadas. ListFaces também recebe uma lista de faces IDs para filtrar os resultados e uma ID de usuário fornecida para listar somente as faces associadas a um determinado usuário.

```
{
    "CollectionId": "MyCollection",
    "MaxResults": 1
}
```

Se a resposta tiver mais faces do que o solicitado por `MaxResults`, um token será retornado que você pode usar para obter o próximo conjunto de resultados, em uma chamada subsequente para `ListFaces`. Por exemplo:

```
{
    "CollectionId": "MyCollection",
    "NextToken": "sm+5ythT3aeEVIR4WA....",
    "MaxResults": 1
}
```

ListFaces resposta da operação

A resposta de `ListFaces` são informações sobre os metadados da face armazenados na coleção especificada.

- FaceModelVersion— A versão do modelo facial associada à coleção. Para obter mais informações, consulte Noções básicas do versionamento de modelo.
- Faces — Informações sobre os faces da coleção. Isso inclui informações sobre BoundingBox, confiança, identificadores de imagem e o ID da face. Para obter mais informações, consulte Face.
- NextToken— O token usado para obter o próximo conjunto de resultados.

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

Excluir faces de uma coleção

Você pode usar a operação DeleteFaces para excluir faces de uma coleção. Para obter mais informações, consulte Gerenciar faces em uma coleção.

Como excluir faces de uma coleção

1. Se ainda não tiver feito isso:
   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.
   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.
2. Use os exemplos a seguir para chamar a operação `DeleteFaces`.

   Java

   Este exemplo exclui uma única face de uma coleção.

   Altere o valor de `collectionId` para a coleção que contém a face que você deseja excluir. Altere o valor de `faces` para o ID da face que você deseja excluir. Para excluir várias faces, adicione a face IDs à `faces` matriz.

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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
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

Esse AWS CLI comando exibe a saída JSON para a operação da `delete-faces` CLI. Substitua o valor de `collection-id` pelo nome da coleção que contém a face que você deseja excluir. Substitua o valor `face-ids` de por uma matriz de faces IDs que você deseja excluir. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
aws rekognition delete-faces --collection-id "collection-id" --face-ids "faceid"
 --profile profile-name
```

Python

Este exemplo exclui uma única face de uma coleção.

Altere o valor de `collectionId` para a coleção que contém a face que você deseja excluir. Altere o valor de `faces` para o ID da face que você deseja excluir. Para excluir várias faces, adicione a face IDs à `faces` matriz. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Este exemplo exclui uma única face de uma coleção.

Altere o valor de `collectionId` para a coleção que contém a face que você deseja excluir. Altere o valor de `faces` para o ID da face que você deseja excluir. Para excluir várias faces, adicione a face IDs à `faces` lista.

```
//Copyright 2018 Amazon.com, Inc. or its 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 solicitação de operação

A entrada para `DeleteFaces` é o ID da coleção que contém as faces e uma matriz de faces IDs para as faces a serem excluídas.

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

DeleteFaces resposta da operação

A `DeleteFaces` resposta contém uma matriz de faces IDs para as faces que foram excluídas.

```
{
    "DeletedFaces": [
```

```
        "daf29cac-f910-41e9-851f-6eeb0e08f973"
    ]
}
```

Se o rosto IDs fornecido na entrada estiver atualmente associado a um usuário, ele será devolvido por motivos válidos. UnsuccessfulFaceDeletions

```
{
   "DeletedFaces": [
        "daf29cac-f910-41e9-851f-6eeb0e08f973"
    ],
     "UnsuccessfulFaceDeletions" : [
        {
           "FaceId" : "0b683aed-a0f1-48b2-9b5e-139e9cc2a757",
           "UserId" : "demoUser1",
           "Reason" : ["ASSOCIATED_TO_AN_EXISTING_USER"]
        }
     ]
}
```

Criar um usuário

Você pode usar a CreateUseroperação para criar um novo usuário em uma coleção usando uma ID de usuário exclusiva fornecida por você. Em seguida, você pode associar várias faces ao usuário recém-criado.

Para criar um usuário (SDK)

1. Se ainda não tiver feito isso:

   a. Crie ou atualize uma conta de usuário do IAM com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use os exemplos a seguir para chamar a operação `CreateUser`.

   Java

   Este exemplo de código Java cria um usuário.

```
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

Esse AWS CLI comando cria um usuário usando a operação `create-user` CLI.

```
aws rekognition create-user --user-id user-id --collection-id collection-name --
region region-name
--client-request-token request-token
```

Python

Este exemplo de código Python cria um usuário.

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

Este exemplo de código Go usa o AWS Go SDK V2 e cria um usuário.

```
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")

}
```

Excluir um usuário

Você pode usar a DeleteUseroperação para excluir um usuário de uma coleção, com base na ID de usuário fornecida. Observe que todas as faces associadas à ID de usuário são desassociadas da ID de usuário antes que a ID de usuário especificada seja excluída.

Para excluir um usuário (SDK)

1. Se ainda não tiver feito isso:

   a. Crie ou atualize uma conta de usuário do IAM com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use os exemplos a seguir para chamar a operação `DeleteUser`.

   Java

   Este exemplo de código Java exclui um usuário.

```
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

Esse AWS CLI comando exclui um usuário usando a operação `create-user` CLI.

```
aws rekognition delete-user --collection-id MyCollection
--user-id user-id --collection-id collection-name --region region-name
```

Python

Este exemplo de código em Python exclui um usuário.

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

Associando faces a um usuário

Você pode usar a AssociateFacesoperação para associar várias faces individuais a um único usuário. Para associar um rosto a um usuário, você deve primeiro criar uma coleção e um usuário. Observe que os vetores de face devem residir na mesma coleção em que o vetor do usuário reside.

Para associar faces (SDK)

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use os exemplos a seguir para chamar a operação `AssociateFaces`.

   Java

   Este exemplo de código Java associa um rosto a um usuário.

```
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

Esse AWS CLI comando associa uma face a um usuário, usando a operação `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

Este exemplo de código em Python associa uma face a um usuário.

```
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 resposta da operação

A resposta para `AssociateFaces` inclui o `UserStatus`, que é o status da solicitação de dissociação, bem como uma lista de itens `FaceIds` a serem associados. Uma lista de `UnsuccessfulFaceAssociations` também é retornada. Depois de enviar uma solicitação para `AssociateFaces`, a operação pode levar cerca de um minuto para ser concluída.

Por esse motivo, o UserStatus é retornado, que pode ter os seguintes valores:

- CRIADO: indica que o 'Usuário' foi criado com sucesso e nenhuma face está associada a ele atualmente. O 'usuário' estará nesse estado antes que qualquer chamada 'AssociateFaces' bem-sucedida seja feita.
- ATUALIZAÇÃO: indica que o "Usuário" está sendo atualizado para refletir as faces recém-associadas/desassociadas e se tornará ATIVO em alguns segundos. Os resultados da pesquisa podem conter "Usuário" nesse estado e os clientes podem optar por ignorá-lo nos resultados retornados.
- ATIVO: indica que o "Usuário" foi atualizado para refletir todas as faces associadas/desassociadas e está em um estado pesquisável.


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

Desassocianda faces de um usuário

Você pode usar a DisassociateFacesoperação para remover a associação entre uma ID de usuário e uma ID facial.

Para desassociar faces (SDK)

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use os exemplos a seguir para chamar a operação `DisassociateFaces`.

   Java

   Este exemplo de Java remove a associação entre um FaceID e um UserID com a operação `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

Esse AWS CLI comando remove a associação entre um FaceID e um UserID com a operação. `DisassociateFaces`

```
aws rekognition disassociate-faces --face-ids list-of-face-ids
--user-id user-id --collection-id collection-name --region region-name
```

Python

O exemplo a seguir remove a associação entre um FaceID e um UserID com a operação `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 resposta da operação

A resposta para `DisassociateFaces` inclui o `UserStatus`, que é o status da solicitação de dissociação, bem como uma lista de `FaceIds` que não devem ser associados. Uma lista de `UnsuccessfulFaceDisassociations` também é retornada. Depois de enviar uma solicitação para DisassociateFaces, a operação pode levar cerca de um minuto para ser concluída. Por esse motivo, o UserStatus é retornado, que pode ter os seguintes valores:

- CRIADO: indica que o 'Usuário' foi criado com sucesso e nenhuma face está associada a ele atualmente. O 'usuário' estará nesse estado antes que qualquer chamada 'AssociateFaces' bem-sucedida seja feita.
- ATUALIZAÇÃO: indica que o "Usuário" está sendo atualizado para refletir as faces recém-associadas/desassociadas e se tornará ATIVO em alguns segundos. Os resultados da pesquisa podem conter "Usuário" nesse estado e os clientes podem optar por ignorá-lo nos resultados retornados.
- ATIVO: indica que o "Usuário" foi atualizado para refletir todas as faces associadas/desassociadas e está em um estado pesquisável.

```
{
    "UserStatus": "UPDATING",
    "DisassociatedFaces": [
        {
            "FaceId": "c92265d4-5f9c-43af-a58e-12be0ce02bc3"
        }
    ],
    "UnsuccessfulFaceDisassociations": [
        {
            "Reasons": [
                "ASSOCIATED_TO_A_DIFFERENT_IDENTITY"
            ],
            "FaceId": "f5817d37-94f6-4335-bfee-6cf79a3d806e",
            "UserId": "demoUser1"
        }
    ]
}
```

Listando usuários em uma coleção

Você pode usar a ListUsersoperação para listar UserIds e UserStatus o. Para ver as faces associadas IDs a uma ID de usuário, use a operação. ListFaces

Listar usuários (SDK)

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use os exemplos a seguir para chamar a operação `ListUsers`.

   Java

   Este exemplo de Java lista os usuários em uma coleção usando a operação `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

Esse AWS CLI comando lista os usuários em uma coleção com a `ListUsers` operação.

```
aws rekognition list-users --collection-id collection-id --max-results number-
of-max-results
```

Python

O exemplo a seguir lista os usuários em uma coleção com a operação `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 resposta da operação

A resposta a uma solicitação ListUsers inclui uma lista dos `Users` itens da coleção junto com os `UsedId` e `UserStatus` do Usuário.

```
{
    "NextToken": "B1asJT3bAb/ttuGgPFV8BZoBZyGQzlUHXbuTNLh48a6enU7kXKw43hpOwizW7LOk/
Gk7Em09lznoq6+FcDCcSq2olrn7A98BLkt5keu+ZRVRUTyrXtT6J7Hmp
+ieQ2an6Zu0qzPfcdPeaJ9eAxG2d0WNrzJgi5hvmjoiSTTfKX3MQz1sduWQkvAAs4hZfhZoKFahFlqWofshCXa/
FHAAY3PL1PjxXbkNeSSMq8V7i1MlKCdrPVykCv9MokpPt7jtNvKPEZGUhxgBTFMxNWLEcFnzAiCWDg91dFy/
LalshPjXA9UVc5Gx9vIJNQ/
eO3cQRghAkCT3FOAiXsLAnA015ODTomZpWWVpqB21wKpI3LYmfAVFrDPGzpbTVlRmLsJm41bkmnBBBw9+DHz1Jn7zW
+qc5Fs3yaHu0f51Xg==",
    "Users": [
```

```
        {
            "UserId": "demoUser4",
            "UserStatus": "CREATED"
        },
        {
            "UserId": "demoUser2",
            "UserStatus": "CREATED"
        }
    ]
}
```

Procurando uma face com um ID facial

Você pode usar a SearchFacesoperação para pesquisar usuários em uma coleção que correspondam à maior face em uma imagem fornecida.

O ID da face é retornado na resposta da operação IndexFaces quando a face é detectada e adicionada a uma coleção. Para obter mais informações, consulte Gerenciar faces em uma coleção.

Para pesquisar uma face em uma coleção usando o ID da face (SDK)

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use os exemplos a seguir para chamar a operação `SearchFaces`.

   Java

   Este exemplo exibe informações sobre faces correspondentes a uma face identificada pelo respectivo ID.

   Altere o valor de `collectionID` para a coleção que contém a face obrigatória. Altere o valor de `faceId` para o identificador da face que você deseja encontrar.

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

```
    }

}
```

Execute o código de exemplo. As informações sobre faces correspondentes são exibidas.

Java V2

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
// 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

Esse AWS CLI comando exibe a saída JSON para a operação da `search-faces` CLI. Substitua o valor de `face-id` pelo identificador da face que você deseja pesquisar e substitua o valor de `collection-id` pela coleção que você deseja pesquisar. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
aws rekognition search-faces --face-id face-id --collection-id "collection-id"
 --profile profile-name
```

Python

Este exemplo exibe informações sobre faces correspondentes a uma face identificada pelo respectivo ID.

Altere o valor de `collectionID` para a coleção que contém a face obrigatória. Altere o valor de `faceId` para o identificador da face que você deseja encontrar. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Este exemplo exibe informações sobre faces correspondentes a uma face identificada pelo respectivo ID.

Altere o valor de `collectionID` para a coleção que contém a face obrigatória. Altere o valor de `faceId` para o identificador da face que você deseja encontrar.

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

Execute o código de exemplo. As informações sobre faces correspondentes são exibidas.

SearchFaces solicitação de operação

Com um determinado ID de face (cada face armazenada na coleção de faces tem um ID), o `SearchFaces` pesquisa a coleção de faces semelhantes na coleção de faces especificada. A resposta não inclui a face que você está procurando. Ela inclui apenas faces semelhantes. Por padrão, `SearchFaces` retorna faces nas quais o algoritmo detecta semelhança maior que 80%. A

semelhança indica o nível de proximidade da face com a face de entrada. Você também pode usar `FaceMatchThreshold` para especificar um valor diferente.

```
{
    "CollectionId": "MyCollection",
    "FaceId": "0b683aed-a0f1-48b2-9b5e-139e9cc2a757",
    "MaxFaces": 2,
    "FaceMatchThreshold": 99
}
```

SearchFaces resposta da operação

A operação retorna uma matriz de correspondências de face encontradas e o ID de face fornecido como entrada.

```
{
    "SearchedFaceId": "7ecf8c19-5274-5917-9c91-1db9ae0449e2",
    "FaceMatches": [ list of face matches found ]
}
```

Para cada correspondência de face encontrada, a resposta inclui metadados de semelhança e face, conforme mostrado na seguinte resposta de exemplo:

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

Procurando um rosto com uma imagem

Você pode usar a operação SearchFacesByImage para pesquisar faces em uma coleção que corresponda à maior face em uma imagem fornecida.

Para obter mais informações, consulte Pesquisa de faces e usuários em uma coleção.

Para pesquisar uma face em uma coleção usando uma imagem (SDK)

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Carregue uma imagem (que contenha uma ou mais faces) no bucket do S3.

   Para obter instruções, consulte Como fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

3. Use os exemplos a seguir para chamar a operação `SearchFacesByImage`.

Java

Este exemplo exibe informações sobre faces correspondentes à maior face em uma imagem. O exemplo de código especifica os parâmetros `FaceMatchThreshold` e `MaxFaces` para limitar os resultados retornados na resposta.

No exemplo a seguir, altere o seguinte: altere o valor de `collectionId` para a coleção que você deseja pesquisar, altere o valor de `bucket` para o bucket que contém a imagem de entrada e altere o valor de `photo` para a imagem de entrada.

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

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.FaceMatch;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.SearchFacesByImageRequest;
import com.amazonaws.services.rekognition.model.SearchFacesByImageResult;
import java.util.List;
import com.fasterxml.jackson.databind.ObjectMapper;


public class SearchFaceMatchingImageCollection {
    public static final String collectionId = "MyCollection";
    public static final String bucket = "bucket";
    public static final String photo = "input.jpg";

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

       AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

      ObjectMapper objectMapper = new ObjectMapper();

       // Get an image object from S3 bucket.
      Image image=new Image()
              .withS3Object(new S3Object()
                      .withBucket(bucket)
```

```
                    .withName(photo));

     // Search collection for faces similar to the largest face in the image.
     SearchFacesByImageRequest searchFacesByImageRequest = new
 SearchFacesByImageRequest()
             .withCollectionId(collectionId)
             .withImage(image)
             .withFaceMatchThreshold(70F)
             .withMaxFaces(2);

      SearchFacesByImageResult searchFacesByImageResult =
              rekognitionClient.searchFacesByImage(searchFacesByImageRequest);

      System.out.println("Faces matching largest face in image from" + photo);
     List < FaceMatch > faceImageMatches =
 searchFacesByImageResult.getFaceMatches();
     for (FaceMatch face: faceImageMatches) {
         System.out.println(objectMapper.writerWithDefaultPrettyPrinter()
                 .writeValueAsString(face));
        System.out.println();
     }
   }
}
```

Java V2

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
// 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

Esse AWS CLI comando exibe a saída JSON para a operação da `search-faces-by-image` CLI. Substitua o valor de `Bucket` pelo bucket do S3 usado na etapa 2. Substitua o valor de `Name` pelo nome de arquivo da imagem usado na etapa 2. Substitua o valor de `collection-id` pela coleção na qual você deseja pesquisar. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Se você estiver acessando a CLI em um dispositivo Windows, use aspas duplas em vez de aspas simples e escape das aspas duplas internas com barra invertida (ou seja, \) para resolver quaisquer erros de analisador que você possa encontrar. Para obter um exemplo, veja o seguinte:

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

Python

Este exemplo exibe informações sobre faces correspondentes à maior face em uma imagem. O exemplo de código especifica os parâmetros `FaceMatchThreshold` e `MaxFaces` para limitar os resultados retornados na resposta.

No exemplo a seguir, altere o seguinte: altere o valor da coleção `collectionId` que você deseja pesquisar e substitua os valores de `bucket` e `photo` pelos nomes do bucket e da imagem do Amazon S3 que você usou na etapa 2. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Este exemplo exibe informações sobre faces correspondentes à maior face em uma imagem. O exemplo de código especifica os parâmetros `FaceMatchThreshold` e `MaxFaces` para limitar os resultados retornados na resposta.

No exemplo a seguir, altere o seguinte: altere o valor da coleção `collectionId` que você deseja pesquisar e substitua os valores de `bucket` e `photo` pelos nomes do bucket e da imagem do Amazon S3 que você usou na etapa 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 solicitação de operação

Os parâmetros de entrada `SearchFacesImageByImage` são a coleção para pesquisa e o local da imagem de origem. Neste exemplo, a imagem de origem é armazenada em um bucket do Amazon S3 (`S3Object`). Também especificados estão o número máximo de faces a serem retornadas (`Maxfaces`) e o mínimo de confiança que deve ser correspondido para uma face ser retornada (`FaceMatchThreshold`).

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "MaxFaces": 2,
    "FaceMatchThreshold": 99
}
```

SearchFacesByImage resposta da operação

Dada uma determinada imagem de entrada (.jpeg ou .png), a operação primeiro detecta a face na imagem de entrada e, em seguida, pesquisa faces semelhantes na coleção de faces especificada.

> **Note**
>
> Se detectar várias faces na imagem de entrada, o serviço usará a maior face detectada para pesquisar a coleção de faces.

A operação retorna uma matriz de correspondências de face encontradas e informações sobre a face de entrada. Isso inclui informações como a caixa delimitadora, além do valor que indica o nível de confiança da caixa delimitadora que contém uma face.

Por padrão, `SearchFacesByImage` retorna faces nas quais o algoritmo detecta semelhança maior que 80%. A semelhança indica o nível de proximidade da face com a face de entrada. Você também

pode usar `FaceMatchThreshold` para especificar um valor diferente. Para cada correspondência de face encontrada, a resposta inclui metadados de semelhança e face, conforme mostrado na seguinte resposta de exemplo:


```
{
    "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
}
```


Pesquisando usuários (ID facial/ID de usuário)

Você pode usar a SearchUsersoperação para pesquisar usuários em uma coleção especificada que correspondam a uma ID facial ou ID de usuário fornecida. A operação lista os retornados `UserIds` classificados pela pontuação de similaridade mais alta acima da solicitada UserMatchThreshold. O ID do usuário é criado na CreateUsers operação. Para obter mais informações, consulte Gerenciar usuários em uma coleção.

Para pesquisar usuários (SDK)

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use os exemplos a seguir para chamar a operação `SearchUsers`.

   Java

   Este exemplo de Java pesquisa os usuários em uma coleção usando a operação `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

Esse AWS CLI comando pesquisa os usuários em uma coleção com a `SearchUsers` operação.

```
aws rekognition search-users --face-id face-id --collection-id collection-id --
region region-name
```

Python

O exemplo a seguir pesquisa os usuários em uma coleção com a operação `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 solicitação de operação

Com um FaceID ou ID de usuário SearchUsers , pesquisa correspondências de usuário no CollectionID especificado. Por padrão, SearchUsers retorna Usuário IDs para o qual a pontuação de similaridade é maior que 80%. A semelhança indica até que ponto o UserID corresponde ao FaceID ou UserID fornecido. Se vários usuários IDs forem retornados, eles serão listados na ordem da maior pontuação de similaridade para a menor. Opcionalmente, você pode usar o UserMatchThreshold para especificar um valor diferente. Para obter mais informações, consulte Gerenciar usuários em uma coleção.

Veja a seguir um exemplo de uma SearchUsers solicitação usando`UserId`:

```
{
   "CollectionId": "MyCollection",
```

```
   "UserId": "demoUser1",
   "MaxUsers": 2,
   "UserMatchThreshold": 99
}
```

Veja a seguir um exemplo de uma SearchUsers solicitação usando`FaceId`:

```
{
    "CollectionId": "MyCollection",
    "FaceId": "bff43c40-cfa7-4b94-bed8-8a08b2205107",
    "MaxUsers": 2,
    "UserMatchThreshold": 99
}
```

SearchUsers resposta da operação

Se pesquisar com um`FaceId`, a resposta para SearchUsers inclui o `FaceId` para`SearchedFace`, bem como uma lista de `UserMatches` e o `UserId` e `UserStatus` para cada usuário.

```
{
    "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 estiver pesquisando com a`UserId`, a resposta `UserId` para SearchUsers inclui a para`SearchedUser`, além dos outros elementos de resposta.

```
{
    "SearchedUser": {
        "UserId": "demoUser1"
    },
    "UserMatches": [
        {
            "User": {
                "UserId": "demoUser2",
                "UserStatus": "ACTIVE"
            },
            "Similarity": 99.97946166992188
        }
    ],
    "FaceModelVersion": "6"
}
```

Pesquisar usuários (imagem)

`SearchUsersByImage`pesquisa o CollectionID especificado em busca de usuários em uma coleção que correspondam ao maior rosto detectado em uma imagem fornecida. Por padrão, SearchUsersByImage retorna Usuário IDs para o qual a pontuação de similaridade é maior que 80%. A semelhança indica até que ponto a ID de usuário corresponde à maior face detectada na imagem fornecida. Se vários usuários IDs forem retornados, eles serão listados na ordem da maior pontuação de similaridade para a menor. Opcionalmente, você pode usar o UserMatchThreshold para especificar um valor diferente. Para obter mais informações, consulte the section called “Gerenciar usuários em uma coleção”.

Para pesquisar usuários por imagem (SDK)

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use os exemplos a seguir para chamar a operação `SearchUsersByImage`.

   Java

   Este exemplo de Java pesquisa os usuários em uma coleção com base em uma imagem de entrada, usando a operação `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

Esse AWS CLI comando pesquisa os usuários em uma coleção com base em uma imagem de entrada, com a `SearchUsersByImage` operação.

```
aws rekognition search-users-by-image --image '{"S3Object":
{"Bucket":"s3BucketName","Name":"file-name"}}' --collection-id MyCollectionId --
region region-name
```

Python

O exemplo a seguir pesquisa os usuários em uma coleção com base em uma imagem de entrada, com a operação `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 solicitação de operação

A solicitação `SearchUsersByImage` inclui a coleção a ser pesquisada e a localização da imagem de origem. Neste exemplo, a imagem de origem é armazenada em um bucket do Amazon S3 (`S3Object`). Também são especificados o número máximo de usuários a serem retornados (`MaxUsers`) e a confiança mínima que deve ser correspondida para que um usuário seja retornado (`UserMatchThreshold`).

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "MaxUsers": 2,
    "UserMatchThreshold": 99
}
```

SearchUsersByImage resposta da operação

A resposta para `SearchUsersByImage` inclui um `FaceDetail` objeto para o`SearchedFace`, bem como uma lista de UserMatches com o `UserIdSimilarity`, e `UserStatus` para cada um. Se a imagem de entrada contiver mais de uma face, uma lista de também UnsearchedFaces será retornada.

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

Pesquisanda faces em vídeos armazenados

Você pode pesquisar uma coleção de faces que corresponda a faces de pessoas detectadas em um vídeo armazenado ou em um vídeo de streaming. Esta seção aborda a pesquisa de faces em um vídeo armazenado. Para obter informações sobre a pesquisa de faces em um vídeo de streaming, consulte Trabalhando com eventos de streaming de vídeo.

As faces que você pesquisa devem primeiro ser indexadas em uma coleção usando IndexFaces. Para obter mais informações, consulte Adicionar faces a uma coleção.

A pesquisa facial do Amazon Rekognition Video segue o mesmo fluxo de trabalho assíncrono de outras operações de vídeo do Amazon Rekognition Video que analisam vídeos armazenados em um bucket do Amazon S3. Para começar a pesquisar rostos em um vídeo armazenado, ligue StartFaceSearche forneça o ID da coleção que você deseja pesquisar. O Amazon Rekognition Video publica o status de conclusão da análise de vídeo em um tópico do Amazon Simple Notification Service (Amazon SNS). Se a análise do vídeo for bem-sucedida, chame GetFaceSearch para obter os resultados de pesquisa. Para obter mais informações sobre como iniciar uma análise de vídeo e obter os resultados, consulte Chamando as operações de vídeo do Amazon Rekognition Video.

O procedimento a seguir mostra como pesquisar uma coleção de faces que correspondem às faces de pessoas detectadas em um vídeo. O procedimento também mostra como obter o rastreamento de dados das pessoas correspondidas no vídeo. O procedimento expande o código em Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK), que usa uma fila do Amazon Simple Queue Service (Amazon SQS) para obter o status de conclusão de uma solicitação de análise de vídeo.

Para pesquisar faces correspondentes em um vídeo (SDK)

1. Crie uma coleção.
2. Indexe uma face na coleção.
3. Execute Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK).
4. Adicione o código a seguir à classe `VideoDetect` criada por você na etapa 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);

  }
```

Na função `main`, substitua as linhas:

```
       StartLabelDetection(bucket, video);

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

por:

```
       String collection="collection";
       StartFaceSearchCollection(bucket, video, collection);

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

Java V2

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
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()
```

Na função `main`, substitua as linhas:

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

por:

```
    collection='tests'
    analyzer.StartFaceSearchCollection(collection)

    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetFaceSearchCollectionResults()
```

Se você já tiver executado um exemplo de vídeo diferente de Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK), o código a ser substituído poderá ser diferente.

5. Altere o valor de `collection` para o nome da coleção criada por você na etapa 1.
6. Execute o código. Uma lista de pessoas no vídeo cujas faces correspondem às da coleção de entrada é exibida. Os dados de rastreamento de cada pessoa correspondida também são exibidos.

GetFaceSearch resposta da operação

Esta é uma resposta JSON de exemplo de `GetFaceSearch`.

A resposta inclui uma matriz das pessoas (`Persons`) detectadas no vídeo cujas faces correspondem a uma face na coleção de entrada. A cada vez que a pessoa é correspondida no vídeo, existe um elemento da matriz, PersonMatch. Cada `PersonMatch` inclui uma matriz de correspondências de faces da coleção de entrada, FaceMatch, informações sobre a pessoa correspondente, PersonDetail, e a hora em que a pessoa foi correspondida no vídeo.


```
{
    "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
    }
}
```

Pesquisando faces em uma coleção em um vídeo de streaming

Você pode usar o Amazon Rekognition Video para detectar e reconhecer faces de uma coleção em streaming de vídeo. Com o Amazon Rekognition Video, você pode criar um CreateStreamProcessorprocessador de streaming () para iniciar e gerenciar a análise do streaming de vídeo.

Para detectar um rosto conhecido em um stream de vídeo (pesquisa facial), o Amazon Rekognition Video usa o Amazon Kinesis Video Streams para receber e processar um stream de vídeo. Os resultados da análise são enviados do Amazon Rekognition Video para um fluxo de dados do Kinesis e depois lidos pelo seu aplicativo cliente.

Para usar o Amazon Rekognition Video com streaming de vídeo, seu aplicativo precisa implementar o seguinte:

- Um stream de vídeo do Kinesis para enviar streaming de vídeo para o Amazon Rekognition Video. Para obter mais informações, consulte o Guia do desenvolvedor do Amazon Kinesis Video Streams.
- Um processador de stream do Amazon Rekognition Video para gerenciar a análise do streaming de vídeo. Para obter mais informações, consulte Visão geral das operações do processador de stream do Amazon Rekognition Video.
- Um consumidor de fluxo de dados do Kinesis para ler os resultados da análise que o Amazon Rekognition Video envia para o fluxo de dados do Kinesis. Para obter mais informações, consulte Consumidores do Kinesis Data Streams.

Esta seção contém informações sobre como escrever um aplicativo que cria o stream de vídeo do Kinesis e outros recursos necessários, transmite vídeos para o Amazon Rekognition Video e recebe os resultados da análise.

Tópicos

- Configurando seus recursos do Amazon Rekognition Video e do Amazon Kinesis
- Pesquisanda faces em um streaming de vídeo
- Streaming usando um GStreamer plug-in
- Solução de problemas de streaming de vídeo

Configurando seus recursos do Amazon Rekognition Video e do Amazon Kinesis

Os procedimentos a seguir descrevem as etapas que você executa para provisionar o stream de vídeo do Kinesis e outros recursos usados para reconhecer faces em um streaming de vídeo.

Pré-requisitos

Para executar esse procedimento, você precisa ter o AWS SDK para Java instalado. Para obter mais informações, consulte Comece a usar o Amazon Rekognition. O Conta da AWS que você usa deve ter permissões de acesso à API do Amazon Rekognition. Para obter mais informações, consulte Ações definidas pelo Amazon Rekognition no Guia do usuário do IAM.

Para reconhecer faces em um fluxo de vídeo (AWS SDK)

1. Se você ainda não o fez, crie um perfil de serviço do IAM para dar ao Amazon Rekognition Video acesse aos seus streams de vídeo do Kinesis e aos seus streams de dados do Kinesis. Anote o ARN. Para obter mais informações, consulte Dando acesso a streams usando AmazonRekognitionServiceRole.
2. Crie uma coleção e anote o identificador da coleção que você usou.
3. Faça a indexação das faces que deseja pesquisar na coleção que você criou na etapa 2.
4. Crie um stream de vídeo do Kinesis e anote o nome de recurso da Amazon (ARN) do stream.
5. Crie um fluxo de dados do Kinesis. Prefixe o nome do stream AmazonRekognitione anote o ARN do stream.

Em seguida, você pode criar o processador de stream de pesquisa facial e iniciar o processador de stream usando o nome do processador de stream que você escolheu.

> **Note**
>
> Você deve iniciar o processador de stream somente depois de verificar se você pode ingerir mídia no stream de vídeo do Kinesis.

Streaming de vídeo para o Amazon Rekognition Video

Para transmitir vídeo para o Amazon Rekognition Video, você usa o SDK do Amazon Kinesis Video Streams para criar e usar um stream de vídeo do Kinesis. A operação `PutMedia` grava fragmentos de dados de vídeo em um stream de vídeo do Kinesis que o Amazon Rekognition Video consome. Cada fragmento de dados de vídeo geralmente tem de 2 a 10 segundos de duração e contém uma sequência independente de quadros de vídeo. O Amazon Rekognition Video oferece suporte a vídeos codificados em H.264, que podem ter três tipos de quadros (I, B e P). Para obter mais informações, consulte Inter Frame. O primeiro quadro no fragmento deve ser um I-frame. Um I-frame pode ser decodificado independentemente de qualquer outro quadro.

Quando os dados de vídeo chegam ao stream de vídeo do Kinesis, o Kinesis Video Streams atribui um número exclusivo ao fragmento. Para ver um exemplo, consulte Exemplo de PutMedia API.

- Se você estiver transmitindo de uma fonte codificada em Matroska (MKV), use a PutMediaoperação para transmitir o vídeo de origem para o stream de vídeo do Kinesis que você criou. Para obter mais informações, consulte Exemplo de PutMedia API.
- Se você estiver transmitindo da câmera de um dispositivo, consulte Streaming usando um GStreamer plug-in.

Conceder ao Amazon Rekognition Video acesso aos seus recursos

Você usa uma função de serviço AWS Identity and Access Management (IAM) para dar ao Amazon Rekognition Video acesso de leitura aos streams de vídeo do Kinesis. Se você estiver usando um processador de stream de pesquisa facial, você usa um perfil de serviço do IAM para dar ao Amazon Rekognition Video acesse de gravação aos fluxos de dados do Kinesis. Se você estiver usando um processador de stream de monitoramento de segurança, use os perfis do IAM para dar ao Amazon Rekognition Video acesso ao bucket do Amazon S3 e a um tópico do Amazon SNS.

Fornecer acesso a processadores de stream de pesquisa facial

Você pode criar uma política de permissões que permita ao Amazon Rekognition Video acessar streams de vídeo individuais do Kinesis e streams de dados do Kinesis.

Para dar ao Amazon Rekognition Video acesso a um processador de stream de pesquisa facial

1. Crie uma nova política de permissões com o editor de políticas IAM JSON e use a política a seguir. Substitua `video-arn` pelo ARN do stream de vídeo do Kinesis desejado. Se você estiver usando um processador de stream de pesquisa facial, substitua o `data-arn` pelo ARN do fluxo de dados Kinesis desejado.

```
{
    "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. Crie um perfil de serviço do IAM ou atualize um perfil de serviço do IAM existente. Use as informações a seguir para criar o perfil de serviço do IAM:

    1. Escolha Rekognition como o nome de serviço.
    2. Escolha Rekognition para o caso de uso da função de serviço.
    3. Anexe a política de permissões que você criou na etapa 1.

3. Anote o ARN da função de serviço. Ele é necessário para iniciar as operações de análise de vídeo.

Dando acesso a streams usando AmazonRekognitionServiceRole

Como opção alternativa para configurar o acesso aos streams de vídeo e fluxos de dados do Kinesis, você pode usar a política de permissões da `AmazonRekognitionServiceRole`. O IAM fornece o caso de uso do perfil de serviço Rekognition que, quando usado com a política de permissões `AmazonRekognitionServiceRole`, pode gravar em vários streams de dados do Kinesis e ler todos os seus streams de vídeo do Kinesis. Para dar ao Amazon Rekognition Video acesso de gravação a vários streams de dados do Kinesis, você pode prefixar os nomes dos streams de dados do Kinesis com —por exemplo,. AmazonRekognition`AmazonRekognitionMyDataStreamName`

Para dar ao Amazon Rekognition Video acesso ao seu stream de vídeo do Kinesis e ao fluxo de dados do Kinesis

1. Crie um perfil de serviço do IAM. Use as informações a seguir para criar o perfil de serviço do IAM:

   1. Escolha Rekognition como o nome de serviço.
   2. Escolha Rekognition para o caso de uso da função de serviço.
   3. Escolha a política de AmazonRekognitionServiceRolepermissões, que dá ao Amazon Rekognition Video acesso de gravação aos streams de dados do Kinesis AmazonRekognitionprefixados e acesso de leitura a todos os seus streams de vídeo do Kinesis.
2. Para garantir que você Conta da AWS esteja seguro, limite o escopo do acesso do Rekognition apenas aos recursos que você está usando. Isso pode ser feito anexando uma política de confiança à suo perfil de serviço do IAM. Para obter informações sobre como fazer isso, consulte Prevenção do problema do substituto confuso entre serviços.
3. Observe o nome de recurso da Amazon (ARN) da função de serviço. Ele é necessário para iniciar as operações de análise de vídeo.

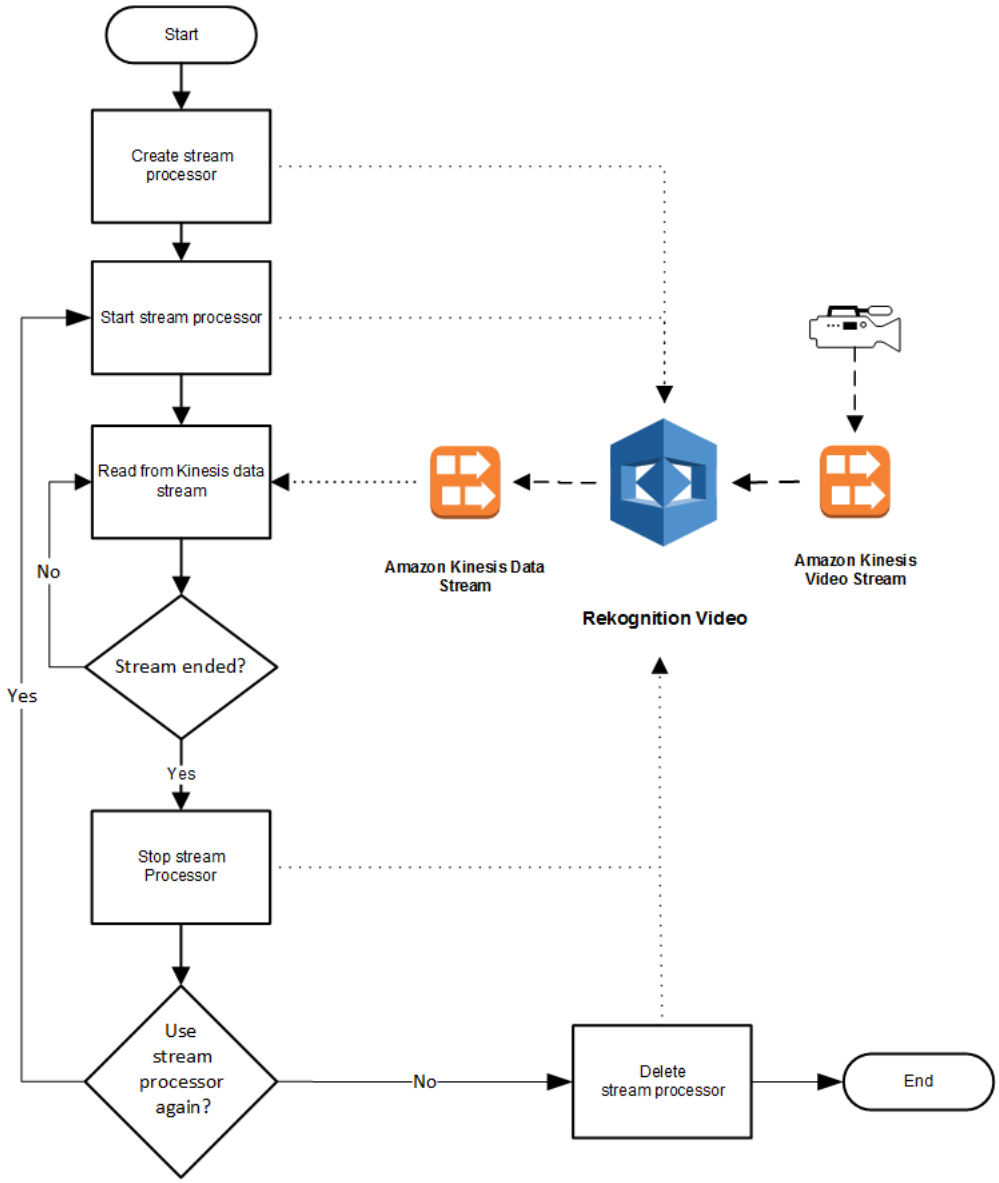
Pesquisanda faces em um streaming de vídeo

O Amazon Rekognition Video pode pesquisar faces em uma coleção que correspondam às faces detectadas em um vídeo de streaming. Para obter mais informações sobre coleções, consulte Pesquisa de faces em uma coleção.

Tópicos

- Criação do processador de stream de pesquisa facial Amazon Rekognition Video
- Iniciando o processador de stream de pesquisa facial do Amazon Rekognition Video
- Usando processadores de stream para pesquisa facial (exemplo de Java V2)
- Usando processadores de stream para pesquisa facial (exemplo de Java V1)
- Lendo os resultados da análise de streaming de vídeo
- Exibindo resultados do Rekognition com o Kinesis Video Streams localmente
- Entender o registro de quadros JSON de reconhecimento facial do Kinesis

O diagrama a seguir mostra como o Amazon Rekognition Video detecta e reconhece faces em um streaming de vídeo.



Criação do processador de stream de pesquisa facial Amazon Rekognition Video

Antes de poder analisar um streaming de vídeo, você cria um processador de streaming de vídeo Amazon Rekognition Video (). CreateStreamProcessor O processador de stream contém informações sobre o fluxo de dados do Kinesis e o stream de vídeo do Kinesis. Ele também contém o identificador para a coleção que contém as faces que você deseja reconhecer no streaming de vídeo de entrada. Você também especifica um nome para o processador de fluxo. Veja a seguir um exemplo de JSON para a solicitação `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
                }
        }
}
```

Esta é uma resposta de exemplo de `CreateStreamProcessor`.

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

Iniciando o processador de stream de pesquisa facial do Amazon Rekognition Video

Inicie a análise de streaming de vídeo chamando StartStreamProcessor com o nome do processador de fluxo que você especificou em `CreateStreamProcessor`. Veja a seguir um exemplo de JSON para a solicitação `StartStreamProcessor`.

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

Se o processador de fluxo iniciar com êxito, uma resposta HTTP 200 é retornada, junto com um corpo JSON vazio.

Usando processadores de stream para pesquisa facial (exemplo de Java V2)

O código de exemplo a seguir mostra como chamar várias operações do processador de stream, como CreateStreamProcessore StartStreamProcessor, usando o AWS SDK for Java versão 2.

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
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.");
    }
}
```

Usando processadores de stream para pesquisa facial (exemplo de Java V1)

O código de exemplo a seguir mostra como chamar várias operações do processador de fluxo, como CreateStreamProcessore StartStreamProcessor, usando o Java V1. O exemplo inclui uma classe de gerenciador de processador de fluxo (StreamManager) que fornece métodos para chamar operações do processador de fluxo. A classe inicial (Starter) cria um StreamManager objeto e chama várias operações.

Para configurar o exemplo:

1. Defina os valores dos campos de membro de classe Starter para seus valores desejados.
2. Na função de classe Starter `main`, exclua a função chamada de função desejada.

Classe inicial

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

Lendo os resultados da análise de streaming de vídeo

Você pode usar a biblioteca de cliente do Amazon Kinesis Data Streams para consumir os resultados da análise que são enviados para o stream de saída do Amazon Kinesis Data Streams. Para obter mais informações, consulte Ler dados de um Kinesis Data Stream. O Amazon Rekognition Video coloca um registro de quadro JSON para cada quadro analisado no stream de saída do Kinesis. O Amazon Rekognition Video não analisa cada quadro que é passado para ele pelo stream de vídeo do Kinesis.

Um registro de quadro enviado para um fluxo de dados do Kinesis contém informações sobre em qual fragmento de stream de vídeo do Kinesis o quadro está, onde o quadro está no fragmento e faces que são reconhecidas no quadro. Ele também inclui informações de status para o processador de fluxo. Para obter mais informações, consulte Entender o registro de quadros JSON de reconhecimento facial do Kinesis.

A biblioteca do Amazon Kinesis Video Streams Parser contém exemplos de testes que consomem os resultados do Amazon Rekognition Video e os integram ao stream de vídeo original do Kinesis. Para obter mais informações, consulte Exibindo resultados do Rekognition com o Kinesis Video Streams localmente.

O Amazon Rekognition Video transmite informações de análise do Amazon Rekognition Video para o fluxo de dados do Kinesis. Veja a seguir um exemplo de JSON para um único registro.


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


No exemplo de JSON, observe:

- InputInformation— Informações sobre o stream de vídeo do Kinesis usado para transmitir vídeo para o Amazon Rekognition Video. Para obter mais informações, consulte InputInformation.
- StreamProcessorInformation— Informações de status do processador de streaming de vídeo Amazon Rekognition Video. O único valor possível para o campo `Status` é RUNNING. Para obter mais informações, consulte StreamProcessorInformation.
- FaceSearchResponse— Contém informações sobre rostos no streaming de vídeo que correspondem aos rostos na coleção de entrada. FaceSearchResponsecontém um DetectedFace objeto, que é uma face que foi detectada no quadro de vídeo analisado. Para cada face detectada, a matriz `MatchedFaces` contém uma matriz de objetos de faces correspondentes (MatchedFace) encontradas na coleção de entrada, juntamente com uma pontuação de semelhança.

Mapeando o stream de vídeo do Kinesis para o fluxo de dados do Kinesis

Talvez você queira mapear os quadros do stream de vídeo do Kinesis para os quadros analisados que são enviados para o fluxo de dados do Kinesis. Por exemplo, durante a exibição de um streaming de vídeo, você pode querer exibir caixas ao redor das faces de pessoas reconhecidas. As coordenadas da caixa delimitadora são enviadas como parte do Registro de Reconhecimento Facial do Kinesis para o fluxo de dados do Kinesis. Para exibir a caixa delimitadora corretamente, você precisa mapear as informações de tempo enviadas com o Kinesis Face Recognition Record com os quadros correspondentes no stream de vídeo de origem do Kinesis.

A técnica usada para mapear o stream de vídeo do Kinesis para o fluxo de dados do Kinesis depende se você está transmitindo mídia ao vivo (como um vídeo ao vivo) ou se você está transmitindo mídia arquivada (como um vídeo armazenado).

Mapeamento quando você está transmitindo mídia ao vivo

Para mapear um quadro de stream de vídeo do Kinesis para um quadro de fluxo de dados do Kinesis

1. Defina o parâmetro `FragmentTimeCodeType` de entrada da PutMediaoperação como`RELATIVE`.
2. Chame `PutMedia` para entregar mídia ao vivo no stream de vídeo do Kinesis.
3. Ao receber um registro de reconhecimento facial do Kinesis do fluxo de dados do Kinesis, armazene os valores de `ProducerTimestamp` e `FrameOffsetInSeconds` do campo KinesisVideo.

4. Calcule o carimbo de data/hora que corresponde ao quadro do stream de vídeo do Kinesis adicionando os valores de `FrameOffsetInSeconds` e do campo `ProducerTimestamp` juntos.

Mapeando quando você está transmitindo mídia arquivada

Para mapear um quadro de stream de vídeo do Kinesis para um quadro de fluxo de dados do Kinesis

1. Ligue PutMediapara entregar mídia arquivada ao stream de vídeo do Kinesis.
2. Quando você receber um objeto `Acknowledgement` da resposta da operação `PutMedia`, armazene o valor do campo `FragmentNumber` do campo Payload (Carga). `FragmentNumber` é o número do fragmento para o cluster do MKV.
3. Ao receber um Kinesis Face Recognition Record do fluxo de dados do Kinesis, armazene o valor do campo `FrameOffsetInSeconds` no campo KinesisVideo.
4. Calcule o mapeamento usando os valores `FragmentNumber` e `FrameOffsetInSeconds` que você armazenou nas etapas 2 e 3. `FrameOffsetInSeconds` é o deslocamento no fragmento com o `FragmentNumber` específico que é enviado para o fluxo de dados do Amazon Kinesis. Para obter mais informações sobre como obter os quadros de vídeo para um determinado número de fragmento, consulte Mídia arquivada do Amazon Kinesis Video Streams.

Exibindo resultados do Rekognition com o Kinesis Video Streams localmente

Você pode ver os resultados do Amazon Rekognition Video exibidos em seu feed do Amazon Kinesis Video Streams usando os exemplos de testes da Amazon Kinesis Video Streams Parser Library fornecidos em - Rekognition Examples. KinesisVideo Ele `KinesisVideoRekognitionIntegrationExample` exibe caixas delimitadoras sobre os rostos detectados e renderiza o vídeo localmente. JFrame Esse processo pressupõe que você tenha conectado com êxito uma entrada de mídia da câmera de um dispositivo a um stream de vídeo do Kinesis e iniciado um processador Amazon Rekognition Stream. Para obter mais informações, consulte Streaming usando um GStreamer plug-in.

Etapa 1: Instalando a biblioteca do Kinesis Video Streams Parser

Para criar um diretório e baixar o repositório Github, execute o seguinte comando:

```
$ git clone https://github.com/aws/amazon-kinesis-video-streams-parser-library.git
```

Navegue até o diretório da biblioteca e execute o seguinte comando do Maven para realizar uma instalação limpa:

```
$ mvn clean install
```

Etapa 2: Configurando o teste de exemplo de integração entre Kinesis Video Streams e Rekognition

Abra o arquivo `KinesisVideoRekognitionIntegrationExampleTest.java`. Remova `@Ignore` a direita após o cabeçalho da classe. Preencha os campos de dados com as informações dos seus recursos do Amazon Kinesis e do Amazon Rekognition. Para obter mais informações, consulte Configurando seus recursos do Amazon Rekognition Video e do Amazon Kinesis. Se você estiver transmitindo vídeo para seu stream de vídeo do Kinesis, remova o parâmetro `inputStream`.

Veja o exemplo de código a seguir:

```
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();
```

Etapa 3: Executando o teste de exemplo de integração entre Kinesis Video Streams e Rekognition

Certifique-se de que seu stream de vídeo do Kinesis esteja recebendo entrada de mídia se você estiver transmitindo para ele e comece a analisar seu stream com um processador Amazon Rekognition Video Stream em execução. Para obter mais informações, consulte Visão geral das operações do processador de stream do Amazon Rekognition Video. Execute a `KinesisVideoRekognitionIntegrationExampleTest` aula como um JUnit teste. Depois de um pequeno atraso, uma nova janela é aberta com um feed de vídeo do seu stream de vídeo do Kinesis com caixas delimitadoras desenhadas sobre faces detectadas.

> **Note**
>
> As faces na coleção usada neste exemplo devem ter a ID de imagem externa (o nome do arquivo) especificada nesse formato para que os rótulos das caixas delimitadoras exibam texto significativo: PersonName 1-Confiável, PersonName 2-Intruso, 3-Neutro, etc. PersonName Os rótulos também podem ser codificados por cores e são personalizáveis no arquivo.java. FaceType

Entender o registro de quadros JSON de reconhecimento facial do Kinesis

O Amazon Rekognition Video pode reconhecer faces em um streaming de vídeo. Para cada quadro analisado, o Amazon Rekognition Video gera um registro de quadro JSON em um fluxo de dados do Kinesis. O Amazon Rekognition Video não analisa cada quadro que é passado para ele pelo stream de vídeo do Kinesis.

O registro de quadros JSON contém informações sobre o stream de entrada e de saída, o status do processador de fluxo e informações sobre faces que são reconhecidas no quadro analisado. Esta seção contém informações de referência para o registro de quadros JSON.

A seguir está a sintaxe JSON para um registro de fluxo de dados do Kinesis. Para obter mais informações, consulte Trabalhando com eventos de streaming de vídeo.

> **Note**
>
> A API Amazon Rekognition Video funciona comparando as faces em seu stream de entrada com uma coleção de faces e retornando as correspondências mais próximas encontradas, junto com uma pontuação de similaridade.

```
{
    "InputInformation": {
        "KinesisVideo": {
            "StreamArn": "string",
            "FragmentNumber": "string",
            "ProducerTimestamp": number,
            "ServerTimestamp": number,
            "FrameOffsetInSeconds": number
        }
    },
    "StreamProcessorInformation": {
        "Status": "RUNNING"
    },
    "FaceSearchResponse": [
        {
            "DetectedFace": {
                "BoundingBox": {
                    "Width": number,
                    "Top": number,
                    "Height": number,
                    "Left": number
                },
                "Confidence": number,
                "Landmarks": [
                    {
                        "Type": "string",
                        "X": number,
                        "Y": number
                    }
                ],
                "Pose": {
                    "Pitch": number,
                    "Roll": number,
                    "Yaw": number
                },
```

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

Registro JSON

O registro JSON inclui informações sobre um quadro processado pelo Amazon Rekognition Video. O registro inclui informações sobre o streaming de vídeo, o status do quadro analisado e informações sobre faces que são reconhecidas no quadro.

InputInformation

Informações sobre o stream de vídeo do Kinesis usado para transmitir vídeo para o Amazon Rekognition Video.

Tipo: objeto InputInformation

StreamProcessorInformation

Informações sobre o processador de stream do Amazon Rekognition Video. Isso inclui informações de status para o status atual do processador de fluxo.

Tipo: objeto StreamProcessorInformation

FaceSearchResponse

Informações sobre as faces detectadas em um quadro de streaming de vídeo e as faces correspondentes encontradas na coleção de entrada.

Tipo: matriz de objetos FaceSearchResponse

InputInformation

Informações sobre um stream de vídeo de origem usado pelo Amazon Rekognition Video. Para obter mais informações, consulte Trabalhando com eventos de streaming de vídeo.

KinesisVideo

Tipo: objeto KinesisVideo

KinesisVideo

Informações sobre o stream de vídeo do Kinesis que transmite o vídeo de origem para o Amazon Rekognition Video. Para obter mais informações, consulte Trabalhando com eventos de streaming de vídeo.

StreamArn

O nome de recurso da Amazon (ARN) do stream de vídeo do Kinesis.

Tipo: String

FragmentNumber

O fragmento de streaming de vídeo que contém o quadro que esse registro representa.

Tipo: String

ProducerTimestamp

O time stamp do Unix do lado do produtor do fragmento. Para obter mais informações, consulte PutMedia.

Tipo: número

ServerTimestamp

O time stamp do Unix do lado do servidor do fragmento. Para obter mais informações, consulte PutMedia.

Tipo: número

FrameOffsetInSeconds

O deslocamento do quadro (em segundos) dentro do fragmento.

Tipo: número

StreamProcessorInformation

Informações de status sobre o processador de fluxo.

Status

O status atual do processador de fluxo. O único valor possível é RUNNING.

Tipo: String

FaceSearchResponse

Informações sobre uma face detectada em um quadro de streaming de vídeo e as faces em uma coleção que correspondem à face detectada. Você especifica a coleção em uma chamada para CreateStreamProcessor. Para obter mais informações, consulte Trabalhando com eventos de streaming de vídeo.

DetectedFace

Detalhes de uma face detectada em um quadro de vídeo analisado.

Tipo: objeto DetectedFace

MatchedFaces

Uma matriz dos detalhes de faces em uma coleção que corresponda à face detectada na `DetectedFace`.

Tipo: matriz de objetos MatchedFace

DetectedFace

Informações sobre uma face detectada em um quadro de streaming de vídeo. As faces correspondentes na coleção de entrada estão disponíveis no campo do objeto MatchedFace.

BoundingBox

A caixa delimitadora coordena para uma face que é detectada dentro de um quadro de vídeo analisado. O BoundingBox objeto tem as mesmas propriedades do BoundingBox objeto usado para análise de imagem.

Tipo: objeto BoundingBox

Confiança

O nível de confiança (1-100) que o Amazon Rekognition Video tem de que a face detectada é, na verdade, uma face. 1 é a confiança mais baixa, 100 é a mais alta.

Tipo: número

Pontos de referência

Uma matriz de pontos de referência faciais.

Tipo: Matriz de objetos de pontos de referência

Pose

Indica a pose da face conforme determinada pelos seus eixos lateral, vertical e longitudinal.

Tipo: Objeto de pose

Qualidade

Identifica o brilho a nitidez da imagem da face.

Tipo: objeto ImageQuality

MatchedFace

Informações sobre uma face que corresponde a uma face detectada em um quadro de vídeo analisado.

Rosto

Informações sobre a correspondência de uma face na coleção de entrada que corresponde à face no objeto DetectedFace.

Tipo: Objeto Face

Semelhança

O nível de confiança (1-100) com o qual as faces coincidem. 1 é a confiança mais baixa, 100 é a mais alta.

Tipo: número

Streaming usando um GStreamer plug-in

O Amazon Rekognition Video pode analisar uma transmissão de vídeo ao vivo a partir da câmera de um dispositivo. Para acessar a entrada de mídia de uma fonte de dispositivo, você precisa instalar GStreamer. GStreamer é um software de estrutura multimídia de terceiros que conecta fontes de mídia e ferramentas de processamento em canais de fluxo de trabalho. Você também precisa instalar o Plug-in de produtor do Amazon Kinesis Video Streams para Gstreamer. Esse processo pressupõe que você tenha configurado com êxito seus recursos do Amazon Rekognition Video e do Amazon Kinesis. Para obter mais informações, consulte Configurando seus recursos do Amazon Rekognition Video e do Amazon Kinesis.

Etapa 1: Instale o Gstreamer

Baixe e instale o Gstreamer, um software de plataforma multimídia de terceiros. Você pode usar um software de gerenciamento de pacotes como o Homebrew (Gstreamer no Homebrew) ou obtê-lo diretamente no site do Freedesktop.

Verifique a instalação bem-sucedida do Gstreamer iniciando um feed de vídeo com uma fonte de teste do seu terminal de linha de comando.

```
$ gst-launch-1.0 videotestsrc ! autovideosink
```

Etapa 2: Instale o plug-in Kinesis Video Streams Producer

Nesta seção, você fará o download da Amazon Kinesis Video Streams Producer Library e instalará o plug-in Kinesis Video Streams Gstreamer.

Crie um diretório e clone o código-fonte do repositório Github. Certifique-se de incluir o parâmetro `--recursive`.

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

Siga as instruções fornecidas pela biblioteca para configurar e criar o projeto. Certifique-se de usar os comandos específicos da plataforma para seu sistema operacional. Use o parâmetro `-DBUILD_GSTREAMER_PLUGIN=ON` ao executar `cmake` para instalar o plug-in Kinesis Video Streams Gstreamer. Esse projeto requer os seguintes pacotes adicionais que estão incluídos na instalação: GCC ou Clang, Curl, Openssl e Log4cplus. Se sua compilação falhar devido à falta de um pacote, verifique se o pacote está instalado e em seu PATH. Se você encontrar um erro "não é possível executar o programa compilado em C" durante a compilação, execute o comando de compilação novamente. Às vezes, o compilador C correto não é encontrado.

Verifique a instalação do plug-in Kinesis Video Streams executando o comando a seguir.

```
$ gst-inspect-1.0 kvssink
```

As informações a seguir, como detalhes de fábrica e do plug-in, devem aparecer:

```
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/

  ...
```

Etapa 3: Execute o Gstreamer com o plugin Kinesis Video Streams

Antes de começar a transmitir da câmera de um dispositivo para o Kinesis Video Streams, talvez seja necessário converter a fonte de mídia em um codec aceitável para o Kinesis Video Streams. Para determinar as especificações e os recursos de formato dos dispositivos atualmente conectados à sua máquina, execute o comando a seguir.

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

Para começar a transmitir, inicie o Gstreamer com o comando de exemplo a seguir e adicione suas credenciais e informações do Amazon Kinesis Video Streams. Você deve usar as chaves de acesse e a região para o perfil de serviço do IAM que você criou ao conceder ao Amazon Rekognition acesse aos seus streams do Kinesis. Para obter mais informações sobre chaves de acesso, consulte Gerenciando chaves de acesso para usuários do IAM no Guia do usuário do IAM. Além disso, você pode ajustar os parâmetros do argumento do formato de vídeo conforme exigido pelo seu uso e disponíveis no seu dispositivo.

```
$ gst-launch-1.0 autovideosrc device=/dev/video0 ! videoconvert ! video/x-
raw,format=I420,width=640,height=480,framerate=30/1 !
                x264enc bframes=0 key-int-max=45 bitrate=500 ! video/x-h264,stream-
format=avc,alignment=au,profile=baseline !
                kvssink stream-name="YOUR_STREAM_NAME" storage-size=512 access-
key="YOUR_ACCESS_KEY" secret-key="YOUR_SECRET_ACCESS_KEY" aws-region="YOUR_AWS_REGION"
```

Para obter mais comandos de inicialização, consulte Exemplos de comandos de GStreamer inicialização.

> **Note**
>
> Se o comando de inicialização terminar com um erro de não negociação, verifique a saída do Device Monitor e certifique-se de que os valores dos parâmetros `videoconvert` sejam recursos válidos do seu dispositivo.

Você verá um feed de vídeo da câmera do seu dispositivo no stream de vídeo do Kinesis após alguns segundos. Para começar a detectar e combinar faces com o Amazon Rekognition, inicie seu processador de stream do Amazon Rekognition Video. Para obter mais informações, consulte Visão geral das operações do processador de stream do Amazon Rekognition Video.

Solução de problemas de streaming de vídeo

Este tópico fornece informações sobre solução de problemas para usar o Amazon Rekognition Video com streaming de vídeos.

Tópicos

- Não sei se meu processador de fluxo foi criado com êxito
- Eu não sei se configurei corretamente o meu processador de fluxos
- Meu processador de fluxos não está retornando resultados
- O estado do meu processador de fluxos é FAILED
- Meu processador de fluxos não está retornando os resultados esperados

Não sei se meu processador de fluxo foi criado com êxito

Use o AWS CLI comando a seguir para obter uma lista dos processadores de stream e seu status atual.

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

Você pode obter detalhes adicionais usando o AWS CLI comando a seguir. Substitua `stream-processor-name` pelo nome do processador de fluxos necessário.

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

Eu não sei se configurei corretamente o meu processador de fluxos

Se seu código não estiver gerando os resultados da análise do Amazon Rekognition Video, seu processador de stream pode não estar configurado corretamente. Faça o seguinte para confirmar se o processador de fluxos está configurado corretamente e pode produzir resultados.

Para determinar se sua solução está configurada corretamente

1. Execute o comando a seguir para confirmar se o processador de fluxos está no estado de execução. Altere `stream-processor-name` para o nome do processador de fluxos. O processador de fluxos estará em execução se o valor de `Status` for `RUNNING`. Se o status for `RUNNING` e você não estiver obtendo resultados, consulte Meu processador de fluxos não está

retornando resultados. Se o status for FAILED, consulte O estado do meu processador de fluxos é FAILED.

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

2. Se seu processador de stream estiver em execução, execute o seguinte PowerShell comando ou Bash para ler dados do stream de dados de saída do Kinesis.

   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. Use a ferramenta Decode no site Base64 Decode para decodificar a saída em uma string legível. Para obter mais informações, consulte Etapa 3: Obter o registro.
4. Se os comandos funcionarem e você vir os resultados da detecção de face no fluxo de dados do Kinesis, sua solução estará configurada corretamente. Se ocorrer uma falha no comando, verifique as outras sugestões de solução de problemas e consulte Conceder ao Amazon Rekognition Video acesso aos seus recursos.

Como alternativa, você pode usar o AWS Lambda blueprint kinesis-process-record "" para registrar mensagens do stream de dados do Kinesis CloudWatch para visualização contínua. Isso acarreta custos adicionais para AWS Lambda e. CloudWatch

Meu processador de fluxos não está retornando resultados

O processador de fluxos pode não retornar resultados por vários motivos.

Motivo 1: seu processador de fluxo não está configurado corretamente

Seu processador de fluxos pode não estar configurado corretamente. Para obter mais informações, consulte Eu não sei se configurei corretamente o meu processador de fluxos.

Motivo 2: Seu processador de fluxos não está no estado RUNNING

Para solucionar o status de um processador de fluxos

1. Verifique o status do processador de stream com o AWS CLI comando a seguir.

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

2. Se o valor de `Status` for `STOPPED`, inicie o processador de fluxos com o seguinte comando:

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

3. Se o valor de `Status` for `FAILED`, consulte O estado do meu processador de fluxos é FAILED.
4. Se o valor de `Status` for `STARTING`, aguarde 2 minutos e verifique o status repetindo a etapa 1. Se o valor de Status ainda for `STARTING`, faça o seguinte:

   a. Exclua o processador de fluxos com o seguinte comando.

   ```
   aws rekognition delete-stream-processor --name stream-processor-name
   ```

   b. Crie um novo processador de fluxos com a mesma configuração. Para obter mais informações, consulte Trabalhando com eventos de streaming de vídeo.

   c. Se você ainda estiver tendo problemas, entre em contato com o AWS Support.
5. Se o valor de `Status` for `RUNNING`, consulte Motivo 3: Não há dados ativos no stream de vídeo do Kinesis.

Motivo 3: Não há dados ativos no stream de vídeo do Kinesis

Para verificar se há dados ativos no stream de vídeo do Kinesis

1. Faça login no e abra AWS Management Console o console do Amazon Kinesis Video Streams https://console.aws.amazon.com/kinesisvideo/em.
2. Selecione o stream de vídeo Kinesis que é a entrada para o processador de stream Amazon Rekognition.
3. Se a pré-visualização indicar Sem dados no stream, então não há dados no stream de entrada para o Amazon Rekognition Video processar.

Para obter informações sobre a produção de vídeo com o Kinesis Video Streams, consulte Bibliotecas do Kinesis Video StreamsProducer.

O estado do meu processador de fluxos é FAILED

Você pode verificar o estado de um processador de stream usando o AWS CLI comando a seguir.

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

Se o valor de Status for FAILED, verifique as informações de solução de problemas para as seguintes mensagens de erro.

Erro: "Acesso negado à função"

O perfil do IAM usado pelo processador de stream não existe ou o Amazon Rekognition Video não tem permissão para assumir a função.

Para solucionar problemas de acesso ao perfil do IAM

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, selecione Roles (Funções) e confirme se a função existe.
3. Se a função existir, verifique se a função tem a política de AmazonRekognitionServiceRolepermissões.
4. Se a função não existir ou não tiver as permissões corretas, consulte Conceder ao Amazon Rekognition Video acesso aos seus recursos.
5. Inicie o processador de stream com o AWS CLI comando a seguir.

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

Erro: "Access denied to Kinesis Video (Acesso negado ao vídeo do Kinesis) ou Access denied to Kinesis Data (Acesso negado aos dados do Kinesis)"

A função não tem acesso às operações `GetMedia` e `GetDataEndpoint` da API do Kinesis Video Streams. Também pode não ter acesso às operações `PutRecord` e `PutRecords` da API Kinesis Data Streams

Para solucionar problemas de permissões da API

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Abra a função e verifique se ela tem a seguinte política de permissões associada.

```
{
    "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 qualquer uma das permissões estiver ausente, atualize a política. Para obter mais informações, consulte Conceder ao Amazon Rekognition Video acesso aos seus recursos.

Erro: “***input-video-stream-name***O stream não existe”

A entrada de stream de vídeo do Kinesis para o processador de stream não existe ou não está configurada corretamente.

Para solucionar problemas do stream de vídeo do Kinesis

1. Use o seguinte comando para confirmar se o stream existe.

```
aws kinesisvideo list-streams
```

2. Se o stream existir, verifique o seguinte.

   - O nome de recurso da Amazon (ARN) é igual ao ARN do stream de entrada para o processador de fluxos.
   - O stream de vídeo do Kinesis está na mesma região do processador de stream.

   Se o processador de stream não estiver configurado corretamente, exclua-o com o AWS CLI comando a seguir.

   ```
   aws rekognition delete-stream-processor --name stream-processor-name
   ```

3. Crie um novo processador de fluxos com o stream de vídeo desejado do Kinesis. Para obter mais informações, consulte Criação do processador de stream de pesquisa facial Amazon Rekognition Video.

Erro: "Coleção não encontrada"

A coleção Amazon Rekognition usada pelo processador de stream para combinar faces não existe ou a coleção errada está sendo usada.

Para confirmar a coleção

1. Use o AWS CLI comando a seguir para determinar se a coleção necessária existe. `region`Mude para a AWS região na qual você está executando seu processador de stream.

   ```
   aws rekognition list-collections --region region
   ```

   Se a coleção necessária não existir, crie uma nova coleção e adicione informações de faces. Para obter mais informações, consulte Pesquisa de faces em uma coleção.

2. Em sua chamada para CreateStreamProcessor, verifique se o valor do parâmetro de entrada `CollectionId` está correto.

3. Inicie o processador de stream com o AWS CLI comando a seguir.

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

Erro: “Transmissão ***output-kinesis-data-stream-name*** na conta ***account-id*** não encontrada”

O stream de dados de saída do Kinesis usado pelo processador de stream não existe na sua região Conta da AWS ou não está na mesma AWS região do seu processador de stream.

Para solucionar problemas no fluxo de dados do Kinesis

1. Use o AWS CLI comando a seguir para determinar se o stream de dados do Kinesis existe. `region`Mude para a AWS região na qual você está usando seu processador de stream.

   ```
   aws kinesis list-streams --region region
   ```

2. Se o fluxo de dados do Kinesis existir, verifique se o nome do stream é igual ao nome do stream de saída usado pelo processador de fluxos.
3. Se o stream de dados do Kinesis não existir, ele poderá existir em outra AWS região. O fluxo de dados do Kinesis deve estar na mesma região que o processador de fluxos.
4. Se necessário, crie um novo fluxo de dados do Kinesis.

   a. Crie um fluxo de dados do Kinesis com o mesmo nome que o usado pelo processador de fluxos. Para obter mais informações, consulte Etapa 1: Criar um fluxo de dados.

   b. Inicie o processador de stream com o AWS CLI comando a seguir.

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

Meu processador de fluxos não está retornando os resultados esperados

Se o processador de fluxos não estiver retornando as correspondências de face esperadas, use as informações a seguir.

- Pesquisa de faces em uma coleção
- Recomendações para a configuração da câmera (streaming de vídeo)

Pessoas trafegando

> **Note**
>
> Aviso de fim do suporte: em 31 de outubro de 2025, a AWS interromperá o suporte para Trajetória de pessoas do Amazon Rekognition. Após 31 de outubro de 2025, você não poderá mais usar o recurso Trajetória de pessoas do Rekognition. Veja esta postagem em blog para obter mais informações.

O Amazon Rekognition Video pode criar um rastreamento do caminho que as pessoas percorrem nos vídeos e fornecer informações como:

- A localização da pessoa no quadro do vídeo na hora em que o caminho foi rastreado.
- Pontos de referência faciais, como a posição do olho esquerdo, quando detectados.

O caminho das pessoas do Amazon Rekognition Video nos vídeos armazenados é uma operação assíncrona. Para iniciar a trajetória das pessoas em videochamadas StartPersonTracking. O Amazon Rekognition Video publica o status de conclusão da análise de vídeo em um tópico do Amazon Simple Notification Service. Se a análise do vídeo for bem-sucedida, ligue GetPersonTrackingpara obter os resultados da análise do vídeo. Para obter mais informações sobre como chamar as operações da API Amazon Rekognition Video, consulte Chamando as operações de vídeo do Amazon Rekognition Video.

O procedimento a seguir mostra como rastrear o caminho das pessoas por meio de um vídeo armazenado em um bucket do Amazon S3. O exemplo expande o código em Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK) no qual usa uma fila do Amazon Simple Queue Service para obter o status de conclusão de uma solicitação de análise de vídeo.

Para detectar pessoas em um vídeo armazenado em um bucket (SDK) do Amazon S3

1. Execute Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK).
2. Adicione o código a seguir à classe `VideoDetect` criada por você na etapa 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);

       }
```

Na função `main`, substitua as linhas:

```
       StartLabelDetection(amzn-s3-demo-bucket, video);

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

por:

```
        StartPersonDetection(amzn-s3-demo-bucket, video);

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

Java V2

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
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
```

Na função `main`, substitua as linhas:

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

por:

```
    analyzer.StartPersonPathing()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetPersonPathingResults()
```

CLI

Execute o AWS CLI comando a seguir para fazer com que as pessoas criem caminhos em um vídeo.

```
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
```

Atualize os seguintes valores:

- Mude `amzn-s3-demo-bucket` e `video-name` para o nome do bucket do Amazon S3 e o nome do arquivo que você especificou na etapa 2.
- Altere `region-name` para a região da AWS que você está usando.
- Substitua o valor de `profile-name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.
- Mude `topic-ARN` para o ARN do tópico do Amazon SNS que você criou na etapa 3 do Configuração do Amazon Rekognition Video.

- Mude `role-ARN` para o ARN do perfil de serviço do IAM que você criou na etapa 7 do Configuração do Amazon Rekognition Video.

Se você estiver acessando a CLI em um dispositivo Windows, use aspas duplas em vez de aspas simples e escape das aspas duplas internas com barra invertida (ou seja, \) para resolver quaisquer erros de analisador que você possa encontrar. Para ver um exemplo, veja abaixo:

```
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
```

Depois de executar o exemplo do código de procedimento, copie o `jobID` retornado e forneça-o ao seguinte comando `GetPersonTracking` abaixo para obter seus resultados, substitua `job-id-number` pelo `jobID` que você recebeu anteriormente:

```
aws rekognition get-person-tracking --job-id job-id-number
```

> **Note**
>
> Se você já tiver executado um exemplo de vídeo diferente de Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK), o código a ser substituído poderá ser diferente.

3. Execute o código. Os identificadores exclusivos para pessoas rastreadas são mostrados ao longo do tempo, em segundos, em que os caminhos das pessoas foram rastreadas.

GetPersonTracking resposta da operação

`GetPersonTracking` retorna uma matriz, `Persons`, de objetos PersonDetection que contêm detalhes sobre essas pessoas detectadas em vídeo e quando seus caminhos foram rastreados.

Você pode classificar `Persons` usando o parâmetro de entrada `SortBy`. Especifique `TIMESTAMP` para classificar os elementos pela hora em que os caminhos das pessoas são rastreados no vídeo. Especifique `INDEX` para classificar por pessoas rastreadas no vídeo. Em cada conjunto de resultados de uma pessoa, os elementos são classificados por confiança decrescente da precisão de rastreamento de caminho. Por padrão, `Persons` é retornado classificado por `TIMESTAMP`.
O exemplo a seguir é a resposta do JSON em `GetPersonDetection`. Os resultados são classificados pelo tempo, em milissegundos desde o início do vídeo, em que os caminhos das pessoas foram rastreadas no vídeo. Na resposta, observe o seguinte:

- Informações da pessoa: o elemento da matriz `PersonDetection` contém informações sobre a pessoa detectada. Por exemplo, a hora em que a pessoa foi detectada (`Timestamp`), a posição da pessoa no quadro do vídeo no momento em que ela foi detectada (`BoundingBox`) e a confiança do Amazon Rekognition Video de que a pessoa foi detectada corretamente (`Confidence`).

  Os traços faciais não são retornados em cada data e hora nos quais o caminho da pessoa é rastreado. Além disso, em algumas circunstâncias, o corpo de uma pessoa rastreada pode não estar visível, em cujo caso apenas o local de sua face é retornado.
- Informações de paginação: o exemplo mostra uma página de informações de detecção de pessoas. Você pode especificar quantos elementos de pessoas retornar no parâmetro de entrada `MaxResults` para `GetPersonTracking`. Se existirem mais resultados além de `MaxResults`, o `GetPersonTracking` retornará um token (`NextToken`) usado para obter a próxima página de resultados. Para obter mais informações, consulte Obter os resultados da análise do Amazon Rekognition Video.
- Índice — Um identificador exclusivo para identificar a pessoa em todo o vídeo.
- Informações de vídeo – a resposta inclui informações sobre o formato do vídeo (`VideoMetadata`) em cada página de informações retornada pelo `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
    }
}
```

Detecção de equipamento de proteção individual

O Amazon Rekognition pode detectar o Equipamento de Proteção Individual (EPI) usado por pessoas em uma imagem. Você pode usar essas informações para melhorar as práticas de segurança no local de trabalho. Por exemplo, você pode usar a detecção de EPI para ajudar a determinar se os trabalhadores em um canteiro de obras estão usando coberturas para a cabeça ou se os profissionais da área médica estão usando coberturas faciais e protetores para as mãos. A imagem a seguir mostra alguns dos tipos de EPI que podem ser detectados.



Para detectar PPE em uma imagem, você chama a DetectProtectiveEquipmentAPI e passa uma imagem de entrada. A resposta é uma estrutura JSON que inclui o seguinte.

- As pessoas detectadas na imagem.
- As partes do corpo em que o EPI é usado (rosto, cabeça, mão esquerda e mão direita).

- Os tipos de EPI detectados em partes do corpo (cobertura facial, cobertura para as mãos e cobertura para a cabeça).
- Para itens de EPI detectados, um indicador de se o EPI cobre ou não a parte corporal correspondente.

As caixas delimitadoras são devolvidas para a localização das pessoas e itens de EPI detectados na imagem.

Opcionalmente, você pode solicitar um resumo dos itens de EPI e das pessoas detectadas em uma imagem. Para obter mais informações, consulte Resumindo o EPI detectado em uma imagem.

> **Note**
>
> A detecção de PPE do Amazon Rekognition não realiza reconhecimento facial ou comparação facial e não consegue identificar as pessoas detectadas.

Tipos de EPI

DetectProtectiveEquipmentdetecta os seguintes tipos de EPI. Se você quiser detectar outros tipos de EPI em imagens, considere usar Amazon Rekognition Custom Labels para treinar um modelo personalizado. Para obter mais informações, consulte Amazon Rekognition Custom Labels.

Proteção facial

`DetectProtectiveEquipment` pode detectar proteções faciais comuns, como máscaras cirúrgicas, N95 e feitas de tecido.

Proteções de mão

`DetectProtectiveEquipment` pode detectar proteções de mão, como luvas cirúrgicas e luvas de segurança.

Proteções de cabeça

`DetectProtectiveEquipment` pode detectar capacetes e chapéus.

A API indica que uma cobertura de cabeça, mão ou rosto foi detectada em uma imagem. A API não retorna informações sobre o tipo de uma capa específica. Por exemplo, "luva cirúrgica" para o tipo de capa de mão.

Confiança na detecção de EP

O Amazon Rekognition faz uma previsão sobre a presença de EPI, pessoas e partes do corpo em uma imagem. A API fornece uma pontuação (50 a 100) que indica a confiança do Amazon Rekognition na precisão de uma previsão.

> **Note**
>
> Se você planeja usar a operação `DetectProtectiveEquipment` para tomar uma decisão que afete os direitos, a privacidade ou o acesso a serviços de um indivíduo, recomendamos que você passe o resultado a um humano para análise e validação antes de agir.

Resumindo o EPI detectado em uma imagem

Opcionalmente, você pode solicitar um resumo dos itens do PPE e das pessoas detectadas em uma imagem. Você pode especificar uma lista de equipamentos de proteção necessários (proteção facial, proteção para as mãos ou proteção para a cabeça) e um limite mínimo de confiança (por exemplo, 80%). A resposta inclui um resumo consolidado do identificador por imagem (ID) de pessoas com o EPI exigido, pessoas sem o EPI exigido e pessoas em que uma determinação não pôde ser feita.

O resumo permite que você responda rapidamente a perguntas como Quantas pessoas não estão usando proteções faciais? ou Todo mundo está usando EPI? Cada pessoa detectada no resumo tem um ID exclusivo. Você pode usar o documento de identidade para descobrir informações como a localização da caixa delimitadora de uma pessoa que não usa EPI.

> **Note**
>
> O ID é gerado aleatoriamente com base na análise por imagem e não é consistente em todas as imagens ou em várias análises da mesma imagem.

Você pode resumir capas faciais, capas de cabeça, capas para mãos ou uma combinação de sua escolha. Para especificar os tipos de EPI necessários, consulte Especificando os requisitos

do resumo. Você também pode especificar um nível mínimo de confiança (50-100) que deve ser atingido para que as detecções sejam incluídas no resumo.

Para obter mais informações sobre a resposta resumida de `DetectProtectiveEquipment`, consulte Entendendo a DetectProtectiveEquipment resposta.

Tutorial: Criando uma AWS Lambda função que detecta imagens com PPE

Você pode criar uma AWS Lambda função que detecte equipamentos de proteção individual (EPI) em imagens localizadas em um bucket do Amazon S3. Consulte o GitHub repositório de exemplos do SDK de AWS documentação para ver este tutorial do Java V2.

Entendendo a API de detecção de equipamentos de proteção individual

As informações a seguir descrevem a DetectProtectiveEquipmentAPI. Para obter um código de exemplo, consulte Detectando equipamento de proteção individual em uma imagem.

Fornecer uma imagem

Você pode fornecer a imagem de entrada (formato JPG ou PNG) como bytes de imagem ou fazer referência a uma imagem armazenada em um bucket do Amazon S3.

Recomendamos usar imagens em que a face da pessoa esteja voltado para a câmera.

Se sua imagem de entrada não estiver girada para a orientação de 0 grau, recomendamos girá-la para a orientação de 0 grau antes de enviá-la para `DetectProtectiveEquipment`. As imagens no formato JPG podem conter informações de orientação nos metadados do formato de arquivo de imagem intercambiável (Exif). Você pode usar essas informações para escrever um código que gira sua imagem. Para obter mais informações, consulte Exif versão 2.32. As imagens no formato PNG não contêm informações de orientação da imagem.

Para passar uma imagem de um bucket do Amazon S3, use um usuário com pelo menos privilégios do ReadOnlyAccess AmazonS3. Use um usuário com privilégios `AmazonRekognitionFullAccess` para fazer chamadas `DetectProtectiveEquipment`.

No exemplo de JSON de entrada a seguir, a imagem é passada em um bucket do Amazon S3. Para obter mais informações, consulte Como trabalhar com imagens. O exemplo solicita um resumo de

todos os tipos de EPI (cobertura de cabeça, capa de mão e cobertura facial) com uma confiança de detecção mínima (`MinConfidence`) de 80%. Você deve especificar um valor de `MinConfidence` entre 50 e 100%, pois `DetectProtectiveEquipment` retorna previsões somente quando a confiança da detecção está entre 50% e 100%. Se você especificar um valor menor que 50%, os resultados serão os mesmos especificando um valor de 50%. Para obter mais informações, consulte Especificando os requisitos do resumo.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "worker.jpg"
        }
    },
    "SummarizationAttributes": {
        "MinConfidence": 80,
        "RequiredEquipmentTypes": [
            "FACE_COVER",
            "HAND_COVER",
            "HEAD_COVER"
        ]
    }
}
```

Se você tiver uma grande coleção de imagens a serem processadas para reconhecimento, considere usar o AWS Batch para processar as chamadas de `DetectProtectiveEquipment` em lotes em segundo plano.

Especificando os requisitos do resumo

Opcionalmente, você pode usar o parâmetro de entrada `SummarizationAttributes` (ProtectiveEquipmentSummarizationAttributes) para solicitar informações resumidas sobre os tipos de PPE detectados em uma imagem.

Para especificar os tipos de PPE a serem resumidos, use o campo de matriz `RequiredEquipmentTypes`. Na matriz, inclua um ou mais de `FACE_COVER`, `HAND_COVER` ou `HEAD_COVER`.

Use o campo `MinConfidence` para especificar uma confiança mínima de detecção (50-100). O resumo não inclui pessoas, partes do corpo, cobertura de partes do corpo e itens de EPI, detectados com uma confiança inferiora `MinConfidence`.

Para obter informações sobre a resposta resumida de `DetectProtectiveEquipment`, consulte Entendendo a DetectProtectiveEquipment resposta.

Entendendo a DetectProtectiveEquipment resposta

`DetectProtectiveEquipment` retorna uma matriz de pessoas detectadas na imagem de entrada. Para cada pessoa, as informações sobre partes do corpo detectadas e itens de EPI detectados são devolvidas. O JSON da imagem a seguir de um trabalhador usando uma cobertura para a cabeça, para as mãos e para a face é o seguinte.



No JSON, observe o seguinte.

- Pessoas detectadas — `Persons` é uma variedade de pessoas detectadas na imagem (incluindo pessoas que não usam EPI). `DetectProtectiveEquipment` pode detectar EPI em até 15 pessoas detectadas em uma imagem. Cada ProtectiveEquipmentPersonobjeto na matriz contém uma identificação pessoal, uma caixa delimitadora para a pessoa, partes do corpo detectadas e

itens detectados de EPI. O valor de `Confidence` in `ProtectiveEquipmentPerson` indica a porcentagem de confiança que o Amazon Rekognition tem de que a caixa delimitadora contém uma pessoa.

- Partes do corpo — `BodyParts` é uma série de partes do corpo (ProtectiveEquipmentBodyPart) detectadas em uma pessoa (incluindo partes do corpo não cobertas pelo EPI). Cada `ProtectiveEquipmentBodyPart` inclui o nome (`Name`) da parte do corpo detectada. `DetectProtectEquipment` pode detectar partes da face, cabeça, mão esquerda e direita do corpo. O campo `Confidence` em `ProtectiveEquipmentBodyPart` indica a porcentagem de confiança que o Amazon Rekognition tem na precisão de detecção da parte do corpo.
- Itens PPE: a matriz `EquipmentDetections` em um objeto `ProtectiveEquipmentBodyPart` contém uma matriz de itens EPI detectados. Cada EquipmentDetectionobjeto contém os seguintes campos.
  - `Type`: o tipo do EPI detectado.
  - `BoundingBox`: uma caixa delimitadora ao redor do EPI detectado.
  - `Confidence`: a confiança que o Amazon Rekognition tem de que a caixa delimitadora contém o EPI detectado.
  - `CoversBodyPart`: indica se o EPI detectado está na parte do corpo correspondente.

  O CoversBodyPartcampo `Value` é um valor booleano que indica se o PPE detectado está na parte correspondente do corpo. O campo `Confidence` indica a confiança na previsão. Você pode usar `CoversBodyPart` para filtrar os casos em que o EPI detectado está na imagem, mas não na verdade na pessoa.

  > **Note**
  >
  > `CoversBodyPart` não indica, nem implica, que a pessoa está adequadamente protegida pelo equipamento de proteção ou que o equipamento de proteção em si está usado adequadamente.

- Informações resumidas — `Summary` contém as informações resumidas especificadas no parâmetro de entrada `SummarizationAttributes`. Para obter mais informações, consulte Especificando os requisitos do resumo.

  `Summary`é um objeto do tipo ProtectiveEquipmentSummaryque contém as seguintes informações.

  - `PersonsWithRequiredEquipment`— Uma série IDs de pessoas em que cada pessoa atende aos seguintes critérios.

  - A pessoa está usando todos os EPI especificados no parâmetro de entrada `SummarizationAttributes`.
  - O `Confidence` para a pessoa (`ProtectiveEquipmentPerson`), parte do corpo (`ProtectiveEquipmentBodyPart`), equipamento de proteção (`EquipmentDetection`) é igual ou maior que o limite mínimo de confiança especificado (`MinConfidence`).
  - O valor de `CoversBodyPart` para todos os itens do EPI é verdadeiro.
- `PersonsWithoutRequiredEquipment`— Uma série IDs de pessoas que atendem a um dos seguintes critérios.
  - O valor `Confidence` da pessoa (`ProtectiveEquipmentPerson`), parte do corpo (`ProtectiveEquipmentBodyPart`) e cobertura da parte do corpo (`CoversBodyPart`) é maior do que o limite mínimo de confiança especificado (`MinConfidence`), mas falta um ou mais EPI especificados (`SummarizationAttributes`) à pessoa.
  - O valor de `CoversBodyPart` é falso para qualquer EPI (`SummarizationAttributes`) especificado que tenha um valor `Confidence` maior que o limite mínimo de confiança especificado (`MinConfidence`). A pessoa também tem todos os EPI especificados (`SummarizationAttributes`) e os `Confidence` valores para pessoa (`ProtectiveEquipmentPerson`), parte do corpo (`ProtectiveEquipmentBodyPart`) e equipamento de proteção (`EquipmentDetection`) são maiores ou iguais ao limite mínimo de confiança (`MinConfidence`.
- `PersonsIndeterminate`— Uma matriz IDs de pessoas detectadas em que o `Confidence` valor da pessoa (`ProtectiveEquipmentPerson`), parte do corpo (`ProtectiveEquipmentBodyPart`), equipamento de proteção (`EquipmentDetection`) ou `CoversBodyPart` booleano é inferior ao limite mínimo de confiança especificado (`MinConfidence`).

Use o tamanho da matriz para obter uma contagem de um resumo específico. Por exemplo, o tamanho de `PersonsWithRequiredEquipment` indica o número de pessoas detectadas usando o tipo especificado de EPI.

Você pode usar o ID pessoal para descobrir mais informações sobre uma pessoa, como a localização da caixa delimitadora da pessoa. O ID da pessoa é mapeado para o campo ID de um objeto `ProtectiveEquipmentPerson`) retornado em `Persons` (matriz de `ProtectiveEquipmentPerson`). Você pode então obter a caixa delimitadora e outras informações do objeto correspondente `ProtectiveEquipmentPerson`.

```
{
   "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": []
    }
}
```

Detectando equipamento de proteção individual em uma imagem

Para detectar equipamentos de proteção individual (EPI) em pessoas em uma imagem, use a operação DetectProtectiveEquipmentde API sem armazenamento.

Você pode fornecer a imagem de entrada como uma matriz de bytes de imagem (bytes de imagem codificados em base64) ou como um objeto do Amazon S3, usando o AWS SDK ou o AWS Command Line Interface (AWS CLI). Esses exemplos usam uma imagem armazenada em um bucket do Amazon S3. Para obter mais informações, consulte Como trabalhar com imagens.

Para detectar EPI em pessoas em uma imagem

1. Se ainda não tiver feito isso:

    a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

    b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Faça upload de uma imagem (que contenha uma ou mais pessoas usando EPI) no seu bucket do S3.

   Para obter instruções, consulte Como fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

3. Use os exemplos a seguir para chamar a operação `DetectProtectiveEquipment`. Para obter informações sobre como exibir caixas delimitadoras em uma imagem, consulte Exibir caixas delimitadoras.

   Java

   Este exemplo exibe informações sobre os itens de EPI detectados em pessoas detectadas em uma imagem.

   Altere o valor de `amzn-s3-demo-bucket` para o nome do bucket do Amazon S3 que contém sua imagem. Altere o valor de `photo` para o nome do seu arquivo de imagem.

```
//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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
//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

Esse AWS CLI comando solicita um resumo do PPE e exibe a saída JSON para a operação da `detect-protective-equipment` CLI.

Mude `amzn-s3-demo-bucket` para o nome de um bucket do Amazon S3 que contém uma imagem. Mude `input.jpg` para o nome da imagem que você deseja usar.

```
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']"
```

Esse AWS CLI comando exibe a saída JSON para a operação da `detect-protective-equipment` CLI.

Mude `amzn-s3-demo-bucket` para o nome de um bucket do Amazon S3 que contém uma imagem. Mude `input.jpg` para o nome da imagem que você deseja usar.

```
aws rekognition detect-protective-equipment \
```

```
  --image "S3Object={Bucket=amzn-s3-demo-bucket,Name=input.jpg}"
```

Python

Este exemplo exibe informações sobre os itens de EPI detectados em pessoas detectadas em uma imagem.

Altere o valor de `amzn-s3-demo-bucket` para o nome do bucket do Amazon S3 que contém sua imagem. Altere o valor de `photo` para o nome do seu arquivo de imagem. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
# 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()
```

Exemplo: Desenhando caixas delimitadoras em torno de capas faciais

Os exemplos a seguir mostram como desenhar caixas delimitadoras em torno de coberturas faciais detectadas em pessoas. Para ver um exemplo que usa AWS Lambda o Amazon DynamoDB, consulte o repositório de exemplos de SDK AWS de documentação. GitHub

Para detectar capas faciais, você usa a operação DetectProtectiveEquipmentde API sem armazenamento. A imagem é carregada do sistema de arquivos local. Você fornece a imagem de entrada `DetectProtectiveEquipment` como uma matriz de bytes de imagem (bytes de imagem codificados em base64). Para obter mais informações, consulte Como trabalhar com imagens.

O exemplo exibe uma caixa delimitadora ao redor das capas faciais detectadas. A caixa delimitadora é verde se a cobertura facial cobrir totalmente a parte do corpo. Caso contrário, uma caixa delimitadora vermelha será exibida. Como aviso, uma caixa delimitadora amarela é exibida dentro da caixa delimitadora da tampa frontal, se a confiança de detecção for menor que o valor de confiança especificado. Se uma cobertura facial não for detectada, uma caixa delimitadora vermelha é desenhada ao redor da pessoa.

A saída da imagem é semelhante à seguinte.



Para exibir caixas delimitadoras nas capas faciais detectadas

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use os exemplos a seguir para chamar a operação `DetectProtectiveEquipment`. Para obter informações sobre como exibir caixas delimitadoras em uma imagem, consulte Exibir caixas delimitadoras.

   Java

   Na função `main`, altere o seguinte:

- O valor de `photo` para o caminho e o nome de arquivo de um arquivo de imagem local (PNG ou JPEG).
- O valor de `confidence` até o nível de confiança desejado (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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
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

Na função `main`, altere o seguinte:

- O valor de `photo` para o caminho e o nome de arquivo de um arquivo de imagem local (PNG ou JPEG).
- O valor de `confidence` até o nível de confiança desejado (50-100).
- Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
#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

No exemplo de CLI a seguir, altere o valor dos argumentos listados abaixo:

- O valor de `photo` para o caminho e o nome de arquivo de um arquivo de imagem local (PNG ou JPEG).
- O valor de `confidence` até o nível de confiança desejado (50-100).
- Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
                                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 você estiver acessando a CLI em um dispositivo Windows, use aspas duplas em vez de aspas simples e escape das aspas duplas internas com barra invertida (ou seja, \) para resolver quaisquer erros de analisador que você possa encontrar. Para obter um exemplo, veja o seguinte:

```
                                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\"]}"
```

Reconhecendo celebridades

O Amazon Rekognition facilita que os clientes reconheçam automaticamente dezenas de milhares de personalidades conhecidas em imagens e vídeos usando machine learning (ML). Os metadados fornecidos pela API de reconhecimento de celebridades reduzem significativamente o esforço manual repetitivo necessário para marcar o conteúdo e torná-lo facilmente pesquisável.

A rápida proliferação de conteúdo de imagem e vídeo significa que as empresas de mídia geralmente têm dificuldade em organizar, pesquisar e utilizar seus catálogos de mídia em grande escala. Os canais de notícias e as emissoras esportivas geralmente precisam encontrar imagens e vídeos rapidamente para responder aos eventos atuais e criar uma programação relevante. Metadados insuficientes dificultam essas tarefas, mas com o Amazon Rekognition você pode marcar automaticamente grandes volumes de conteúdo novo ou arquivado para facilitar a busca por um conjunto abrangente de celebridades internacionais amplamente conhecidas, como atores, esportistas e criadores de conteúdo on-line.

O reconhecimento de celebridades do Amazon Rekognition foi projetado para ser usado exclusivamente nos casos em que você espera que haja uma celebridade conhecida em uma imagem ou vídeo. Para obter informações sobre como reconhecer faces que não são de celebridades, consulte Pesquisa de faces em uma coleção.

> **Note**
>
> Se você é uma celebridade e não quer ser incluído nesse recurso, entre em contato com o AWS Support ou envie um e-mail para `<rekognition-celebrity-opt-out@amazon .com>`.

Tópicos

- Reconhecimento de celebridades comparado à busca facial
- Reconhecer celebridades em uma imagem
- Reconhecendo celebridades em um vídeo armazenado
- Obter informações sobre uma celebridade

Reconhecimento de celebridades comparado à busca facial

O Amazon Rekognition oferece a funcionalidade de reconhecimento de celebridades e reconhecimento facial. Essas funcionalidades têm algumas diferenças importantes em seus casos de uso e nas melhores práticas.

O reconhecimento de celebridades vem pré-treinado com a capacidade de reconhecer centenas de milhares de pessoas famosas em campos como esportes, mídia, política e negócios. Essa funcionalidade foi desenvolvida para ajudar você a pesquisar grandes volumes de imagens ou vídeos, a fim de identificar um pequeno conjunto que pode conter uma determinada celebridade. Não se destina a ser usado para combinar faces de pessoas diferentes que não sejam celebridades. Em situações em que a precisão da correspondência com a celebridade é importante, recomendamos também o uso de operadores humanos que possam examinar um número menor de conteúdos marcados para ajudar a garantir um alto nível de precisão e a aplicação apropriada de avaliação humana. O reconhecimento de celebridades não deve ser usado de uma maneira que possa resultar em um impacto negativo nas liberdades civis.

Por outro lado, o reconhecimento facial é uma funcionalidade mais geral que permite criar suas próprias coleções de faces com seus próprios vetores faciais para verificar identidades ou pesquisar qualquer pessoa, não apenas celebridades. O reconhecimento facial pode ser usado em aplicativos como autenticação de acesso a edifícios, segurança pública e mídias sociais. Em todos esses casos, é recomendável que você utilize as melhores práticas, os limiares de confiança apropriados (incluindo 99% para casos de uso de segurança pública) e revisão humana, em situações em que a precisão da correspondência é importante.

Para obter mais informações, consulte Pesquisa de faces em uma coleção.

Reconhecer celebridades em uma imagem

Para reconhecer celebridades em imagens e obter informações adicionais sobre celebridades reconhecidas, use a operação da API de não armazenamento RecognizeCelebrities. Por exemplo, nas mídias sociais ou nos setores de notícias e entretenimento, onde a coleta de informações pode ser urgente, você pode usar a operação `RecognizeCelebrities` para identificar até 64 celebridades em uma imagem e retornar links para páginas da web de celebridades, se estiverem disponíveis. O Amazon Rekognition não lembra em qual imagem detectou uma celebridade. O aplicativo deve armazenar essas informações.

Se você não tiver armazenado as informações adicionais de uma celebridade retornadas pelo `RecognizeCelebrities` e quiser evitar analisar uma imagem novamente para obtê-las, use GetCelebrityInfo. Para ligar `GetCelebrityInfo`, você precisa do identificador exclusivo que o Amazon Rekognition atribui a cada celebridade. O identificador é retornado como parte da resposta de `RecognizeCelebrities` para cada celebridade reconhecida em uma imagem.

Se você tiver uma grande coleção de imagens a serem processadas para reconhecimento de celebridades, considere usar o AWS Batch para processar as chamadas de `RecognizeCelebrities` em lotes em segundo plano. Ao adicionar uma nova imagem à sua coleção, você pode usar uma AWS Lambda função para reconhecer celebridades ligando `RecognizeCelebrities` enquanto a imagem é carregada em um bucket do S3.

Chamando RecognizeCelebrities

Você pode fornecer a imagem de entrada como uma matriz de bytes de imagem (bytes de imagem codificados em base64) ou como um objeto do Amazon S3, usando o (AWS Command Line Interface)AWS CLI ou o AWS SDK. No AWS CLI procedimento, você carrega uma imagem no formato.jpg ou .png em um bucket do S3. Nos procedimentos do AWS SDK, você usa uma imagem que é carregada do seu sistema de arquivos local. Para obter informações sobre recomendações de imagens de entrada, consulte Como trabalhar com imagens.

Para executar esse procedimento, você precisa de um arquivo de imagem que contenha uma ou mais faces de celebridades.

Como reconhecer celebridades em uma imagem

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use os exemplos a seguir para chamar a operação `RecognizeCelebrities`.

   Java

   Este exemplo exibe informações sobre as celebridades detectadas em uma imagem.

Altere o valor de photo para o caminho e o nome de arquivo de um arquivo de imagem que contenha uma ou mais faces de celebridades.

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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
//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

Esse AWS CLI comando exibe a saída JSON para a operação da `recognize-celebrities` CLI.

Mude `amzn-s3-demo-bucket` para o nome de um bucket do Amazon S3 que contém uma imagem. Altere `input.jpg` para o nome de arquivo de uma imagem que contenha uma ou mais faces de celebridades.

Substitua o valor de `profile_name` com o nome do seu perfil de desenvolvedor.

```
aws rekognition recognize-celebrities \
  --image "S3Object={Bucket=amzn-s3-demo-bucket,Name=input.jpg}"
```

Se você estiver acessando a CLI em um dispositivo Windows, use aspas duplas em vez de aspas simples e escape das aspas duplas internas com barra invertida (ou seja, \) para resolver quaisquer erros de analisador que você possa encontrar. Para obter um exemplo, veja o seguinte:

```
                                    aws rekognition recognize-celebrities --
image \
                                    "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --profile profile-name
```

Python

Este exemplo exibe informações sobre as celebridades detectadas em uma imagem.

Altere o valor de `photo` para o caminho e o nome de arquivo de um arquivo de imagem que contenha uma ou mais faces de celebridades.

Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Este exemplo exibe informações sobre as celebridades detectadas em uma imagem.

Altere o valor de photo para o caminho e o nome de arquivo de um arquivo de imagem que contenha uma ou mais faces de celebridades. Altere o valor de amzn-s3-demo-bucket para o nome do bucket S3 que contém o arquivo de imagem fornecido. Altere o valor de REGION para o nome da região associada ao seu usuário. Substitua o valor de profile_name na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
// 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

Este exemplo exibe informações sobre as celebridades detectadas em uma imagem.

Altere o valor de photo para o caminho e o nome de arquivo de um arquivo de imagem que contenha uma ou mais faces de celebridades (formato .jpg ou .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. Registre o valor de uma das celebridades IDs exibidas. Você precisará dele em Obter informações sobre uma celebridade.

RecognizeCelebrities solicitação de operação

A entrada de `RecognizeCelebrities` é uma imagem. Neste exemplo, a imagem é passada como bytes de imagem. Para obter mais informações, consulte Como trabalhar com imagens.

```
{
    "Image": {
        "Bytes": "/AoSiyvFpm....."
    }
}
```

RecognizeCelebrities resposta da operação

Estas são a entrada e a saída JSON de exemplo `RecognizeCelebrities`.

O `RecognizeCelebrities` retorna um conjunto de celebridades reconhecidas e um conjunto de faces não reconhecidas. No exemplo, observe:

- Celebridades reconhecidas — `Celebrities` é uma variedade de celebridades reconhecidas. Cada objeto Celebrity na matriz contém o nome da celebridade e uma lista de URLs referências para conteúdo relacionado, por exemplo, o link IMDB ou Wikidata da celebridade. O Amazon Rekognition ComparedFaceretorna um objeto que seu aplicativo pode usar para determinar onde o rosto da celebridade está na imagem e um identificador exclusivo para a celebridade. Use o identificador exclusivo para recuperar informações sobre a celebridade posteriormente com a operação da API GetCelebrityInfo.
- Faces não reconhecidas — `UnrecognizedFaces` é uma variedade de faces que não combinam com nenhuma celebridade conhecida. Cada objeto ComparedFace na matriz contém uma caixa delimitadora (bem como outras informações) que você pode usar para localizar a face na imagem.

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

Reconhecendo celebridades em um vídeo armazenado

O reconhecimento de celebridades do Amazon Rekognition Video em vídeos armazenados é uma operação assíncrona. Para reconhecer celebridades em um vídeo armazenado, use StartCelebrityRecognitionpara iniciar a análise de vídeo. O Amazon Rekognition Video publica o status de conclusão da análise de vídeo em um tópico do Amazon Simple Notification Service. Se a análise do vídeo for bem-sucedida, chame GetCelebrityRecognition para obter os resultados da análise. Para obter mais informações sobre como iniciar uma análise de vídeo e obter os resultados, consulte Chamando as operações de vídeo do Amazon Rekognition Video.

Esse procedimento expande o código em Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK), que usa uma fila do Amazon SQS para obter o status de conclusão de uma solicitação de análise de vídeo. Para executar esse procedimento, você precisa de um arquivo de vídeo que contenha uma ou mais faces de celebridades.

Para detectar celebridades em um vídeo armazenado em um bucket (SDK) do Amazon S3

1. Execute Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK).
2. Adicione o código a seguir à classe `VideoDetect` criada por você na etapa 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);

       }
```

Na função `main`, substitua a linha:

```
       StartLabelDetection(amzn-s3-demo-bucket, video);

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

por:

```
        StartCelebrityDetection(amzn-s3-demo-bucket, video);

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

Java V2

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
//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
```

Na função `main`, substitua as linhas:

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

por:

```
    analyzer.StartCelebrityDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetCelebrityDetectionResults()
```

Node.JS

No exemplo de código Node.Js a seguir, substitua o valor de `amzn-s3-demo-bucket` pelo nome do bucket do S3 que contém seu vídeo e o valor de `videoName` pelo nome do arquivo de vídeo. Você também precisará substituir o valor de `roleArn` pelo Arn associado à suo perfil de serviço do IAM. Por fim, substitua o valor de `region` pelo nome da região operacional associada à sua conta. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Execute o AWS CLI comando a seguir para começar a detectar celebridades em um vídeo.

```
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
```

Atualize os seguintes valores:

- Mude `amzn-s3-demo-bucket` e `video-name` para o nome do bucket do Amazon S3 e o nome do arquivo que você especificou na etapa 2.

- Altere `region-name` para a região da AWS que você está usando.
- Substitua o valor de `profile-name` com o nome do seu perfil de desenvolvedor.
- Mude `topic-ARN` para o ARN do tópico do Amazon SNS que você criou na etapa 3 do Configuração do Amazon Rekognition Video.
- Mude `role-ARN` para o ARN do perfil de serviço do IAM que você criou na etapa 7 do Configuração do Amazon Rekognition Video.

Se você estiver acessando a CLI em um dispositivo Windows, use aspas duplas em vez de aspas simples e escape das aspas duplas internas com barra invertida (ou seja, \) para resolver quaisquer erros de analisador que você possa encontrar. Para ver um exemplo, veja abaixo:

```
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
```

Depois de executar o exemplo do código de procedimento, copie o `jobID` retornado e forneça-o ao seguinte comando `GetCelebrityRecognition` abaixo para obter seus resultados, substitua `job-id-number` pelo `jobID` que você recebeu anteriormente:

```
aws rekognition get-celebrity-recognition --job-id job-id-number --profile
 profile-name
```

> **Note**
>
> Se você já tiver executado um exemplo de vídeo diferente de Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK), o código a ser substituído poderá ser diferente.

3. Execute o código. As informações sobre as celebridades reconhecidas no vídeo são mostradas.

GetCelebrityRecognition resposta da operação

O seguinte é um exemplo de resposta do JSON. A resposta inclui o seguinte:

- Celebridades reconhecidas — `Celebrities` é uma variedade de celebridades e as vezes em que elas são reconhecidas em um vídeo. Existe um objeto CelebrityRecognition para cada vez em que a celebridade é reconhecida no vídeo. Cada `CelebrityRecognition` contém informações sobre uma celebridade reconhecida (CelebrityDetail) e a hora (`Timestamp`) em que a celebridade foi reconhecida no vídeo. `Timestamp` é medida em milissegundos desde o início do vídeo.
- CelebrityDetail— Contém informações sobre uma celebridade reconhecida. Inclui o nome da celebridade (`Name`), o identificador (`ID`), o gênero conhecido da celebridade (`KnownGender`) e uma lista de referências para URLs conteúdo relacionado (`Urls`). Também inclui o nível de confiança que o Amazon Rekognition Video tem na precisão do reconhecimento e detalhes sobre o rosto da celebridade. FaceDetail Se precisar obter o conteúdo relacionado mais tarde, você poderá usar `ID` com getCelebrityInfo.
- VideoMetadata— Informações sobre o vídeo que foi analisado.


```
{
    "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
    }
}
```

Obter informações sobre uma celebridade

Nestes procedimentos, você obtém informações sobre celebridades usando a operação da API getCelebrityInfo. A celebridade é identificada usando o ID da celebridade retornado em uma chamada anterior a RecognizeCelebrities.

Chamando GetCelebrityInfo

Esses procedimentos exigem o ID de celebridade de uma celebridade que o Amazon Rekognition conhece. Use o ID da celebridade anotado em Reconhecer celebridades em uma imagem.

Para obter informações sobre uma celebridade (SDK)

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI AWS SDKs e. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use os exemplos a seguir para chamar a operação `GetCelebrityInfo`.

   Java

   Este exemplo exibe o nome e as informações sobre uma celebridade.

   `id`Substitua por uma das celebridades IDs exibidas emReconhecer celebridades em uma imagem.

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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
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

Esse AWS CLI comando exibe a saída JSON para a operação da `get-celebrity-info` CLI. `ID`Substitua por uma das celebridades IDs exibidas emReconhecer celebridades em uma imagem. Substitua o valor de `profile-name` com o nome do seu perfil de desenvolvedor.

```
aws rekognition get-celebrity-info --id celebrity-id --profile profile-name
```

Python

Este exemplo exibe o nome e as informações sobre uma celebridade.

`id`Substitua por uma das celebridades IDs exibidas emReconhecer celebridades em uma imagem. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Este exemplo exibe o nome e as informações sobre uma celebridade.

`id`Substitua por uma das celebridades IDs exibidas em[Reconhecer celebridades em uma imagem](#).

```
//Copyright 2018 Amazon.com, Inc. or its 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 solicitação de operação

Estas são a entrada e a saída JSON de exemplo `GetCelebrityInfo`.

A entrada para `GetCelebrityInfo` é o ID da celebridade obrigatório.

```
{
    "Id": "nnnnnnn"
}
```

GetCelebrityInfo resposta da operação

`GetCelebrityInfo` retorna uma matriz (`Urls`) de links para informações sobre a celebridade solicitada.

```
{
    "Name": "Celebrity Name",
    "Urls": [
        "www.imdb.com/name/nmnnnnnnn"
    ]
}
```

Como moderar um conteúdo

Você pode usar o Amazon Rekognition para detectar conteúdo impróprio, indesejado ou ofensivo. Você pode usar a APIs moderação do Rekognition em situações de mídia social, mídia de transmissão, publicidade e comércio eletrônico para criar uma experiência de usuário mais segura, fornecer garantias de segurança da marca aos anunciantes e cumprir as regulamentações locais e globais.

Atualmente, muitas empresas dependem inteiramente de moderadores humanos para revisar conteúdo de terceiros ou gerado por usuários, enquanto outras simplesmente reagem às reclamações dos usuários para remover imagens, anúncios ou vídeos ofensivos ou inapropriados. No entanto, os moderadores humanos sozinhos não conseguem escalonar para atender a essas necessidades com qualidade ou velocidade suficientes, o que gera uma experiência ruim para o usuário, altos custos para alcançar essa escala ou até mesmo uma perda de reputação da marca. Ao usar o Rekognition para moderação de imagens e vídeos, os moderadores humanos podem revisar um conjunto muito menor de conteúdo, normalmente de 1 a 5% do volume total, já sinalizado pelo machine learning. Isso permite concentrar os esforços em atividades mais valiosas e ainda abordar a moderação de forma completa por uma fração do custo atual. Para configurar a força de trabalho humana e realizar tarefas de revisão humana, você pode usar o Amazon Augmented AI, que já está integrado ao Rekognition.

Você pode aprimorar a precisão do modelo de aprendizado profundo de moderação com o recurso de moderação personalizada. Com a moderação personalizada, você treina um adaptador de moderação personalizado enviando suas imagens e fazendo anotações nessas imagens. O adaptador treinado pode então ser fornecido à DetectModerationLabelsoperação para melhorar seu desempenho em suas imagens. Consulte Aprimorando a precisão com moderação personalizada para obter mais informações.

Rótulos aceitos pelas operações de moderação de conteúdo do Rekognition

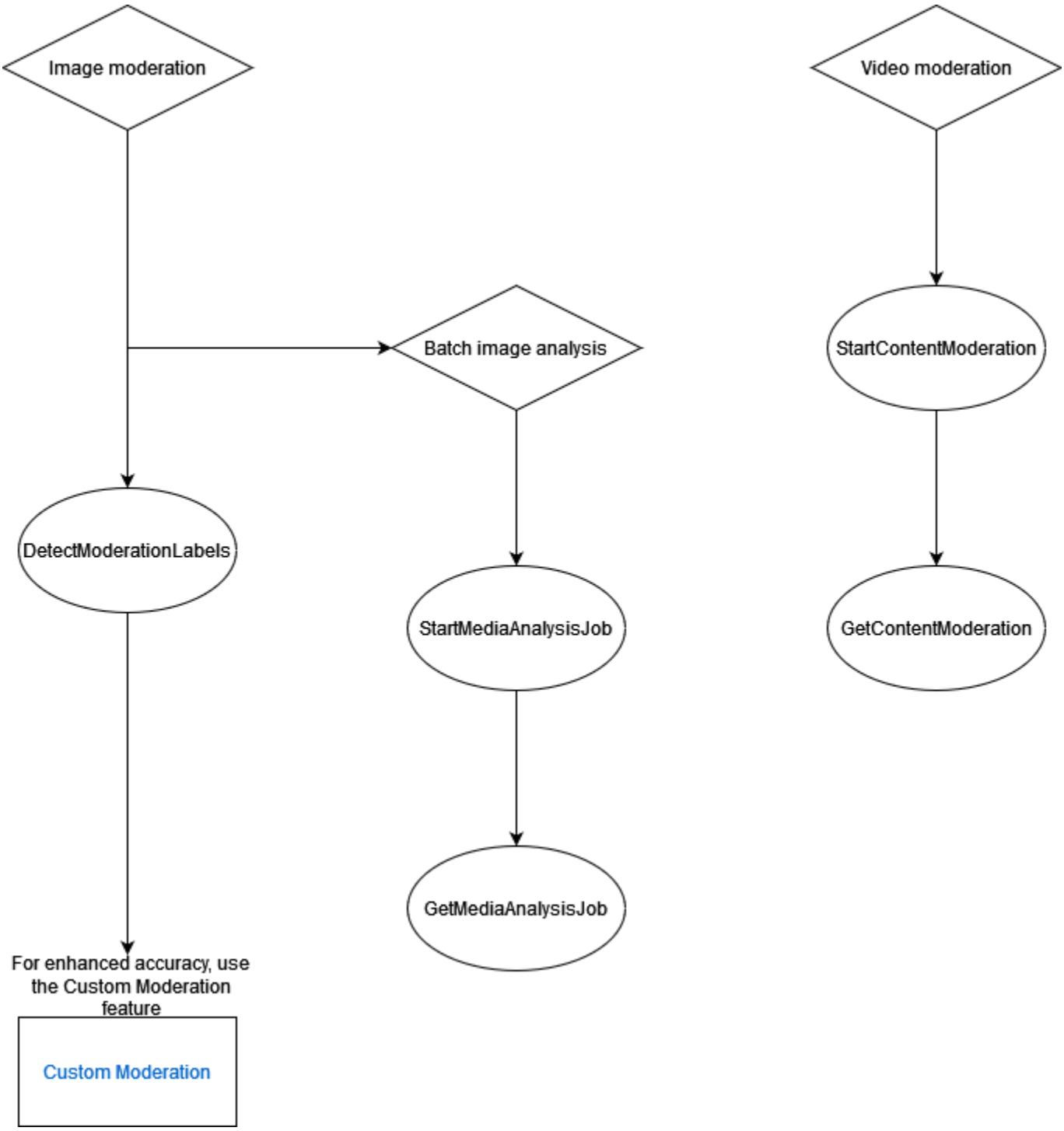
- Para baixar uma lista dos rótulos de moderação, clique aqui.

Tópicos

- Usando a moderação de imagens e vídeos APIs
- Testar a versão 7 da moderação de conteúdo e transformar a resposta da API
- Detectando imagens inapropriadas

- Detectando vídeos armazenados inapropriados
- Aprimorando a precisão com moderação personalizada
- Revisão de conteúdo inadequado com o Amazon Augmented AI

O diagrama a seguir mostra a ordem das operações de chamada, dependendo de suas metas de uso dos componentes de imagem ou vídeo da moderação de conteúdo:



Usando a moderação de imagens e vídeos APIs

Na API do Amazon Rekognition Image, você pode detectar conteúdo impróprio, indesejado ou ofensivo de forma síncrona usando e usando e operando de forma assíncrona. DetectModerationLabelsStartMediaAnalysisJobGetMediaAnalysisJob Você pode usar a API Amazon

Rekognition Video para detectar esse conteúdo de forma assíncrona usando as operações e. StartContentModerationGetContentModeration

Categorias de etiquetas

O Amazon Rekognition usa uma taxonomia hierárquica de três níveis para rotular categorias de conteúdo impróprio, indesejado ou ofensivo. Cada rótulo com nível de taxonomia 1 (L1) tem vários rótulos de taxonomia de nível 2 (L2), e alguns rótulos de taxonomia de nível 2 podem ter rótulos de taxonomia de nível 3 (L3). Isso permite uma classificação hierárquica do conteúdo.

Para cada rótulo de moderação detectado, a API também retorna o `TaxonomyLevel`, que contém o nível (1, 2 ou 3) ao qual o rótulo pertence. Por exemplo, uma imagem pode ser rotulada de acordo com a seguinte categorização:

L1: nudez não explícita de partes íntimas e beijos, L2: nudez não explícita, L3: nudez implícita.

> **Note**
>
> Recomendamos usar categorias L1 ou L2 para moderar seu conteúdo e usar categorias L3 somente para remover conceitos específicos que você não deseja moderar (ou seja, para detectar conteúdo que você talvez não queira categorizar como conteúdo impróprio, indesejado ou ofensivo com base em sua política de moderação).

A tabela a seguir mostra as relações entre os níveis de categoria e os possíveis rótulos para cada nível. Para baixar uma lista dos rótulos de moderação, clique aqui.

Categoria de nível superior (L1)	Categoria de segundo nível (L2)	Categoria de terceiro nível (L3)	Definições
Explícito	Nudez explícita	Genitália masculina exposta	Genitália masculina humana, incluindo o pênis (ereto ou flácido), a bolsa escrotal e qualquer pêlo pubiano perceptível. Este termo é aplicável

	em contextos que envolvam atividade sexual ou qualquer conteúdo visual em que os órgãos genitais masculinos sejam exibidos total ou parcialmente.
Genitália feminina exposta	Partes externas do sistema reprodutor feminino, abrangendo a vulva, a vagina e qualquer pêlo pubiano observável. Esse termo é aplicável em cenários que envolvam atividade sexual ou qualquer conteúdo visual em que esses aspectos da anatomia feminina sejam exibidos total ou parcialmente.

Nádegas ou ânus expostos	Nádegas ou ânus humanos, incluindo casos em que as nádegas estão nuas ou quando são discerníveis por meio de roupas transparentes. A definição se aplica especificamente a situações em que as nádegas ou o ânus são direta e completamente visíveis, excluindo cenários em que qualquer forma de roupa íntima ou roupa oferece cobertura total ou parcial.
Mamilo feminino exposto	Mamilos femininos humanos, incluindo aerola totalmente visível e parcialmente visível (área ao redor dos mamilos) e mamilos.

	Atividade sexual explícita	N/D	Representação de atos sexuais reais ou simulados que englobam relações sexuais humanas, sexo oral, bem como estimulação genital masculina e estimulação genital feminina por outras partes do corpo e objetos. O termo também inclui ejaculação ou fluidos vaginais em partes do corpo e práticas eróticas ou dramatizações envolvendo escravidão, disciplina, domínio e submissão e sadomasoquismo.
	Brinquedos sexuais	N/D	Objetos ou dispositivos usados para estimulação ou prazer sexual, por exemplo, dildo, vibrador, plugue anal, batidas, etc.

Nudez não explícita de partes íntimas e beijos	Nudez não explícita	Costas nuas	Parte posterior humana, onde a maior parte da pele é visível do pescoço até o final da coluna. Este termo não se aplica quando as costas do indivíduo estão parcialmente ou totalmente ocluídas.
		Mamilo masculino exposto	Mamilos humanos masculinos, incluindo mamilos parcialmente visíveis.
		Nádegas parcialmente expostas	Nádegas humanas parcialmente expostas. Este termo inclui uma região parcialmente visível das nádegas ou nádegas devido a roupas curtas ou parte superior parcialmente visível da fenda anal. O termo não se aplica aos casos em que as nádegas estão totalmente nuas.

		Seio feminino parcialmente exposto	Seio feminino humano parcialmente exposto, onde uma parte do seio feminino é visível ou descoberta, sem revelar o seio inteiro. Este termo se aplica quando a região da prega mamária interna é visível ou quando a dobra mamária inferior é visível com o mamilo totalmente coberto ou ocluído.
		Nudez implícita	Um indivíduo que está nu, de topless ou sem fundo, mas com partes íntimas, como nádegas, mamilos ou genitália, cobertas, ocluídas ou não totalmente visíveis.
	Partes íntimas obstruídas	Mamilo feminino obstruído	Representação visual de uma situação em que os mamilos de uma mulher estão cobertos por roupas ou coberturas opacas, mas suas formas são claramente visíveis.

		Genitália masculina obstruída	Representação visual de uma situação em que a genitália ou o pênis de um homem estão cobertos por roupas ou cobertura s opacas, mas sua forma é claramente visível. Este termo se aplica quando a genitália obstruída na imagem está em close-up.
	Beijando na boca	N/D	Representação dos lábios de uma pessoa fazendo contato com os lábios de outra pessoa.
Roupa de banho ou roupa íntima	Roupa de banho ou roupa íntima feminina	N/D	Roupas humanas para roupas de banho femininas (por exemplo, roupas de banho de uma peça, biquínis, tanquínis , etc.) e roupas íntimas femininas (por exemplo, sutiãs, calcinhas, cuecas, lingerie, tangas, etc.)

	Roupa de banho ou roupa íntima masculina	N/D	Roupas humanas para trajes de banho masculinos (por exemplo, calções de banho, calções de banho, etc.) e roupas íntimas masculinas (por exemplo, cuecas, boxers, etc.)
Violência	Armas	N/D	Instrumentos ou dispositivos usados para causar danos ou danos a seres vivos, estruturas ou sistemas. Isso inclui armas de fogo (por exemplo, armas, rifles, metralhad oras etc.), armas afiadas (por exemplo, espadas, facas, etc.), explosivos e munições (por exemplo, mísseis, bombas, balas etc.).
	Violência gráfica	Violência de armas	O uso de armas para causar danos, danos, ferimentos ou morte a si mesmo, a outros indivíduos ou propriedades.

Violência física	O ato de causar danos a outras pessoas ou proprieda des (por exemplo, bater, brigar, puxar cabelos, etc.) ou outro ato de violência envolvendo uma multidão ou vários indivíduos.
Automutilação	O ato de causar danos a si mesmo, geralmente cortando partes do corpo, como braços ou pernas, onde os cortes geralmente são visíveis.
Sangue e sangue	Representação visual da violência contra uma pessoa, um grupo de indivíduos ou animais, envolvend o feridas abertas, derramamento de sangue e partes do corpo mutiladas.
Explosões e explosões	Representação de uma explosão violenta e destrutiva de chamas intensas com fumaça espessa ou poeira e fumaça saindo do solo.

Visualmente perturbador	Morte e emaciação	Corpos enfraquecidos	Corpos humanos extremamente magros e desnutridos, com grave perda física e depleção de tecido muscular e adiposo.
		Cadáveres	Cadáveres humanos na forma de corpos mutilados, cadáveres pendurados ou esqueletos.
	Falhas	Acidente aéreo	Incidentes de veículos aéreos, como aviões, helicópteros ou outros veículos voadores, resultando em danos, ferimentos ou morte. Este termo se aplica quando partes dos veículos aéreos são visíveis.

Drogas e tabaco	Produtos	Comprimidos	Mesas ou cápsulas pequenas, sólidas, geralmente redondas ou ovais. Este termo se aplica a pílulas apresentadas como autônomas, em um frasco ou pacote transparente e não se aplica a uma representação visual de uma pessoa tomando pílulas.
	Parafernália e uso de drogas e tabaco	Fumar	O ato de inalar, expirar e acender substâncias queimadas, incluindo cigarros, charutos, cigarros eletrônicos, narguilé ou cigarro.
Álcool	Uso de álcool	Beber	O ato de beber bebidas alcoólicas em garrafas ou copos de álcool ou licor.

	Bebidas alcoólicas	N/D	Feche uma ou várias garrafas de álcool ou licor, copos ou canecas com álcool ou licor e copos ou canecas com álcool ou licor segurados por um indivíduo. Este termo não se aplica a um indivíduo que bebe garrafas ou copos de álcool ou licor.
Gestos rudes	Dedo do meio	N/D	A representação visual de um gesto com a mão com o dedo médio é estendida para cima enquanto os outros dedos estão dobrados para baixo.
Jogos de aposta	N/D	N/D	O ato de participa r de jogos de azar para ter a chance de ganhar um prêmio em cassinos, por exemplo, cartas de baralho, blackjacks, roleta, caça-níqueis em cassinos, etc.

Símbolos de ódio	Partido Nazista	N/D	Representação visual de símbolos, bandeiras ou gestos associados ao Partido Nazista.
	Supremacia branca	N/D	Representação visual de símbolos ou roupas associados à Ku Klux Klan (KKK) e imagens com bandeiras confederadas.
	Extremista	N/D	Imagens contendo bandeiras de grupos extremistas e terroristas.

Nem todo rótulo na categoria L2 tem um rótulo compatível na categoria L3. Além disso, os rótulos L3 em “Produtos” e os rótulos L2 em “Parafernália e uso de drogas e tabaco” não são precisos. Esses rótulos L2 abrangem conceitos além dos rótulos L3 mencionados e, nesses casos, apenas os rótulos L2 são retornados na resposta da API.

Você determina a adequação do conteúdo ao aplicativo. Por exemplo, imagens de natureza sugestiva podem ser aceitáveis, mas imagens contendo nudez, não. Para filtrar imagens, use a matriz de rótulos ModerationLabel retornada por `DetectModerationLabels` (imagens) e por `GetContentModeration` (vídeos).

Tipo de conteúdo

A API também pode identificar tipos de conteúdo animado ou ilustrado, e o tipo de conteúdo é retornado como parte da resposta.

- Conteúdo animado inclui videogame e animação (por exemplo, desenho animado, quadrinhos, mangá, anime).
- Conteúdo ilustrado inclui desenho, pintura e esboços.

Confiança

Você pode definir o limite de confiança que o Amazon Rekognition usa para detectar conteúdo impróprio especificando o parâmetro de entrada `MinConfidence`. Os rótulos não são retornados para conteúdo impróprio detectado com menos confiança do que `MinConfidence`.

Especificar um valor para `MinConfidence` que seja inferior a 50% provavelmente retornará um número elevado de resultados falsos positivos (ou seja, maior recordação, menor precisão). Por outro lado, especificar `MinConfidence` acima de 50% provavelmente retornará um número menor de resultados falsos positivos (ou seja, menor recordação, maior precisão). Se você não especificar um valor para `MinConfidence`, o Amazon Rekognition retornará rótulos para conteúdo impróprio detectado com pelo menos 50% de confiança.

A matriz `ModerationLabel` contém rótulos nas categorias anteriores e uma confiança estimada na precisão do conteúdo reconhecido. Um rótulo de nível superior é retornado com todos os rótulos de segundo nível que foram identificados. Por exemplo, o Amazon Rekognition pode retornar "Nudez explícita" com uma alta pontuação de confiança como rótulo de alto nível. Isso pode ser suficiente para suas necessidades de filtragem. No entanto, se for necessário, você poderá usar a pontuação de confiança de um rótulo de segundo nível (como "Nudez masculina gráfica") para obter uma filtragem mais granular. Para obter um exemplo, consulte Detectando imagens inapropriadas.

Versionamento

Tanto o Amazon Rekognition Image quanto o Amazon Rekognition Video retornam a versão do modelo de detecção de moderação que é usado para detectar conteúdo impróprio (`ModerationModelVersion`).

Classificação e agregação

Ao recuperar resultados com GetContentModeration, você pode classificar e agregar seus resultados.

Ordem de classificação — A matriz de etiquetas retornadas é classificada por hora. Para classificar por rótulo, especifique `NAME` no parâmetro de entrada `SortBy` para `GetContentModeration`. Se o rótulo aparecer várias vezes no vídeo, haverá várias instâncias do elemento `ModerationLabel`.

Informações do rótulo — O elemento da ModerationLabels matriz contém um `ModerationLabel` objeto que, por sua vez, contém o nome do rótulo e a confiança que o Amazon Rekognition tem na precisão do rótulo detectado. O carimbo de data/hora é a hora em que a `ModerationLabel`

foi detectada, definida como o número de milissegundos decorridos desde o início do vídeo. Para resultados agregados por `SEGMENTS` do vídeo, as estruturas `StartTimestampMillis`, `EndTimestampMillis` e `DurationMillis` são retornadas, definindo a hora de início, a hora de término e a duração de um segmento, respectivamente.

Agregação: especifica como os resultados são agregados quando retornados. O padrão é agregar por `TIMESTAMPS`. Você também pode optar por agregar por `SEGMENTS`, o que agrega os resultados em uma janela de tempo. Somente rótulos detectados durante os segmentos são retornados.

Status de adaptador de moderação personalizado

Os adaptadores de moderação personalizados podem ter um dos seguintes status: TRAINING_IN_PROGRESS, TRAINING_COMPLETED, TRAINING_FAILED, DELETING, DEPRECATED ou EXPIRED. Para obter uma explicação completa dos status desses adaptadores, consulte Gerenciar adaptadores.

> **Note**
>
> O Amazon Rekognition não é uma autoridade e, de forma alguma, afirma ser um filtro exaustivo de conteúdo impróprio ou ofensivo. Além disso, a moderação de imagens e vídeos APIs não detecta se uma imagem inclui conteúdo ilegal, como CSAM.

Testar a versão 7 da moderação de conteúdo e transformar a resposta da API

O Rekognition atualizou o modelo de machine learning para os componentes de imagem e vídeo do atributo de detecção de rótulos de moderação de conteúdo da versão 6.1 para a 7. Essa atualização aprimorou a precisão geral e introduziu várias novas categorias, além de modificar outras.

Se você é um usuário de vídeo atual da versão 6.1, recomendamos executar as seguintes ações para fazer uma transição tranquila para a versão 7:

1. Faça o download e use um SDK privado da AWS (consulte othe section called “AWS SDK e guia de uso para moderação de conteúdo, versão 7 ”) para chamar a StartContentModeration API.
2. Analise a lista atualizada de rótulos e pontuações de confiança retornados na resposta da API ou no console. Ajuste a lógica de pós-processamento do aplicativo adequadamente, se necessário.

3. Sua conta permanecerá na versão 6.1 até 13 de maio de 2024. Se você quiser usar a versão 6.1 após 13 de maio de 2024, entre em contato com a equipe do AWS Support até 30 de abril de 2024 para solicitar uma prorrogação. Podemos prorrogar sua conta para permanecer na versão 6.1 até 10 de junho de 2024. Se não recebermos sua solicitação até 30 de abril de 2024, sua conta será migrada automaticamente para a versão 7.0 a partir de 13 de maio de 2024.

AWS SDK e guia de uso para moderação de conteúdo, versão 7

Faça o download do SDK que corresponde à linguagem de desenvolvimento escolhida e consulte o guia do usuário apropriado.

Link para o SDK	Guia de instalação/usuário
Java-1.X	Guia - Java 1.pdf
Java 2.X	Guia - Java 2.pdf
JavaScript v2	Guia - JavaScript v2.pdf
JavaScript v3	Guia - JavaScript v3.pdf
Python	Guia - Python e AWS CLI.pdf
Ruby	Guia - RubyV3.pdf
go_v1	Guia - GO V1.pdf
go_v2	Guia - GO V2.pdf
DotNet	Guia - .net.pdf
php	Guia - PHP.pdf

Mapeamentos de rótulos para as versões 6.1 a 7

A versão 7 de moderação de conteúdo adicionou novas categorias de rótulos e modificou os nomes de rótulos existentes. Consulte a tabela de taxonomia encontrada em the section called “ Categorias de etiquetas ” ao decidir como mapear rótulos da versão 6.1 na versão 7.

Alguns exemplos de mapeamentos de rótulos são encontrados na seção a seguir. Recomendamos revisar esses mapeamentos e as definições de rótulos antes de fazer as atualizações necessárias com base na lógica de pós-processamento do seu aplicativo.

Mapeamento de esquemas L1

Se você usa uma lógica de pós-processamento que filtra somente na categoria de nível superior (L1) (como `Explicit Nudity`, `Suggestive`, `Violence` etc.), consulte a tabela abaixo para atualizar seu código.

V6.1 L1	V7 L1
Nudez explícita	Explícito
Sugestiva	Nudez não explícita de partes íntimas e beijos
	Roupa de banho ou roupa íntima
Violência	Violência
Visualmente perturbador	Visualmente perturbador
Gestos rudes	Gestos rudes
Drogas	Drogas e tabaco
Tabaco	Drogas e tabaco
Álcool	Álcool
Jogos de aposta	Jogos de aposta
Símbolos de ódio	Símbolos de ódio

Mapeamento de esquemas L2

Se você usa uma lógica de pós-processamento que filtra as categorias L1 e L2 (como `Explicit Nudity / Nudity, Suggestive / Female Swimwear Or Underwear`, `Violence / Weapon Violence` etc.), consulte a tabela abaixo para atualizar seu código.

V6.1 L1	V6.1 L2	V7 L1	V7 L2	V7 L3	V7 ContentTypes
Nudez explícita	Nudez	Explícito	Nudez explícita	Mamilo feminino exposto	
				Nádegas ou ânus expostos	
	Nudez masculina gráfica	Explícito	Nudez explícita	Genitália masculina exposta	
	Nudez feminina gráfica	Explícito	Nudez explícita	Genitália feminina exposta	
	Atividade sexual	Explícito	Atividade sexual explícita		
	Nudez explícita ilustrada	Explícito	Nudez explícita		Mapeie para “Animado” e “Ilustrado”
	Nudez explícita ilustrada	Explícito	Atividade sexual explícita		Mapeie para “Animado” e “Ilustrado”
	Brinquedos adultos	Explícito	Brinquedos sexuais		
Sugestiva	Roupa de banho ou roupa íntima feminina	Roupa de banho ou roupa íntima	Roupa de banho ou roupa íntima feminina		

Roupa de banho ou roupa íntima masculina	Roupa de banho ou roupa íntima	Roupa de banho ou roupa íntima masculina	
Nudez parcial	Nudez não explícita de partes íntimas e beijos	Nudez não explícita	Nudez implícita
Homem sem camisa	Nudez não explícita de partes íntimas e beijos	Nudez não explícita	Mamilo masculino exposto
Roupas transparentes	Nudez não explícita de partes íntimas e beijos	Nudez não explícita	
	Nudez não explícita de partes íntimas e beijos	Partes íntimas obstruídas	
Situações sexuais	Nudez não explícita de partes íntimas e beijos	Beijando na boca	

Violência	Violência gráfica ou excesso de sangue	Violência	Violência gráfica	Sangue e sangue
	Violência física	Violência	Violência gráfica	Violência física
	Violência de armas	Violência	Violência gráfica	Violência de armas
	Armas	Violência	Armas	
	Autolesão	Violência	Violência gráfica	Automutilação
Visualmente perturbador	Corpos enfraquecidos	Visualmente perturbador	Morte e emaciação	Corpos enfraquecidos
	Cadáveres	Visualmente perturbador	Morte e emaciação	Cadáveres
	Enforcamento	Visualmente perturbador	Morte e emaciação	Cadáveres
	Acidente aéreo	Visualmente perturbador	Falhas	Acidente aéreo
	Explosões e estouros	Violência	Violência gráfica	Explosões e explosões
Gestos rudes	Dedo do meio	Gestos rudes	Dedo do meio	
Drogas	Produtos farmacêuticos	Drogas e tabaco	Produtos	

	Uso de drogas	Drogas e tabaco	Paraferná lia e uso de drogas e tabaco	
	Comprimidos	Drogas e tabaco	Produtos	Comprimidos
	Parafernália de drogas	Drogas e tabaco	Paraferná lia e uso de drogas e tabaco	
Tabaco	Produtos de tabaco	Drogas e tabaco	Produtos	
	Fumar	Drogas e tabaco	Paraferná lia e uso de drogas e tabaco	Fumar
Álcool	Beber	Álcool	Uso de álcool	Beber
	Bebidas alcoólicas	Álcool	Bebidas alcoólicas	
Jogos de aposta	Jogos de aposta	Jogos de aposta		
Símbolos de ódio	Partido Nazista	Símbolos de ódio	Partido Nazista	
	Supremacia branca	Símbolos de ódio	Supremacia branca	
	Extremista	Símbolos de ódio	Extremista	

Detectando imagens inapropriadas

Você pode usar a DetectModerationLabelsoperação para determinar se uma imagem contém conteúdo impróprio ou ofensivo. Para obter uma lista de rótulos de moderação no Amazon Rekognition, consulte Uso da moderação de imagens e vídeos. APIs

Detectando conteúdo impróprio em uma imagem

A imagem deve estar em um formato .jpg ou .png. Você pode fornecer a imagem de entrada como uma matriz de bytes de imagem (bytes de imagem codificados em base64) ou especificar um objeto Amazon S3. Nestes procedimentos, você faz upload de uma imagem (.jpg ou .png) em seu bucket do S3.

Para executar esses procedimentos, você precisa ter o SDK AWS CLI ou o AWS SDK apropriado instalado. Para obter mais informações, consulte Comece a usar o Amazon Rekognition. A conta da AWS que você usa deve ter permissões de acesso à API do Amazon Rekognition. Para obter mais informações, consulte Ações definidas pelo Amazon Rekognition.

Para detectar rótulos de moderação em uma imagem (SDK)

1. Se ainda não tiver feito isso:

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Faça upload de uma imagem no bucket do S3.

   Para obter instruções, consulte Como fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

3. Use os exemplos a seguir para chamar a operação `DetectModerationLabels`.

   Java

   Este exemplo mostra nomes de rótulos de conteúdo inadequados, níveis de confiança e o rótulo principal detectados para rótulos de moderação detectados.

Substitua os valores de `amzn-s3-demo-bucket` e `photo` pelo nome do bucket do S3 e o nome do arquivo de imagem usado na etapa 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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
//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

Esse AWS CLI comando exibe a saída JSON para a operação da `detect-moderation-labels` CLI.

Substitua `amzn-s3-demo-bucket` e `input.jpg` pelo nome do bucket do S3 e o nome do arquivo de imagem usado na etapa 2. Substitua o valor de `profile_name` com o nome do seu perfil de desenvolvedor. Para usar um adaptador, forneça o ARN da versão do projeto para o parâmetro `project-version`.

```
aws rekognition detect-moderation-labels --image "{S3Object:{Bucket:<amzn-s3-
demo-bucket>,Name:<image-name>}}" \
--profile profile-name \
--project-version "ARN"
```

Se você estiver acessando a CLI em um dispositivo Windows, use aspas duplas em vez de aspas simples e escape das aspas duplas internas com barra invertida (ou seja, \) para

resolver quaisquer erros de analisador que você possa encontrar. Para obter um exemplo, veja o seguinte:

```
aws rekognition detect-moderation-labels --image "{\"S3Object\":{\"Bucket\":
\"amzn-s3-demo-bucket\",\"Name\":\"image-name\"}}" \
--profile profile-name
```

Python

Este exemplo mostra nomes de rótulos de conteúdo inadequados ou ofensivos detectados, níveis de confiança e o rótulo principal para rótulos de conteúdo inadequados detectados.

Na função `main`, substitua os valores de `amzn-s3-demo-bucket` e `photo` pelo nome do bucket do S3 e o nome do arquivo de imagem usado na etapa 2. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

Este exemplo mostra nomes de rótulos de conteúdo impróprios ou ofensivos, níveis de confiança e o rótulo principal detectados para rótulos de moderação detectados.

Substitua os valores de `amzn-s3-demo-bucket` e `photo` pelo nome do bucket do S3 e o nome do arquivo de imagem usado na etapa 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 solicitação de operação

A entrada de `DetectModerationLabels` é uma imagem. Neste exemplo de entrada JSON, a imagem de origem é carregada de um bucket do Amazon S3. `MinConfidence` é a confiança mínima que o Amazon Rekognition Image deve ter na precisão da etiqueta detectada para que ela seja retornada na resposta.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "input.jpg"
        }
    },
    "MinConfidence": 60
}
```

DetectModerationLabels resposta da operação

O `DetectModerationLabels` pode recuperar imagens de entrada a partir de um bucket do S3 ou você pode fornecê-las como bytes de imagens. O exemplo a seguir é a resposta de uma chamada para `DetectModerationLabels`.

Na resposta JSON de exemplo a seguir, observe o seguinte:

- Informações de detecção de imagem impróprias — O exemplo mostra uma lista de rótulos para conteúdo impróprio ou ofensivo encontrado na imagem. A lista inclui o rótulo de nível superior e cada rótulo de segundo nível detectado na imagem.

  Rótulo — Cada rótulo tem um nome, uma estimativa da confiança que o Amazon Rekognition tem de que a etiqueta é precisa e o nome do rótulo principal. O nome pai de um rótulo de nível superior é "".

  Confiança do rótulo — Cada rótulo tem um valor de confiança entre 0 e 100 que indica a porcentagem de confiança que o Amazon Rekognition tem de que o rótulo está correto. Você especifica o nível de confiança necessário para que um rótulo seja retornado na resposta na solicitação de operação da 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"
```

```
        }
    ]
}
```

Detectando vídeos armazenados inapropriados

A detecção de conteúdo impróprio ou ofensivo do Amazon Rekognition Video em vídeos armazenados é uma operação assíncrona. Para começar a detectar conteúdo impróprio ou ofensivo, ligue. StartContentModeration O Amazon Rekognition Video publica o status de conclusão da análise de vídeo em um tópico do Amazon Simple Notification Service. Se a análise do vídeo for bem-sucedida, chame GetContentModeration para obter os resultados da análise. Para obter mais informações sobre como iniciar uma análise de vídeo e obter os resultados, consulte Chamando as operações de vídeo do Amazon Rekognition Video. Para obter uma lista de rótulos de moderação no Amazon Rekognition, consulte Uso da moderação de imagens e vídeos. APIs

Este procedimento expande o código em Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK), que usa uma fila do Amazon Simple Queue Service para obter o status de conclusão de uma solicitação de análise de vídeo.

Para detectar conteúdo impróprio ou ofensivo em um vídeo armazenado em um bucket (SDK) do Amazon S3

1. Execute Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK).
2. Adicione o código a seguir à classe `VideoDetect` criada por você na etapa 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);

        }
```

Na função `main`, substitua as linhas:

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

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

por:

```
        StartUnsafeContentDetection(amzn-s3-demo-bucket, video);

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

Java V2

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
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
```

Na função `main`, substitua as linhas:

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

por:

```
    analyzer.StartUnsafeContent()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetUnsafeContentResults()
```

> **Note**
>
> Se você já tiver executado um exemplo de vídeo diferente de Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK), o código a ser substituído poderá ser diferente.

3. Execute o código. É exibida uma lista de rótulos de conteúdo impróprio detectados no vídeo.

GetContentModeration resposta da operação

A resposta de `GetContentModeration` é uma matriz, `ModerationLabels`, de objetos ContentModerationDetection. A matriz contém um elemento para cada vez que um rótulo de conteúdo impróprio é detectado. Dentro de um `ContentModerationDetectionObject` objeto, ModerationLabelcontém informações sobre um item detectado de conteúdo impróprio ou ofensivo. `Timestamp`é a hora, em milissegundos, a partir do início do vídeo, em que a etiqueta foi detectada. Os rótulos são organizados hierarquicamente da mesma forma que os rótulos detectados pela análise inadequada de imagens de conteúdo. Para obter mais informações, consulte Como moderar um conteúdo.

Veja a seguir um exemplo de resposta de `GetContentModeration`, classificada por `NAME` e agregada por `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"
    }
}
```

Veja a seguir um exemplo de resposta de `GetContentModeration`, classificada por `NAME` e agregada por `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"
    }
}
```

Aprimorando a precisão com moderação personalizada

A DetectModerationLabelsAPI do Amazon Rekognition permite detectar conteúdo impróprio, indesejado ou ofensivo. O recurso de moderação personalizada do Rekognition permite que você aprimore a precisão usando adaptadores. DetectModerationLabels Os adaptadores são componentes modulares que podem ser adicionados a um modelo existente de aprendizado profundo do Rekognition, ampliando seus recursos para as tarefas nas quais ele é treinado. Ao criar um adaptador e fornecê-lo para a DetectModerationLabelsoperação, você pode obter maior precisão nas tarefas de moderação de conteúdo relacionadas ao seu caso de uso específico.

Ao personalizar o modelo de moderação de conteúdo do Rekognition para rótulos de moderação específicos, você deve criar um projeto e treinar um adaptador em um conjunto de imagens que você fornece. Em seguida, você pode verificar iterativamente o desempenho do adaptador e treiná-lo novamente até o nível de precisão desejado. Os projetos são usados para conter as diferentes versões dos adaptadores.

Você pode usar o console do Rekognition para criar projetos e adaptadores. Como alternativa, você pode usar um AWS SDK e o associado APIs para criar um projeto, treinar um adaptador e gerenciar seus adaptadores.

Criação e uso de adaptadores

Os adaptadores são componentes modulares que podem ser adicionados ao modelo de aprendizado profundo existente do Rekognition, ampliando seus recursos para as tarefas nas quais ele é treinado. Ao treinar um modelo de aprendizado profundo com adaptadores, você pode obter maior precisão nas tarefas de análise de imagens relacionadas ao seu caso de uso específico.

Para criar e usar um adaptador, você deve fornecer dados de treinamento e teste ao Rekognition. Você pode fazer isso de duas maneiras diferentes:

- Análise e verificação em massa: você pode criar um conjunto de dados de treinamento analisando em massa imagens que o Rekognition analisará e atribuirá rótulos. Em seguida, você pode revisar as anotações geradas para suas imagens e verificar ou corrigir as previsões. Para obter mais informações sobre como a análise em massa de imagens funciona, consulte Análise em massa.
- Anotação manual: com essa abordagem, você cria seus dados de treinamento carregando e anotando imagens. Você cria seus dados de teste carregando e anotando imagens ou dividindo-os automaticamente.

Escolha um dos tópicos a seguir para saber mais:

Tópicos

- Análise e verificação em massa
- Anotação manual

Análise e verificação em massa

Com essa abordagem, você carrega um grande número de imagens que deseja usar como dados de treinamento e, em seguida, usa o Rekognition para obter previsões para essas imagens, que atribui rótulos automaticamente a elas. Você pode usar essas previsões como ponto de partida para seu adaptador. Você pode verificar a precisão das previsões e depois treinar o adaptador com base nas previsões verificadas. Isso pode ser feito com o AWS console.

Análise em massa e moderação personalizada

Faça upload de imagens para análise em massa

Para criar um conjunto de dados de treinamento para seu adaptador, faça upload de imagens em massa para o Rekognition prever rótulos. Para obter melhores resultados, forneça o máximo possível de imagens para treinamento até o limite de 10.000 e garanta que as imagens representem todos os aspectos do seu caso de uso.

Ao usar o AWS console, você pode carregar imagens diretamente do seu computador ou fornecer um bucket do Amazon Simple Storage Service que armazena suas imagens. No entanto, ao usar o APIs Rekognition com um SDK, você deve fornecer um arquivo de manifesto que faça referência às imagens armazenadas em um bucket do Amazon Simple Storage Service. Consulte Análise em massa para obter mais informações.

Revise as previsões

Depois de enviar suas imagens para o console do Rekognition, o Rekognition gerará rótulos para elas. Em seguida, você pode verificar as previsões como uma das seguintes categorias: verdadeiro positivo, falso positivo, verdadeiro negativo, falso negativo. Depois de verificar as previsões, você pode treinar um adaptador com base no seu feedback.

Treine o adaptador

Depois de concluir a verificação das previsões retornadas pela análise em massa, você poderá iniciar o processo de treinamento do seu adaptador.

Obtenha o AdapterId

Depois que o adaptador for treinado, você poderá obter a ID exclusiva para seu adaptador usar com a análise de imagem do Rekognition. APIs

Chame a operação da API

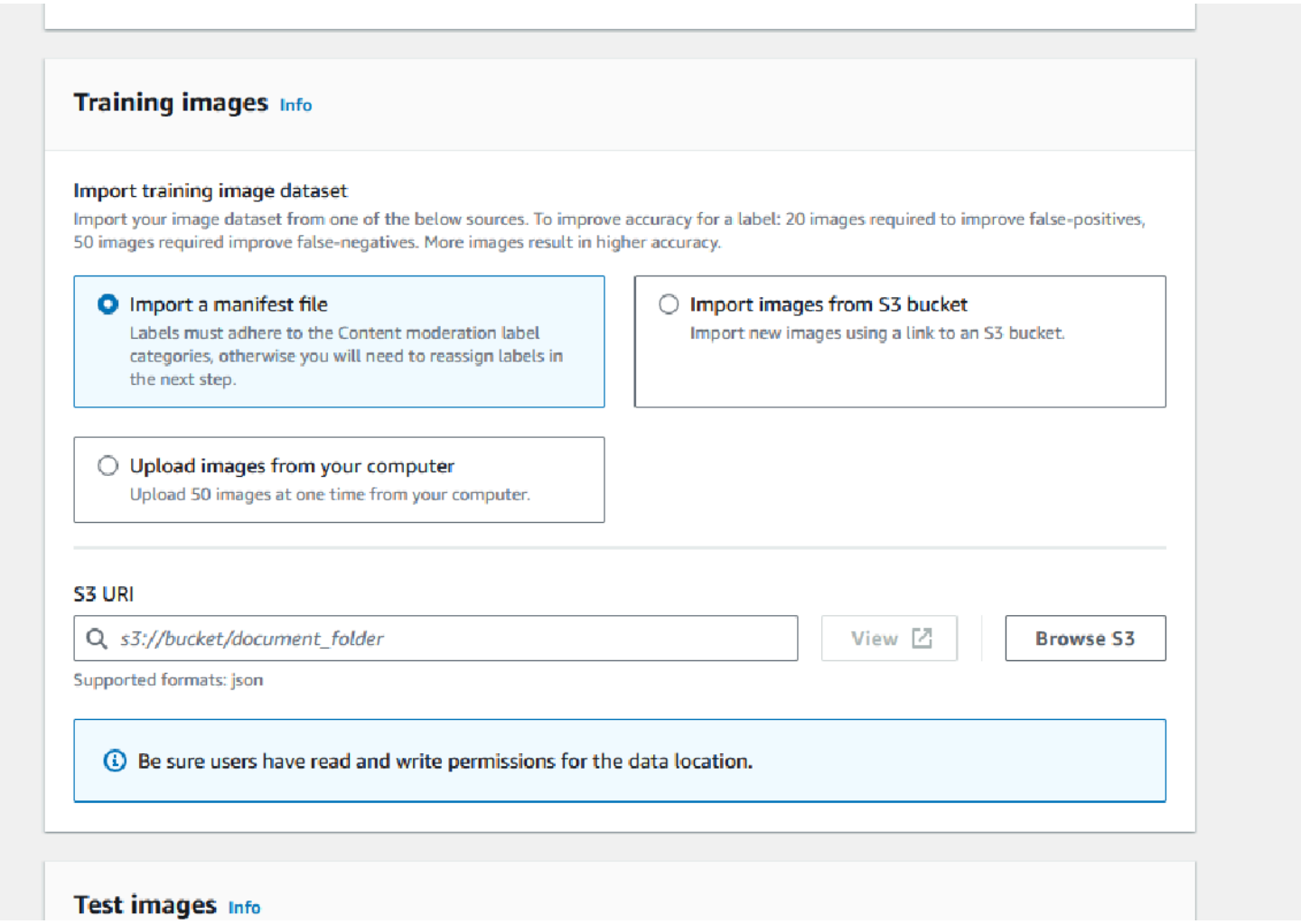
Para aplicar seu adaptador personalizado, forneça sua ID ao chamar uma das análises de imagem APIs que suporta adaptadores. Isso aprimora a precisão das previsões para suas imagens.

Anotação manual

Com essa abordagem, você cria seus dados de treinamento carregando e anotando imagens manualmente. Você cria seus dados de teste carregando e anotando imagens de teste ou dividindo automaticamente para que o Rekognition use automaticamente uma parte dos seus dados de treinamento como imagens de teste.

Carregar e anotar imagens

Para treinar o adaptador, você precisará fazer o upload de um conjunto de imagens de amostra representativas do seu caso de uso. Para obter melhores resultados, forneça o máximo possível de imagens para treinamento até o limite de 10.000 e garanta que as imagens representem todos os aspectos do seu caso de uso.



Ao usar o AWS console, você pode fazer upload de imagens diretamente do seu computador, fornecer um arquivo de manifesto ou fornecer um bucket do Amazon S3 que armazene suas imagens.

No entanto, ao usar o APIs Rekognition com um SDK, você deve fornecer um arquivo de manifesto que faça referência às imagens armazenadas em um bucket do Amazon S3.

Você pode usar a interface de anotação do console do Rekognition para anotar suas imagens. Anote suas imagens marcando-as com rótulos, isso estabelece uma "verdade fundamental" para o treinamento. Você também deve designar conjuntos de treinamento e teste ou usar o recurso de

divisão automática antes de treinar um adaptador. Ao terminar de designar seus conjuntos de dados e anotar suas imagens, você pode criar um adaptador com base nas imagens anotadas em seu conjunto de teste. Em seguida, você pode avaliar o desempenho do seu adaptador.

Crie um conjunto de testes

Você precisará fornecer um conjunto de testes anotado ou usar o recurso de divisão automática. O conjunto de treinamento é usado para realmente treinar o adaptador. O adaptador aprende os padrões contidos nessas imagens anotadas. O conjunto de teste é usado para avaliar o desempenho do modelo antes de finalizar o adaptador.

Treine o adaptador

Depois de terminar de anotar os dados de treinamento ou fornecer um arquivo de manifesto, você poderá iniciar o processo de treinamento para seu adaptador.

Obtenha a ID do adaptador

Depois que o adaptador for treinado, você poderá obter a ID exclusiva para seu adaptador usar com a análise de imagem do Rekognition. APIs

Chame a operação da API

Para aplicar seu adaptador personalizado, forneça sua ID ao chamar uma das análises de imagem APIs que suporta adaptadores. Isso aprimora a precisão das previsões para suas imagens.

Preparando seus conjuntos de dados

A criação de um adaptador exige que você forneça ao Rekognition dois conjuntos de dados, um conjunto de dados de treinamento e um conjunto de dados de teste. Cada conjunto de dados é composto por dois elementos: imagens e anotações/rótulos. As seções a seguir explicam para que rótulos e imagens são usados e como eles se juntam para criar conjuntos de dados.

Imagens

Você precisará treinar um adaptador em amostras representativas de suas imagens. Ao selecionar imagens para treinamento, tente incluir pelo menos algumas imagens que demonstrem a resposta esperada para cada um dos rótulos que você está segmentando com seu adaptador.

Para criar um conjunto de dados de treinamento, você precisa fornecer um dos dois tipos de imagem a seguir:

- Imagens com previsões de falsos positivos. Por exemplo, quando um modelo básico prevê que uma imagem tem álcool, mas isso não acontece.
- Imagens com previsões de falsos negativos. Por exemplo, quando um modelo básico prevê que uma imagem não tem álcool, mas tem.

Para criar um conjunto de dados balanceado, é recomendável fornecer um dos dois tipos de imagem a seguir:

- Imagens com previsões verdadeiramente positivas. Por exemplo, quando um modelo básico prediz corretamente que uma imagem tem álcool. É recomendável fornecer essas imagens se você fornecer imagens de falso positivo.
- Imagens com previsões de Falso negativo. Por exemplo, quando um modelo básico prediz corretamente que uma imagem não tem álcool. É recomendável fornecer essas imagens se você fornecer imagens em falsos negativos.

Rótulos

Um rótulo se refere a qualquer um dos seguintes: objetos, eventos, conceitos ou atividades. Para moderação de conteúdo, um rótulo é uma instância de conteúdo impróprio, indesejado ou ofensivo.

No contexto da criação de um adaptador treinando o modelo básico do Rekognition, quando um rótulo é atribuído a uma imagem, ele é chamado de anotação. Ao treinar um adaptador com o Rekognition Console, você usará o console para adicionar anotações às suas imagens escolhendo uma etiqueta e marcando as imagens que correspondam à etiqueta. Por meio desse processo, o modelo aprende a identificar elementos de suas imagens com base no rótulo atribuído. Esse processo de vinculação permite que o modelo se concentre no conteúdo mais relevante quando um adaptador é criado, resultando em maior precisão na análise de imagens.

Como alternativa, você pode fornecer arquivos de manifesto, que contêm informações sobre imagens e as anotações que as acompanham.

Conjuntos de dados de treinamento e teste

O conjunto de dados de treinamento é a base para ajustar o modelo e criar um adaptador personalizado. Você deve fornecer um conjunto de dados de treinamento anotado para o modelo aprender. O modelo aprende com esse conjunto de dados para melhorar seu desempenho no tipo de imagem que você fornece.

Para melhorar a precisão, você deve criar seu conjunto de dados de treinamento anotando/rotulando imagens. É possível fazer isso de duas maneiras:

- Atribuição manual de rótulos: você pode usar o Rekognition Console para criar um conjunto de dados de treinamento fazendo o upload das imagens que você deseja que seu conjunto de dados contenha e, em seguida, atribuir manualmente rótulos a essas imagens.
- Arquivo de manifesto — Você pode usar um arquivo de manifesto para treinar seu adaptador. O arquivo de manifesto contém informações sobre as anotações verdadeiras de suas imagens de treinamento e teste, bem como a localização de suas imagens de treinamento. Você pode fornecer o arquivo de manifesto ao treinar um adaptador usando o APIs Rekognition ou ao usar o Console. AWS

O conjunto de dados de teste é usado para avaliar o desempenho do adaptador após o treinamento. Para garantir uma avaliação confiável, o conjunto de dados de teste é criado usando uma fatia do conjunto de dados de treinamento original que o modelo nunca viu antes. Esse processo garante que o desempenho do adaptador seja avaliado com novos dados, criando medidas e métricas precisas. Para obter melhorias de precisão ideais, consulte Práticas recomendadas para adaptadores de treinamento.

Gerenciando adaptadores com a AWS CLI e SDKs

O Rekognition permite que você use vários recursos que utilizam modelos de visão computacional pré-treinados. Com esses modelos, você pode realizar tarefas como detecção de rótulos e moderação de conteúdo. Você também pode personalizar esses determinados modelos usando um adaptador.

Você pode usar a criação e o gerenciamento de projetos do Rekognition APIs (como e CreateProjectVersion) para criar CreateProjecte treinar adaptadores. As páginas a seguir descrevem como usar as operações de API para criar, treinar e gerenciar seus adaptadores usando o AWS console, o AWS SDK escolhido ou a CLI AWS .

Depois de treinar um adaptador, você pode usá-lo ao executar a inferência com os recursos compatíveis. Atualmente, há suporte para adaptadores ao usar o recurso de moderação de conteúdo.

Ao treinar um adaptador usando um AWS SDK, você deve fornecer seus rótulos verdadeiros (anotações de imagem) na forma de um arquivo de manifesto. Como alternativa, você pode usar o Console do Rekognition para criar e treinar um adaptador.

> **Note**
>
> Os adaptadores não podem ser copiados. Apenas as versões do projeto Rekognition Custom Labels podem ser copiadas.

Tópicos

- Status do adaptador
- Criação de um projeto
- Descrevendo projetos
- Excluir um projeto
- Criando uma versão do projeto
- Descrevendo uma versão do projeto
- Excluindo uma versão do projeto

Status do adaptador

O adaptador de moderação personalizado (versões de projetos) pode estar em um dos seguintes status:

- TRAINING_IN_PROGRESS: o adaptador está em processo de treinamento nos arquivos que você forneceu como documentos de treinamento.
- TRAINING_COMPLETED: o adaptador concluiu com êxito o treinamento e está pronto para que você analise seu desempenho.
- TRAINING_FAILED: o adaptador falhou ao concluir seu treinamento por algum motivo. Consulte o arquivo de manifesto de saída e o resumo do manifesto de saída para obter informações sobre a causa da falha.
- DELETING: o adaptador está em processo de ser excluído.
- DEPRECATED: o adaptador foi treinado em uma versão mais antiga do modelo básico de moderação de conteúdo. Está em um período de carência e expirará dentro de 60 a 90 dias após o lançamento da nova versão do modelo básico. Durante o período de carência, você ainda pode usar o adaptador para inferência DetectModerationLabelsou operações de StartMediaAnalysisJobAPI. Consulte o console de moderação personalizado para saber a data de validade dos seus adaptadores.

- EXPIRADO - O adaptador foi treinado em uma versão mais antiga do modelo básico de moderação de conteúdo e não pode mais ser usado para obter resultados personalizados com as operações da StartMediaAnalysisJob API DetectModerationLabels ou da API. Se um adaptador expirado for especificado em uma solicitação de inferência, ele será ignorado e a resposta será retornada da versão mais recente do modelo básico de moderação personalizada.

Criação de um projeto

Com a CreateProjectoperação, você pode criar um projeto que conterá um adaptador para as operações de detecção de etiquetas do Rekognition. Um projeto é um grupo de recursos e, no caso de operações de detecção de rótulos DetectModerationLabels, como, um projeto permite armazenar adaptadores que podem ser usados para personalizar o modelo básico do Rekognition. Ao invocar CreateProject, você fornece o nome do projeto que deseja criar para o ProjectName argumento.

Para criar um projeto com o AWS console:

- Faça login no Console do Rekognition
- Clique em Moderação personalizada
- Escolha Criar projeto
- Selecione Criar um novo projeto ou Adicionar a um projeto existente
- Adicione um Nome do Projeto
- Adicione um Nome do adaptador
- Adicione uma descrição, se desejar
- Escolha como você deseja importar suas imagens de treinamento: arquivo de manifesto, do bucket do S3 ou do seu computador
- Escolha se você deseja dividir automaticamente seus dados de treinamento ou importar um arquivo de manifesto
- Selecione se você deseja ou não que o projeto seja atualizado automaticamente
- Clique em Criar projeto

Para criar um projeto com a AWS CLI e o SDK:

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e o. AWS SDKs Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.
2. Use o código a seguir para criar um projeto:

CLI

```
# Request
# Creating Content Moderation Project
aws rekognition create-project \
    --project-name "project-name" \
    --feature CONTENT_MODERATION \
    --auto-update ENABLED
    --profile profile-name
```

Descrevendo projetos

Você pode usar a DescribeProjectsAPI para obter informações sobre seus projetos, incluindo informações sobre todos os adaptadores associados a um projeto.

Para descrever projetos com a AWS CLI e o SDK:

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e o. AWS SDKs Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.
2. Use o código a seguir para descrever um projeto:

CLI

```
# Request
# Getting CONTENT_MODERATION project details
aws rekognition describe-projects \
    --features CONTENT_MODERATION
    --profile profile-name
```

Excluir um projeto

Você pode excluir um projeto usando o console do Rekognition ou chamando a API. DeleteProject Para excluir um projeto, primeiro você deve excluir cada um dos adaptadores associados. Um projeto ou modelo excluído não pode ser excluído.

Para excluir um projeto com o AWS console:

- Faça login no Console do Rekognition.
- Clique em Moderação personalizada.
- Você deve excluir cada adaptador associado ao seu projeto antes de excluir o projeto em si. Exclua todos os adaptadores associados ao projeto selecionando o adaptador e, em seguida, selecionando Excluir.
- Selecione o projeto e, em seguida, selecione o botão Excluir.

Para excluir um projeto com a AWS CLI e o SDK:

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e o. AWS SDKs Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.
2. Use o código a seguir para excluir um projeto:

CLI

```
aws rekognition delete-project
  --project-arn project_arn \
  --profile profile-name
```

Criando uma versão do projeto

Você pode treinar um adaptador para implantação usando a CreateProjectVersionoperação. CreateProjectVersion primeiro cria uma nova versão de um adaptador associado a um projeto e, em seguida, começa a treinar o adaptador. A resposta de CreateProjectVersion é um Amazon Resource Name (ARN) para a versão do modelo. O treinamento demora um pouco para ser concluído. Você pode obter o status atual ligando DescribeProjectVersions. Ao treinar um modelo, o Rekognition usa os conjuntos de dados de treinamento e teste associados ao projeto. Você cria conjuntos de dados usando o console. Para obter mais informações, consulte a seção sobre conjuntos de dados.

Para criar uma versão do projeto com o console do Rekognition:

- Faça login no console do AWS Rekognition
- Clique em Moderação personalizada

- Selecione um projeto.
- Na página "Detalhes do projeto", escolha Criar adaptador
- Na página "Criar um projeto", preencha os detalhes necessários para Detalhes do projeto, imagens de treinamento e imagens de teste e selecione Criar projeto .
- Na página "Atribuir rótulos às imagens", adicione rótulos às suas imagens e, ao terminar, selecione Iniciar treinamento

Para criar uma versão do projeto com a AWS CLI e o SDK:

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e o. AWS SDKs Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.
2. Use o código a seguir para criar uma versão do projeto:

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
```

Descrevendo uma versão do projeto

Você pode listar e descrever os adaptadores associados a um projeto usando a DescribeProjectVersionsoperação. Você pode especificar até 10 versões do modelo em ProjectVersionArns. Se você não especificar um valor, as descrições de todas as versões do modelo no projeto serão retornadas.

Para descrever uma versão do projeto com a AWS CLI e o SDK:

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e o. AWS SDKs Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Use o código a seguir para descrever uma versão do projeto:

CLI

```
aws rekognition describe-project-versions
  --project-arn project_arn \
  --version-names [versions]
```

Excluindo uma versão do projeto

Você pode excluir um adaptador Rekognition associado a um projeto usando a operação. DeleteProjectVersion Você não pode excluir um adaptador se ele estiver em execução ou em treinamento. Para verificar o status de um adaptador, chame a DescribeProjectVersions operação e verifique o campo Status retornado por ela. Para interromper uma chamada de adaptador em execução StopProjectVersion. Se o modelo estiver treinando, espere até que ele termine o treinamento para excluí-lo. Você deve excluir cada adaptador associado ao seu projeto antes de excluir o projeto em si.

Para excluir uma versão do projeto com o console do Rekognition:

- Faça login no Console do Rekognition
- Clique em Moderação personalizada
- Na guia Projetos, você pode ver todos os seus projetos e adaptadores associados. Selecione um adaptador e, em seguida, selecione Excluir.

Para excluir uma versão do projeto com a AWS CLI e o SDK:

1. Se você ainda não tiver feito isso, instale e configure a AWS CLI e o. AWS SDKs Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.
2. Use o código a seguir para excluir uma versão do projeto:

CLI

```
# Request
aws rekognition delete-project-version
  --project-version-arn model_arn \
  --profile profile-name
```

Tutorial de adaptador de moderação personalizado

Este tutorial mostra como criar, treinar, avaliar, usar e gerenciar adaptadores usando o Console do Rekognition. Para criar, usar e gerenciar adaptadores com o AWS SDK, consulte. Gerenciando adaptadores com a AWS CLI e SDKs

Os adaptadores permitem que você aprimore a precisão das operações de API do Rekognition, personalizando o comportamento do modelo para atender às suas próprias necessidades e casos de uso. Depois de criar um adaptador com este tutorial, você poderá usá-lo ao analisar suas próprias imagens com operações como DetectModerationLabels, além de retreinar o adaptador para futuras melhorias.

Neste tutorial, você aprenderá como:

- Criar um projeto usando o console do Rekognition
- Anotar os dados de treinamento
- Treinar o adaptador no conjunto de dados de treinamento
- Analisar o desempenho do adaptador
- Usar o adaptador para análisar a imagem

Pré-requisitos

Antes de concluir este tutorial, é recomendável que você leia Criação e uso de adaptadores.

Para criar um adaptador, você pode usar a ferramenta de console do Rekognition para criar um projeto, carregar e anotar suas próprias imagens e, em seguida, treinar um adaptador nessas imagens. Consulte Criar um projeto e treinar um adaptador para começar.

Como alternativa, você pode usar o console ou a API do Rekognition para recuperar previsões de imagens e depois verificar as previsões antes de treinar um adaptador sobre essas previsões. Consulte Análise em massa, verificação de previsão e treinamento de um adaptador para começar.

Anotação de imagem

Você mesmo pode anotar imagens rotulando imagens com o console do Rekognition ou usar a análise em massa do Rekognition para anotar imagens que você pode verificar se foram rotuladas corretamente. Escolha um dos tópicos abaixo para começar.

Tópicos

- Criar um projeto e treinar um adaptador
- Análise em massa, verificação de previsão e treinamento de um adaptador

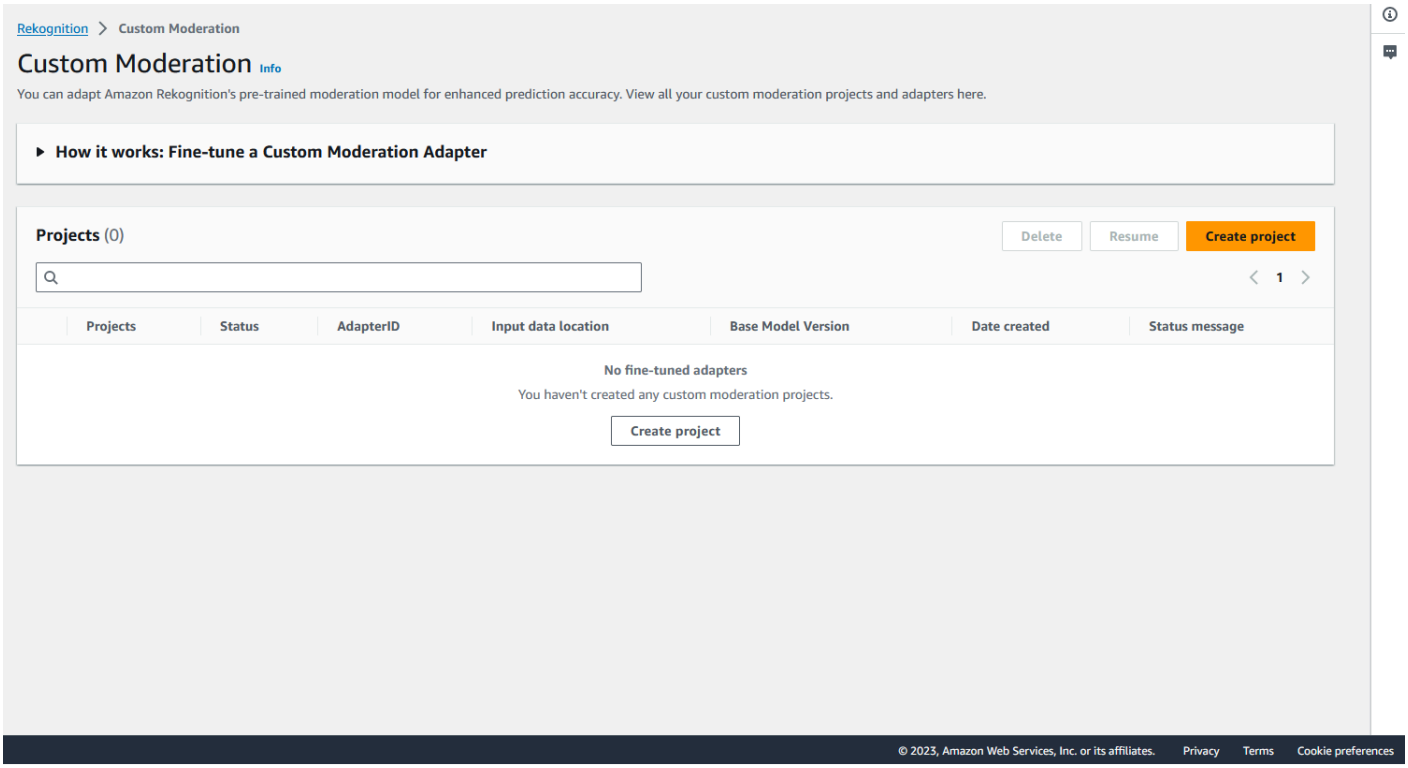
Criar um projeto e treinar um adaptador

Conclua as etapas a seguir para treinar seu adaptador anotando imagens usando o console do Rekognition.

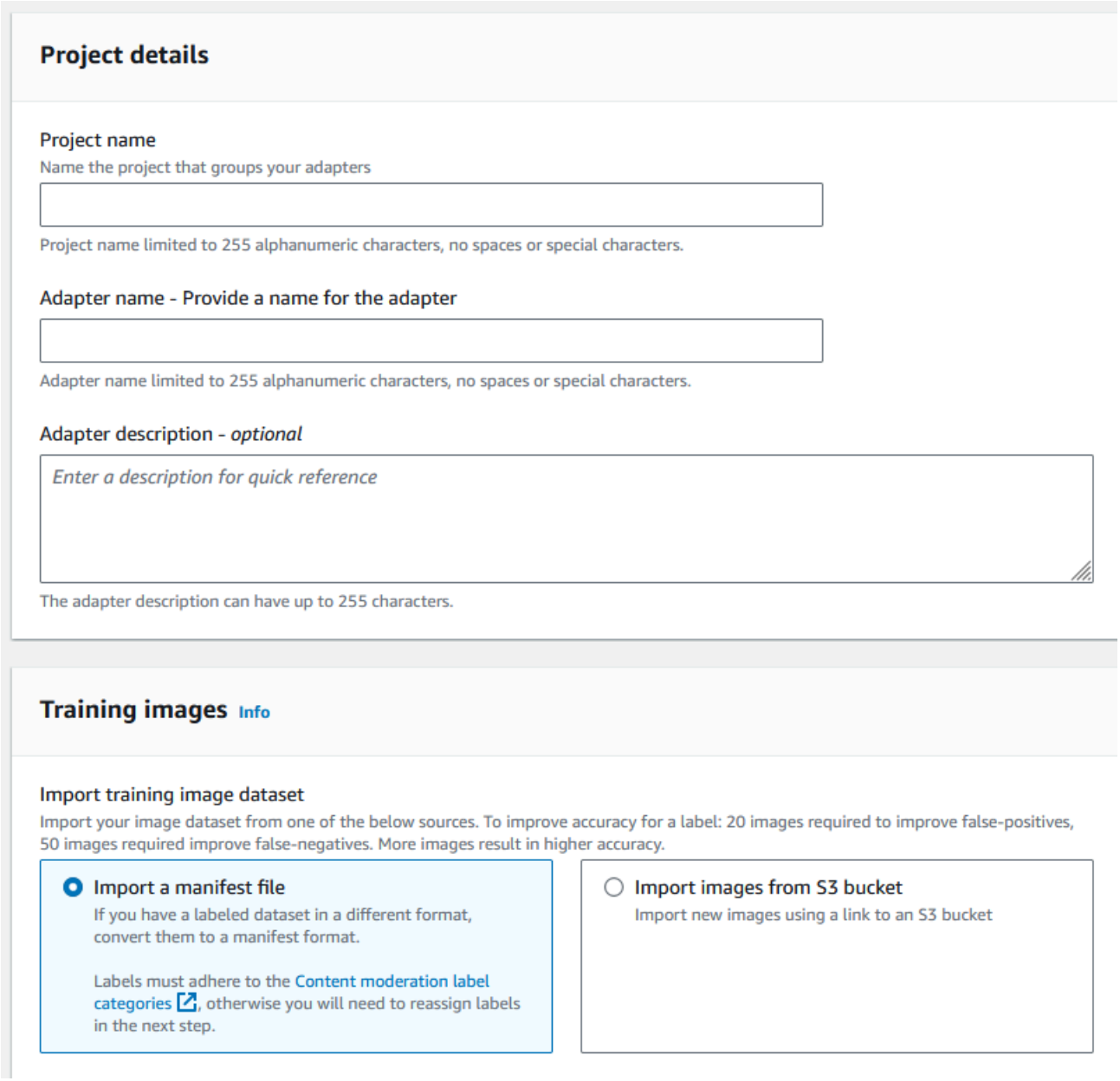
Criar um projeto

Antes de treinar ou usar um adaptador, você deve criar o projeto que o conterá. Você também deve fornecer as imagens usadas para treinar seu adaptador. Para criar um projeto, um adaptador e seus conjuntos de dados de imagem:

1. Faça login no AWS Management Console e abra o console do Rekognition em. https://console.aws.amazon.com/rekognition/
2. No painel esquerdo, escolha Moderação personalizada. A página de destino da Moderação personalizada do Rekognition é exibida.



3. A página inicial da Moderação personalizada mostra uma lista de todos os seus projetos e adaptadores, além de um botão para criar um adaptador. Escolha Criar projeto para criar um novo projeto e adaptador.
4. Se esta for sua primeira vez criando um adaptador, você será solicitado a criar um bucket do Amazon S3 para armazenar arquivos relacionados ao seu projeto e ao seu adaptador. Escolha Criar bucket do Amazon S3.
5. Na página seguinte, preencha o nome do adaptador e o nome do projeto. Forneça uma descrição do adaptador, se desejar.



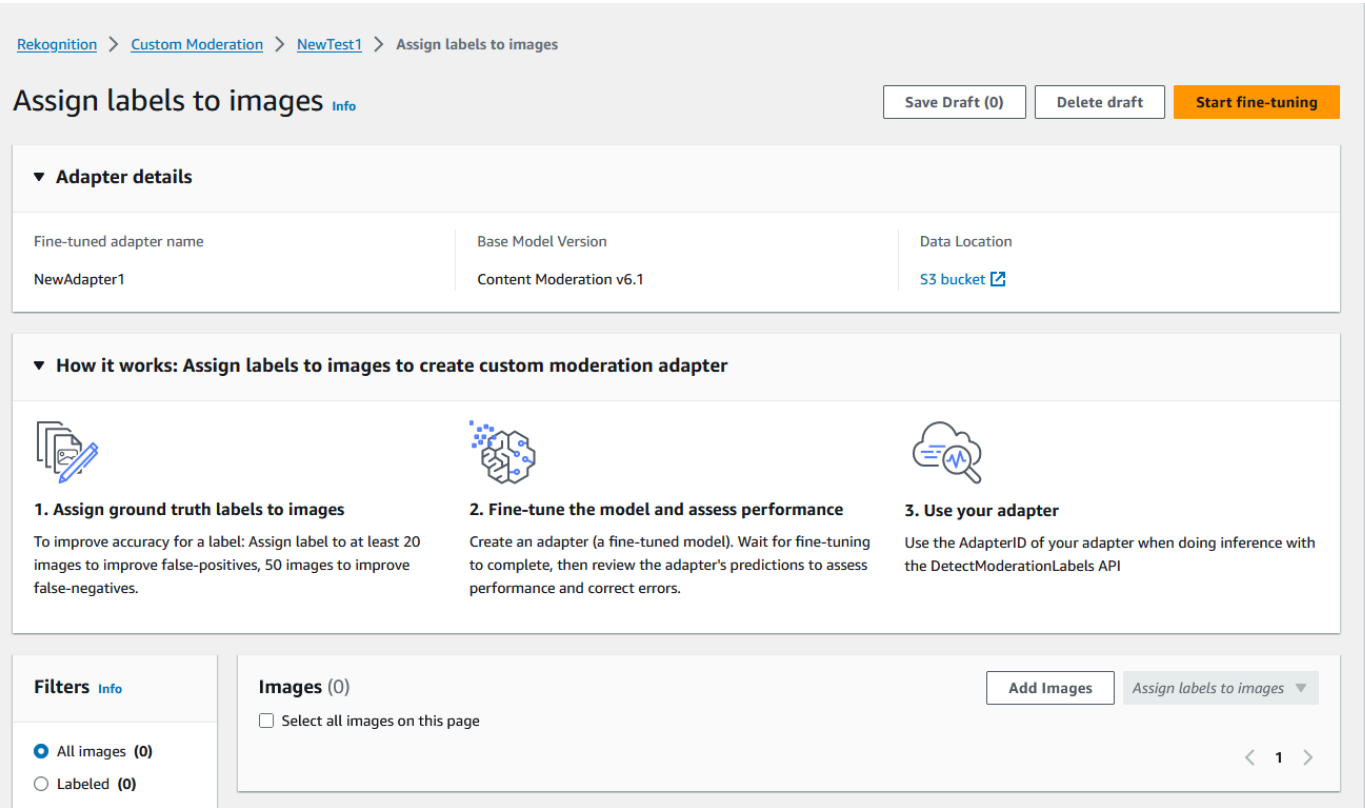
6. Nesta etapa, você também fornecerá as imagens do seu adaptador. Você pode selecionar: Importe imagens do seu computador, importe o arquivo manifesto ou importe imagens do bucket do Amazon S3. Se você optar por importar suas imagens de um bucket do Amazon S3, forneça o caminho para o bucket e a pasta que contém suas imagens de treinamento. Se você enviar suas imagens diretamente do seu computador, observe que só poderá carregar até 30 imagens por vez. Se estiver usando um arquivo de manifesto que contenha anotações, pule as etapas listadas abaixo que abrangem a anotação de imagens e prossiga para a seção sobre Analisando o desempenho do adaptador.

7. Na seção Detalhes do conjunto de dados de teste, escolha Divisão automática para que o Rekognition selecione automaticamente a porcentagem apropriada de suas imagens como dados de teste ou escolha Importar manualmente o arquivo manifesto.
8. Depois de preencher essas informações, selecione Criar projeto.

Treinar um adaptador

Para treinar um adaptador com suas próprias imagens não anotadas:

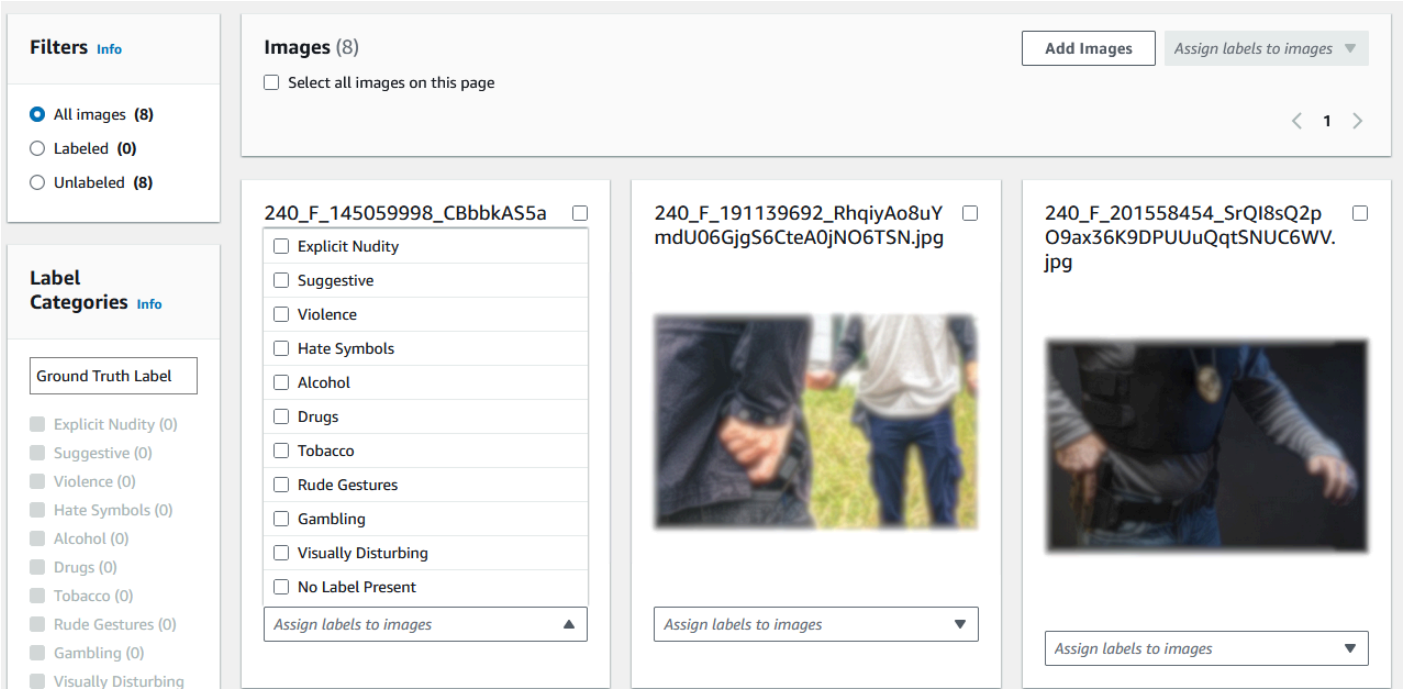
1. Selecione o projeto que contém seu adaptador e, em seguida, escolha a opção Atribuir rótulo às imagens.
2. Na página Atribuir rótulo às imagens, você pode ver todas as imagens que foram carregadas como imagens de treinamento. Você pode filtrar essas imagens por status rotulado/não rotulado e por categoria de rótulo usando os dois painéis de seleção de atributos à esquerda. Você pode adicionar imagens adicionais ao seu conjunto de dados de treinamento selecionando o botão Adicionar imagens .



3. Depois de adicionar imagens ao conjunto de dados de treinamento, você deve anotar suas imagens com rótulos. Depois de enviar suas imagens, a página "Atribuir rótulos às imagens" será atualizada para mostrar as imagens que você enviou. Você será solicitado a selecionar o rótulo

apropriado para suas imagens em uma lista suspensa de rótulos compatíveis com o Rekognition Moderation. Você pode selecionar mais de um rótulo.

4. Continue esse processo até adicionar rótulos a cada uma das imagens em seus dados de treinamento.
5. Depois de rotular todos os seus dados, selecione Iniciar treinamento para começar a treinar o modelo, que cria seu adaptador.



6. Antes de iniciar o processo de treinamento, você pode adicionar quaisquer Tags ao adaptador que desejar. Você também pode fornecer ao adaptador uma chave de criptografia personalizada ou usar uma chave AWS KMS. Depois de terminar de adicionar as tags desejadas e personalizar a criptografia ao seu gosto, selecione Treinar adaptador para iniciar o processo de treinamento do seu adaptador.
7. Aguarde até que seu adaptador termine o treinamento. Depois que o treinamento for concluído, você receberá uma notificação de que seu adaptador terminou de ser criado.

Quando o status do seu adaptador for "Treinamento concluído", você poderá revisar as métricas do seu adaptador

Análise em massa, verificação de previsão e treinamento de um adaptador

Conclua as etapas a seguir para treinar seu adaptador verificando as previsões de análise em massa do modelo de moderação de conteúdo do Rekognition.

Para treinar um adaptador verificando as previsões do modelo de moderação de conteúdo do Rekognition, você deve:

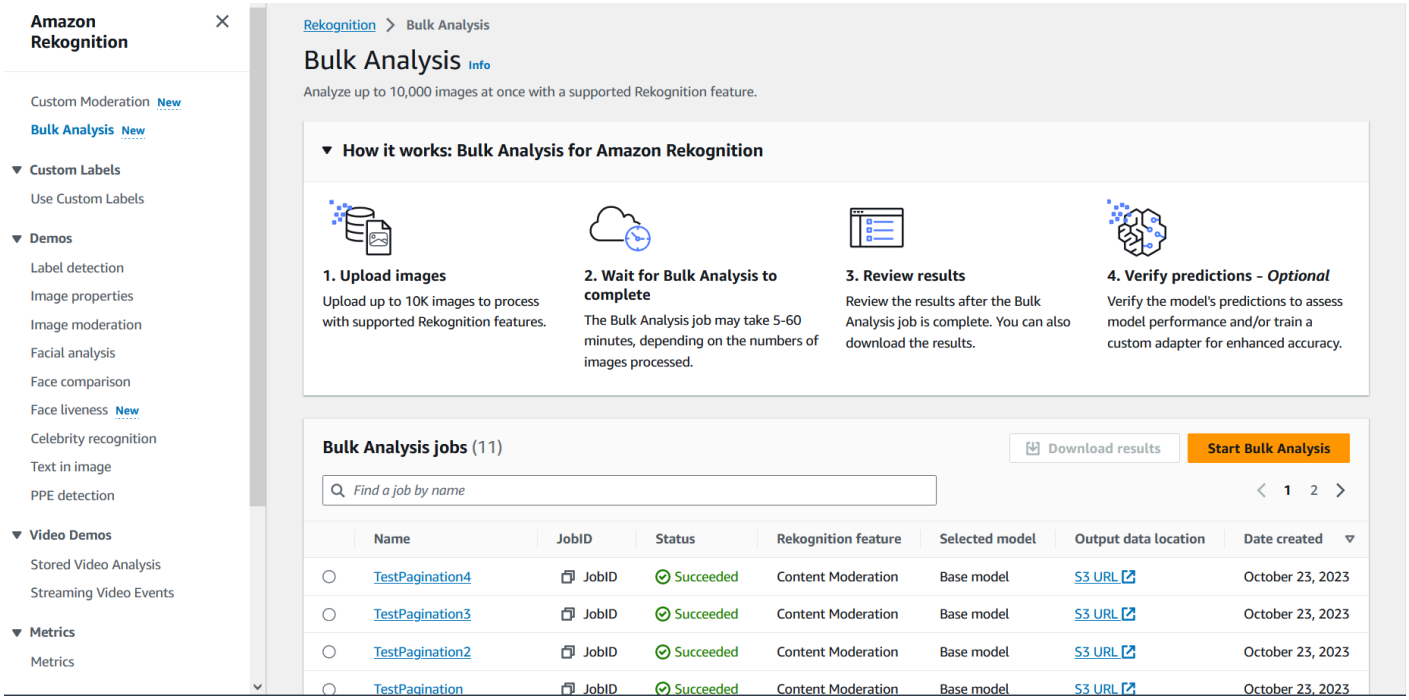
1. Realize análises em massa em suas imagens
2. Verifique as previsões retornadas para suas imagens

Você pode obter previsões para imagens realizando análises em massa com o modelo básico do Rekognition ou com um adaptador que você já criou.

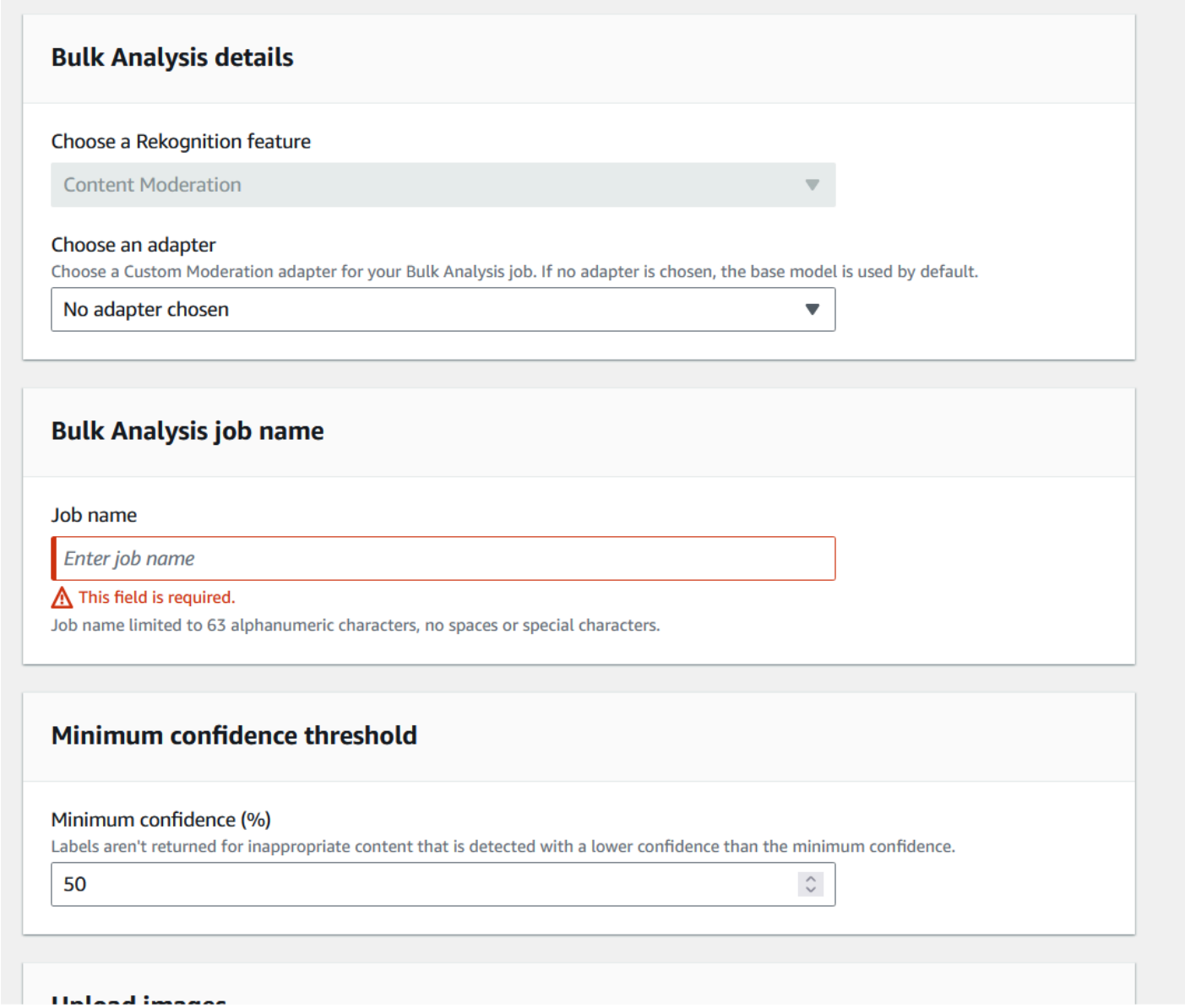
Execute análises em massa em suas imagens

Para treinar um adaptador com base nas previsões que você verificou, primeiro você deve iniciar um trabalho de análise em massa para analisar um lote de imagens usando o modelo básico do Rekognition ou um adaptador de sua escolha. Para executar um trabalho de análise em massa:

1. Faça login no AWS Management Console e abra o console do Amazon Rekognition em. https://console.aws.amazon.com/rekognition/
2. No painel esquerdo, escolha Análise em massa. A página inicial da análise em massa é exibida. Escolha Iniciar análise em massa. A visão geral do atributo de análise em massa mostra as etapas para fazer upload de imagens, aguardar a análise, revisar os resultados e, opcionalmente, verificar as predições de modelo. Lista trabalhos recentes de análise em massa para moderação de conteúdo usando o modelo básico.



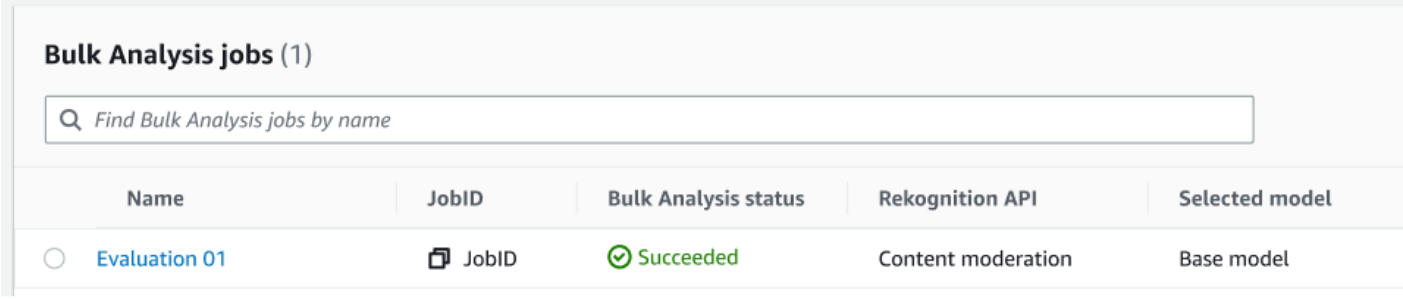
3. Se esta for sua primeira vez criando um adaptador, você será solicitado a criar um bucket do Amazon Simple Storage Service para armazenar arquivos relacionados ao seu projeto e ao seu adaptador. Escolha Criar bucket do Amazon S3.
4. Selecione o adaptador que você deseja usar para a análise em massa usando o menu suspenso Escolher um adaptador . Se nenhum adaptador for escolhido, o modelo básico será usado por padrão. Para os fins deste tutorial, não escolha um adaptador.



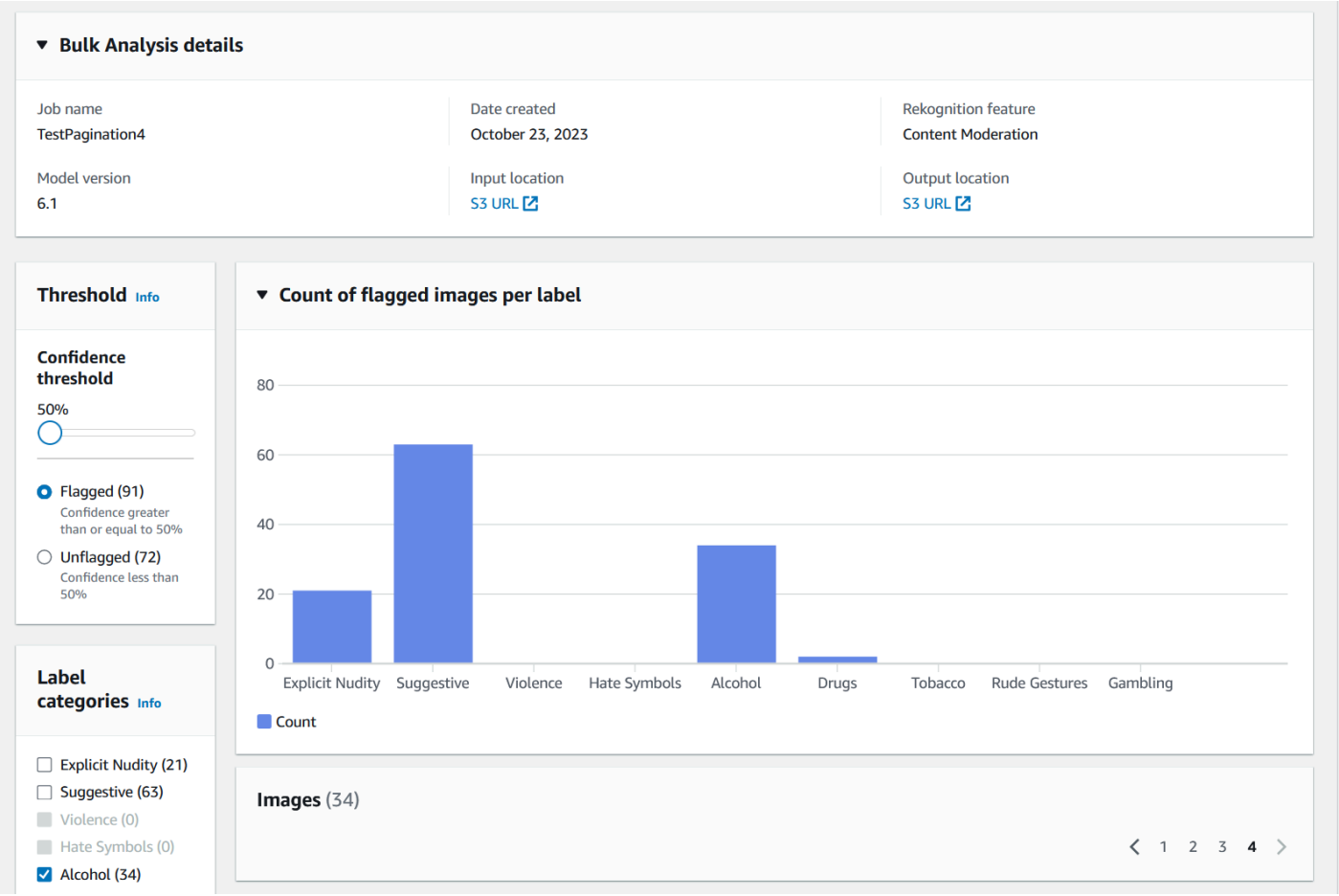
5. No campo Nome do trabalho de análise em massa, preencha o nome do trabalho de análise em massa.
6. Escolha um valor para o Limite mínimo de confiança. As previsões de rótulos com menos do que o limite de confiança escolhido não serão retornadas. Observe que, ao avaliar o desempenho do

modelo posteriormente, você não poderá ajustar o limite de confiança abaixo do limite mínimo de confiança escolhido.

7. Nesta etapa, você também fornecerá as imagens que deseja analisar com a análise em massa. Essas imagens também podem ser usadas para treinar seu adaptador. Você pode escolher Carregar imagens do seu computador ou Importar imagens do bucket Amazon S3. Se você optar por importar seus documentos de um bucket do Amazon S3, forneça o caminho para o bucket e a pasta que contém suas imagens de treinamento. Se você enviar seus documentos diretamente do seu computador, observe que só poderá fazer upload de 50 imagens por vez.
8. Depois de preencher essas informações, escolha Iniciar análise. Isso iniciará o processo de análise usando o modelo básico do Rekognition.
9. Você pode verificar o status de seu trabalho de análise em massa verificando o status de análise em massa do trabalho na página principal de análise em massa. Quando o status da análise em massa se torna "Bem-sucedida", os resultados da análise estão prontos para análise.



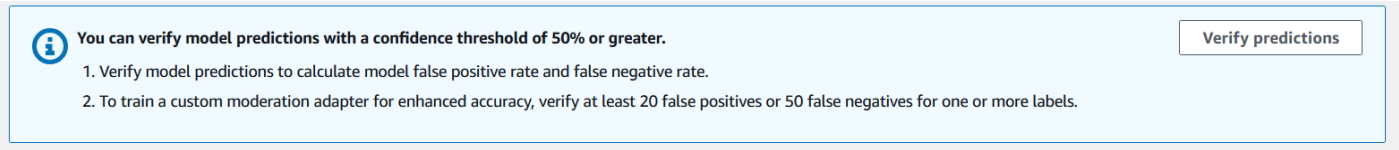
10. Escolha a análise que você criou na lista de Trabalhos de análise em massa.
11. Na página de detalhes da análise em massa, você pode ver as previsões que o modelo básico do Rekognition fez para as imagens que você enviou.
12. Analise o desempenho do modelo básico. Você pode alterar o limite de confiança que seu adaptador deve ter para atribuir um rótulo a uma imagem usando o controle deslizante de limite de confiança. O número de instâncias sinalizadas e não sinalizadas mudará à medida que você ajusta o limite de confiança. O painel Categorias de rótulos exibe as categorias de nível superior que o Rekognition reconhece, e você pode selecionar uma categoria nessa lista para exibir qualquer imagem que tenha sido atribuída a esse rótulo.



Verificar as previsões

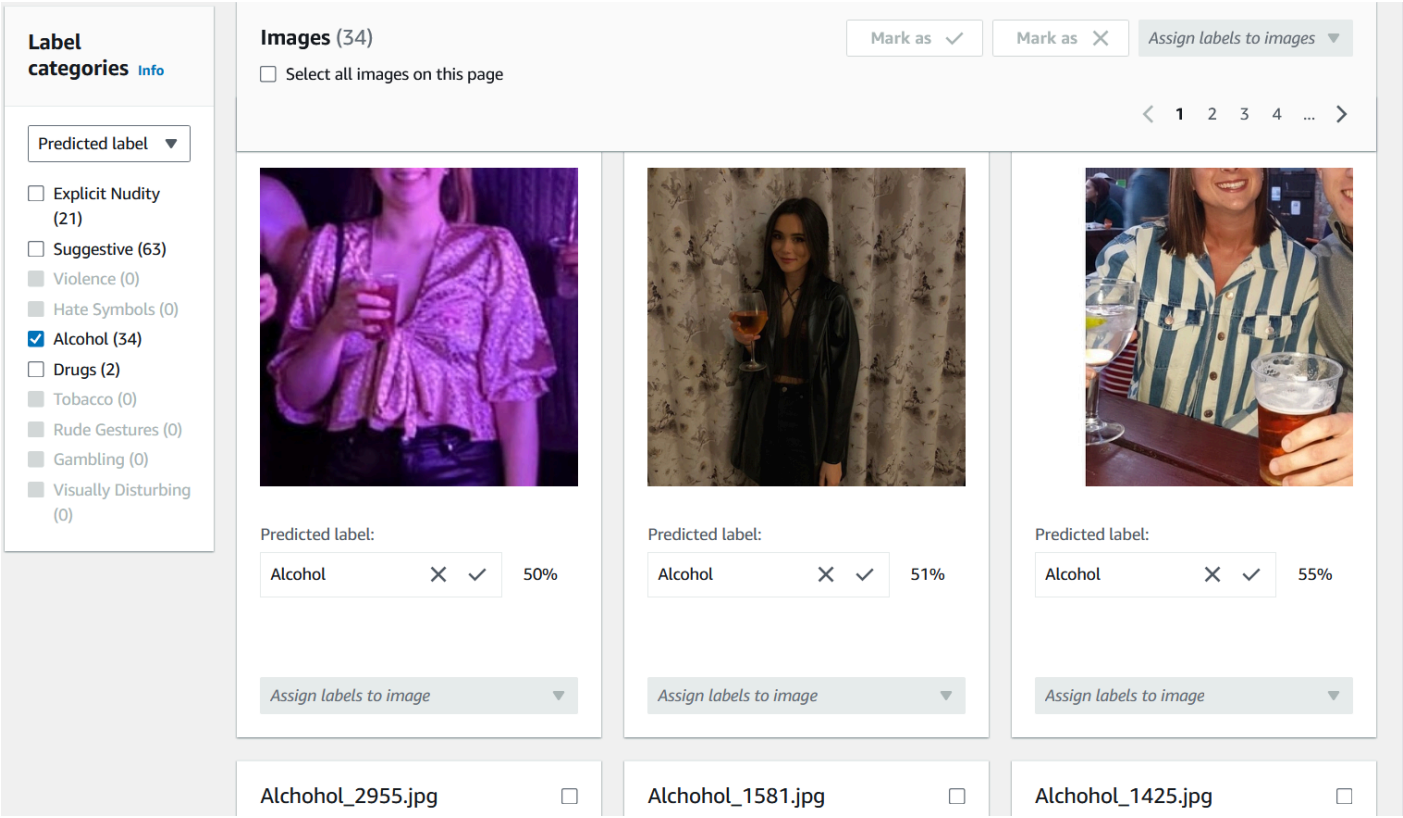
Se você analisou a precisão do modelo básico do Rekognition ou de um adaptador escolhido e deseja melhorar essa precisão, você pode utilizar o fluxo de trabalho de verificação:

1. Depois de terminar de revisar o desempenho do modelo básico, você desejará verificar as previsões. A correção das previsões permitirá que você treine um adaptador. Escolha Verificar previsões na parte superior da página de análise em massa.

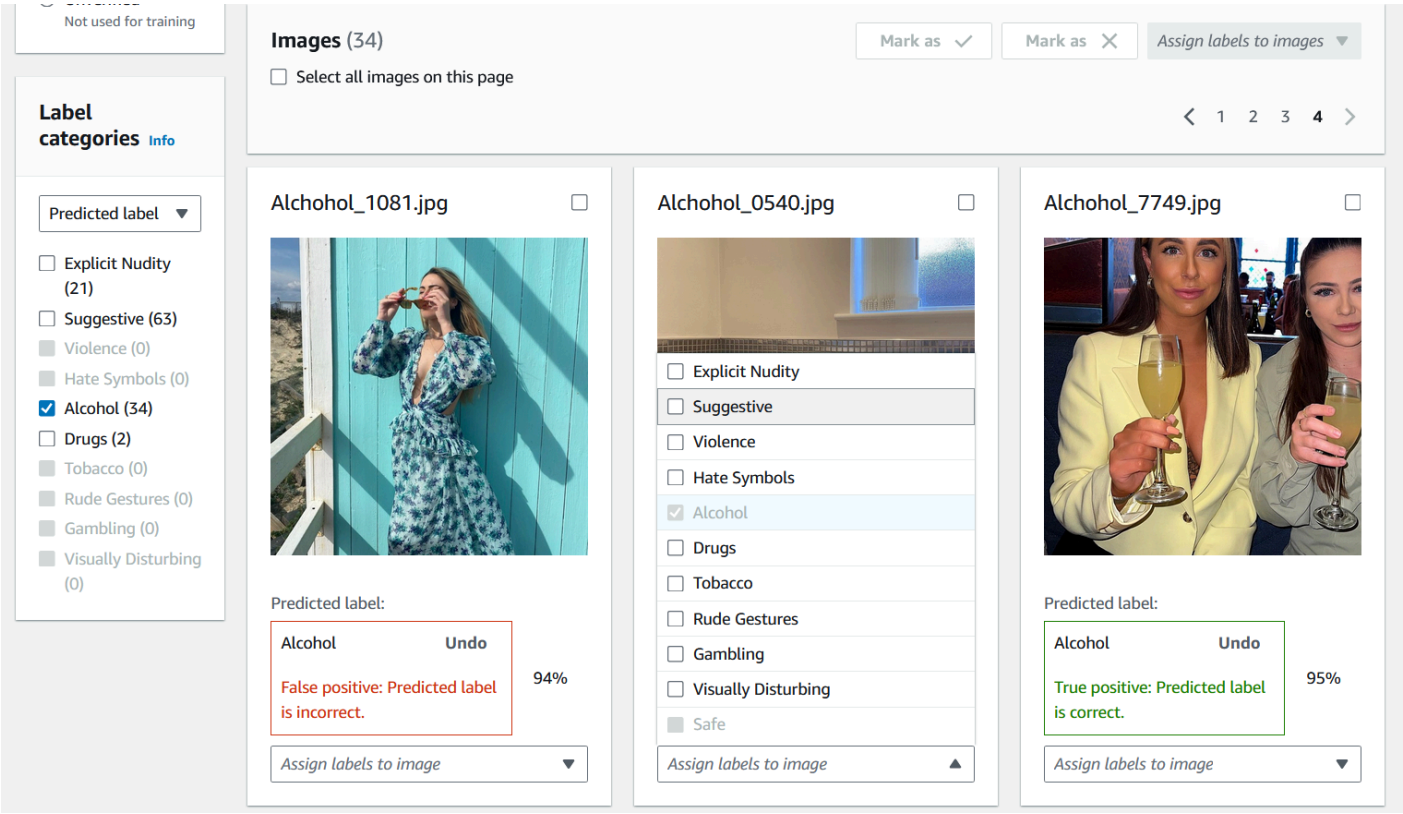


2. Na página Verificar previsões, você pode ver todas as imagens que forneceu ao modelo básico do Rekognition, ou a um adaptador escolhido, junto com o rótulo previsto para cada imagem. Você deve verificar se cada previsão está correta ou incorreta usando os botões abaixo da imagem. Use o botão "X" para marcar uma previsão como incorreta e o botão de marca de seleção para

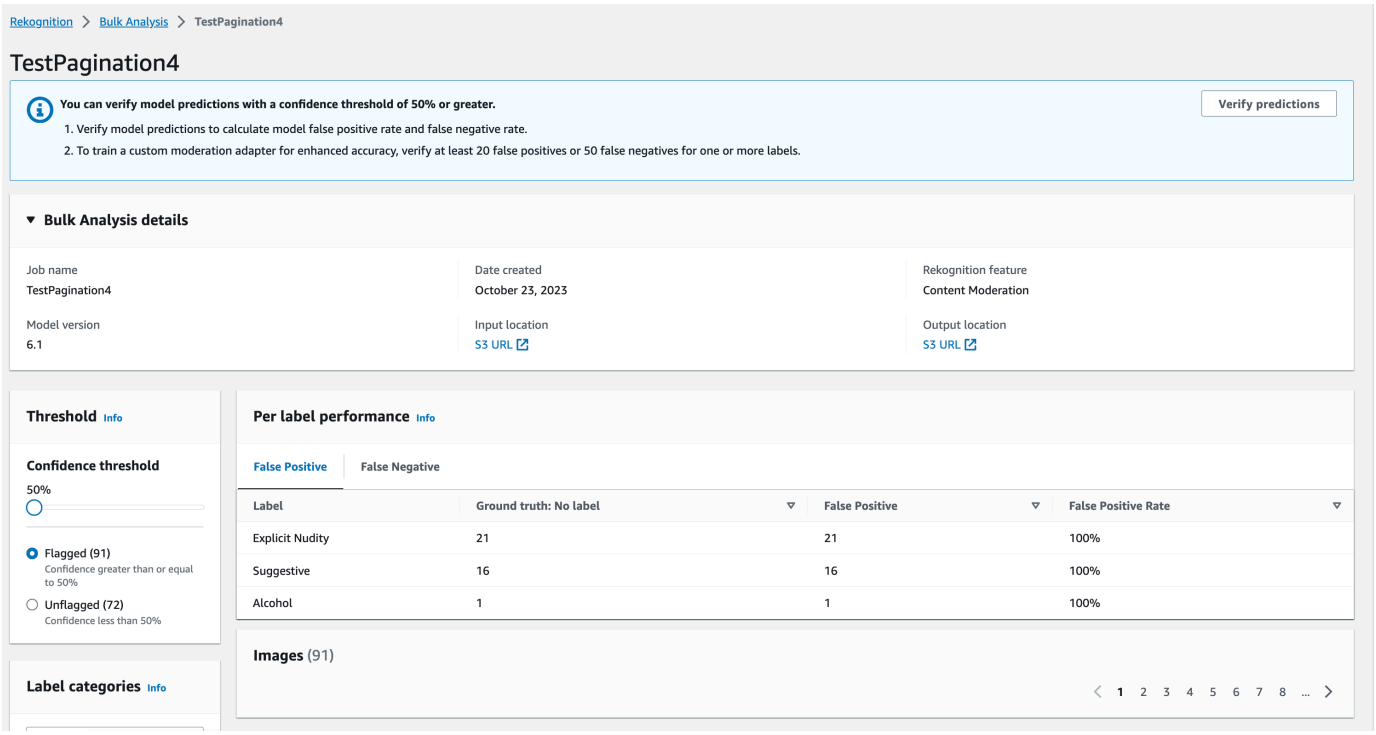
marcar uma previsão como correta. Para treinar um adaptador, você precisará verificar pelo menos 20 previsões falso-positivas e 50 predições falso-negativas para um determinado rótulo. Quanto mais previsões você verificar, melhor será o desempenho do adaptador.



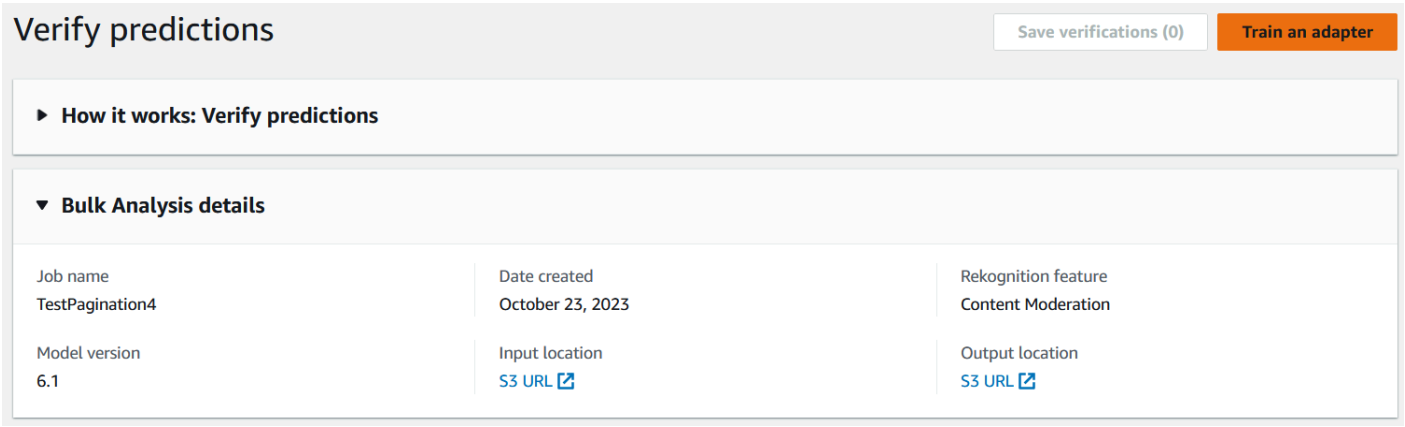
Depois de verificar uma previsão, o texto abaixo da imagem será alterado para mostrar o tipo de previsão que você verificou. Depois de verificar uma imagem, você também pode adicionar rótulos adicionais à imagem usando o menu Atribuir rótulos à imagem . Você pode ver quais imagens estão marcadas ou não marcadas pelo modelo de acordo com o limite de confiança escolhido ou filtrar imagens por categoria.



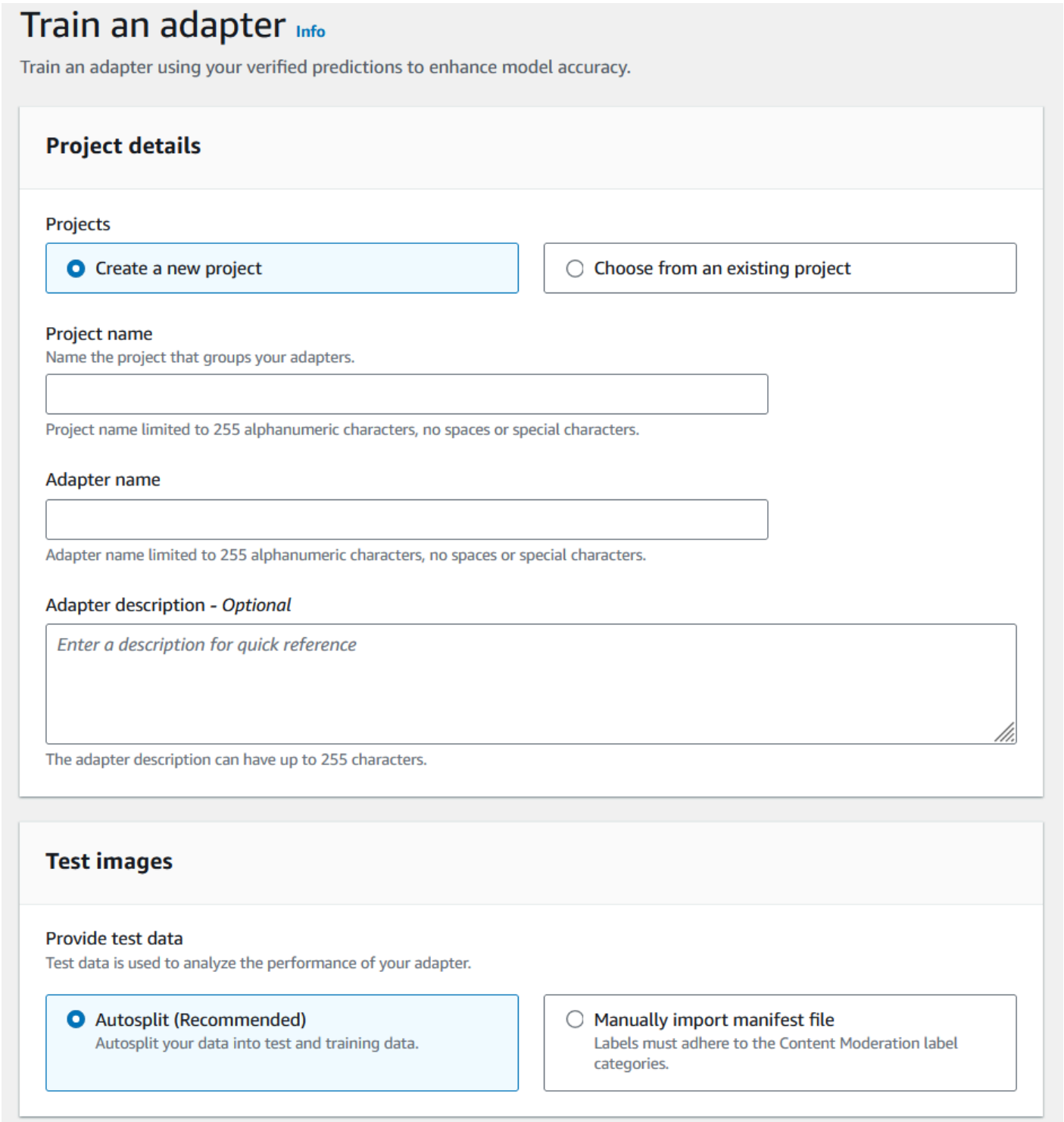
3. Depois de concluir a verificação de todas as previsões que deseja verificar, você poderá ver estatísticas sobre suas previsões verificadas na seção Desempenho por rótulo da página Verificação. Você também pode retornar à página de detalhes da análise em massa para ver essas estatísticas.



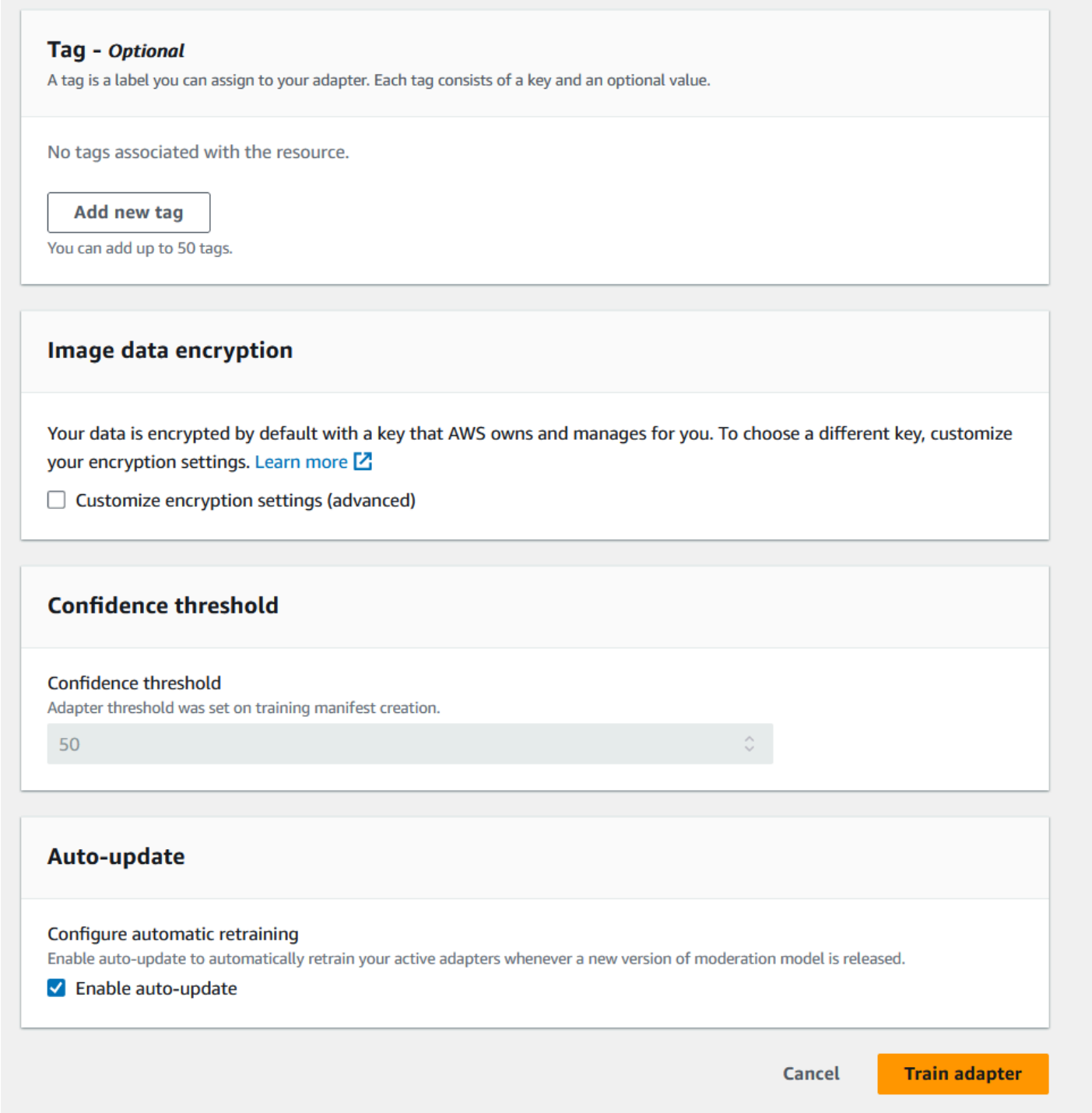
4. Quando estiver satisfeito com as estatísticas sobre o Desempenho por rótulo, acesse a página Verificar previsões novamente e selecione o botão Treinar um adaptador para começar a treinar o adaptador.



5. Na página Treinar um adaptador, você será solicitado a criar um projeto ou escolher um projeto existente. Nomeie o projeto e o adaptador que serão contidos no projeto. Você também deve especificar a origem das imagens de teste. Ao especificar as imagens, você pode escolher a divisão automática para que o Rekognition use automaticamente uma parte dos seus dados de treinamento como imagens de teste, ou você pode especificar manualmente um arquivo de manifesto. É recomendável escolher Divisão Automática.



6. Especifique as tags desejadas, bem como uma AWS KMS chave, se não quiser usar a AWS chave padrão. É recomendável deixar a Atualização automática ativada.
7. Escolha Treinar adaptador.

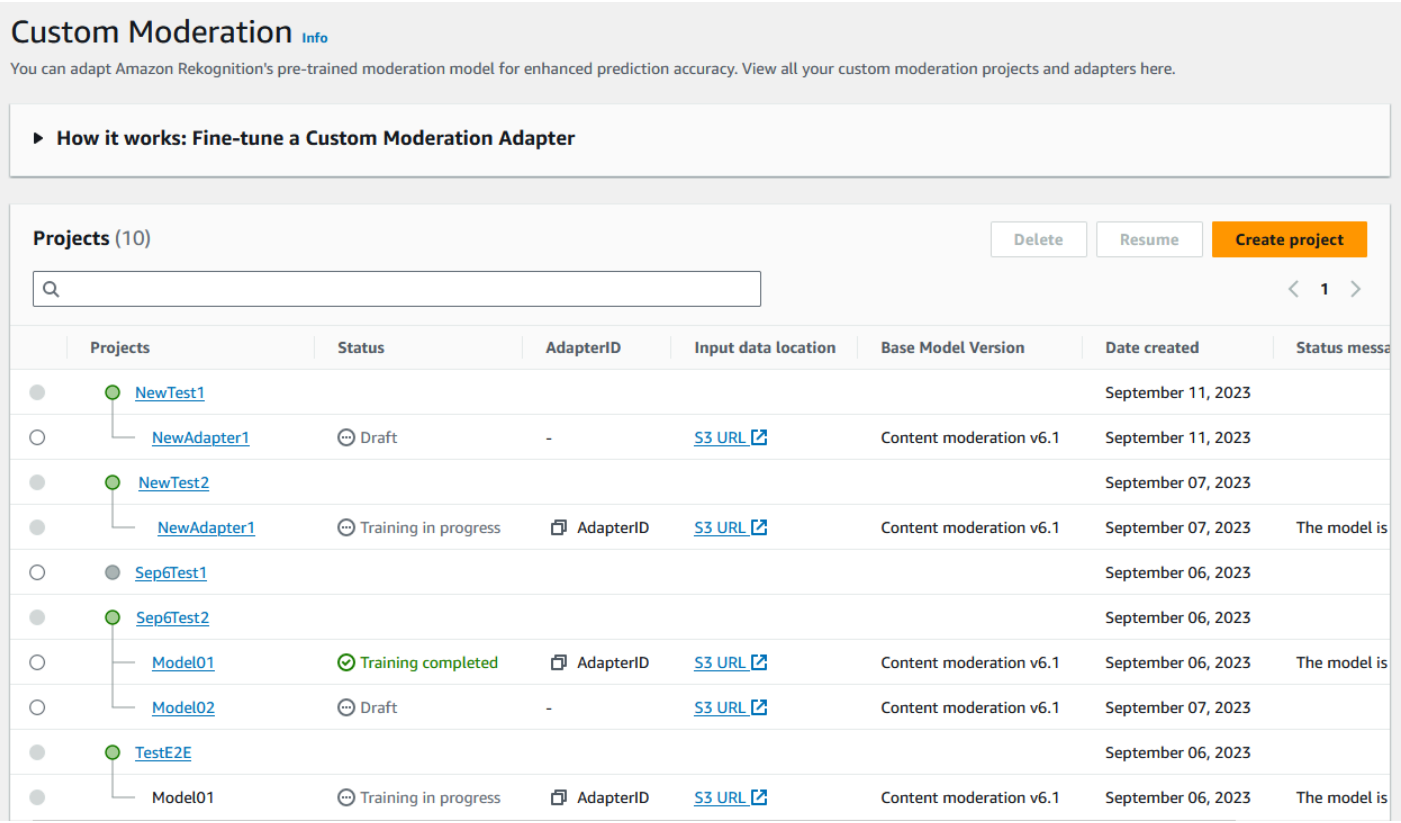


8. Quando o status do seu adaptador na página inicial da Moderação personalizada se tornar "Treinamento concluído", você poderá revisar o desempenho do adaptador. Consulte Analisando o desempenho do adaptador para obter mais informações.

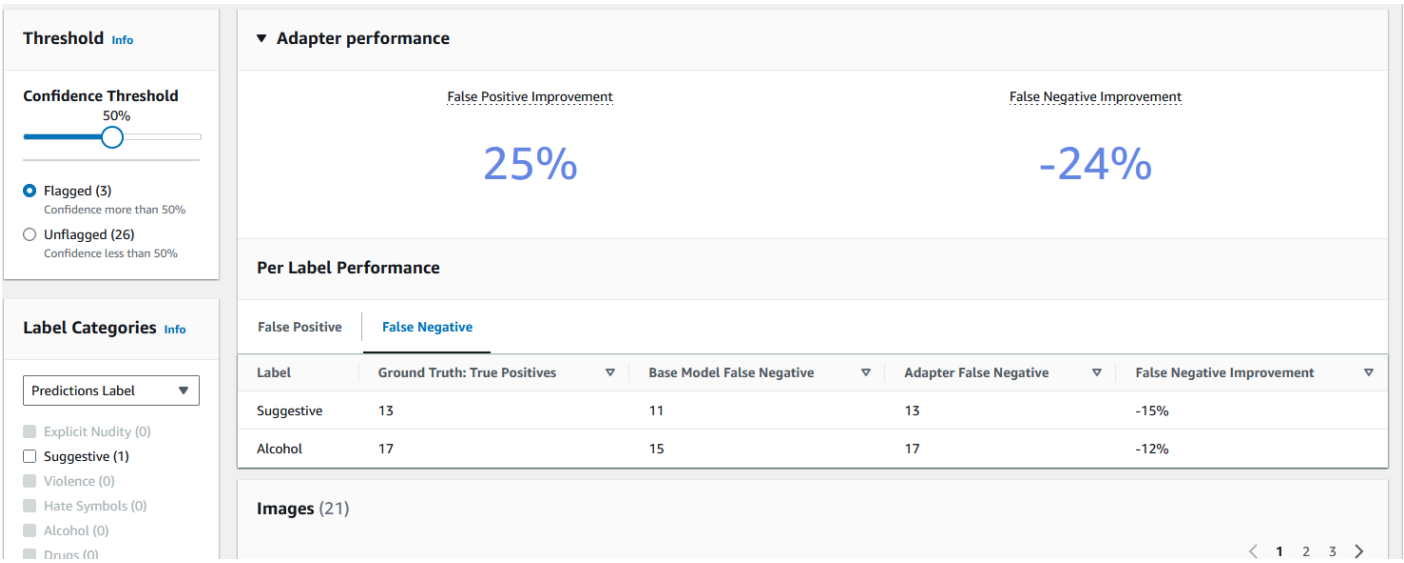
Analisando o desempenho do adaptador

Para analisar o desempenho do adaptador:

1. Ao usar o console, você poderá ver o status de qualquer adaptador associado a um projeto na guia Projetos na página inicial da Moderação personalizada. Navegue até a página inicial da Moderação personalizada.



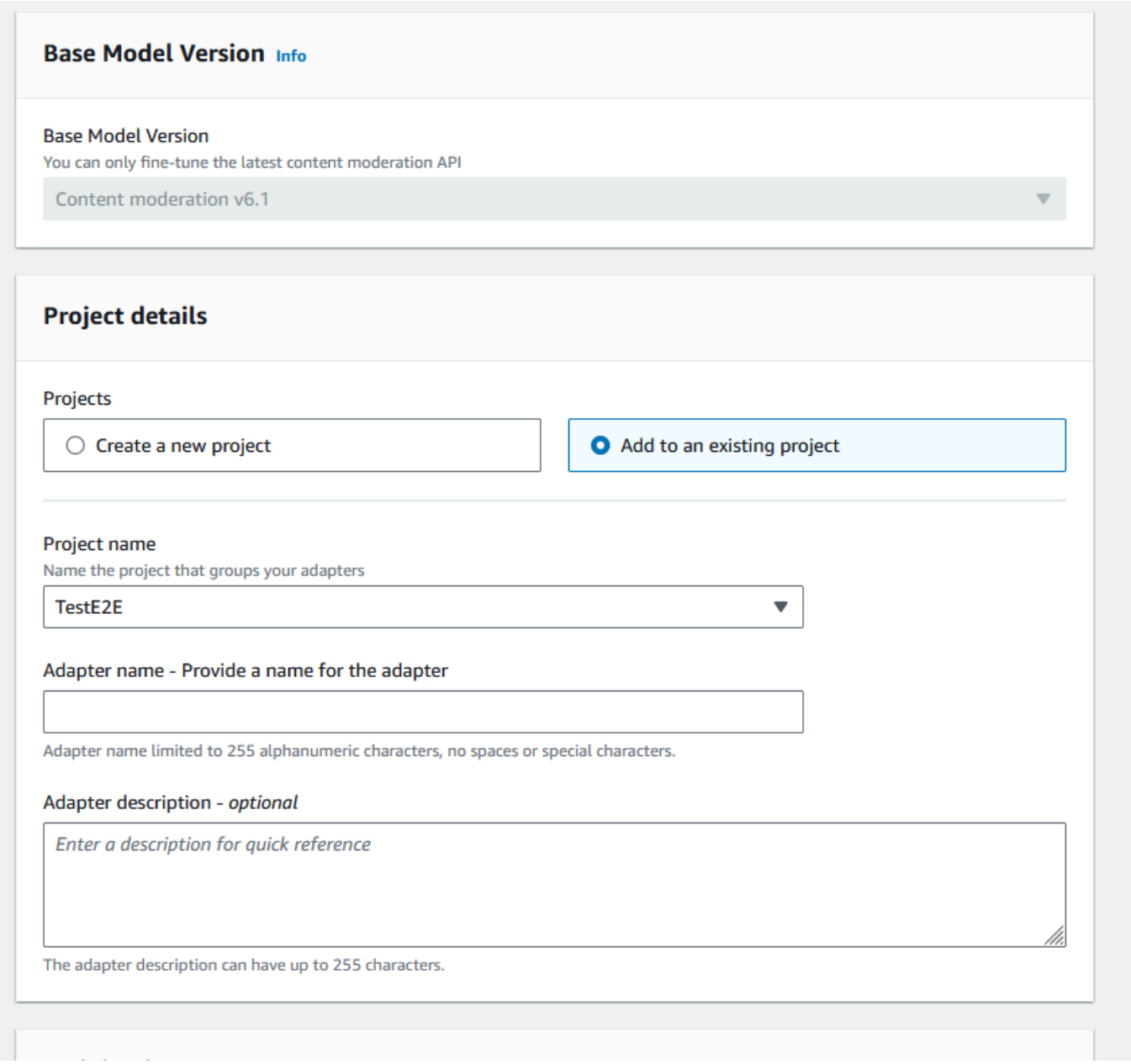
2. Selecione o adaptador que você deseja revisar nesta lista. Na página de detalhes do adaptador a seguir, você pode ver uma variedade de métricas do adaptador.



3. Com o painel Limite, você pode alterar o limite mínimo de confiança que seu adaptador deve ter para atribuir uma etiqueta a uma imagem. O número de instâncias sinalizadas e não sinalizadas mudará à medida que você ajusta o limite de confiança. Você também pode filtrar por categoria de rótulo para ver as métricas das categorias que você selecionou. Defina o limite escolhido.
4. Você pode avaliar o desempenho do adaptador nos dados de teste examinando as métricas no painel Desempenho do adaptador. Essas métricas são calculadas comparando as extrações do adaptador com as anotações de "verdade fundamental" no conjunto de teste.

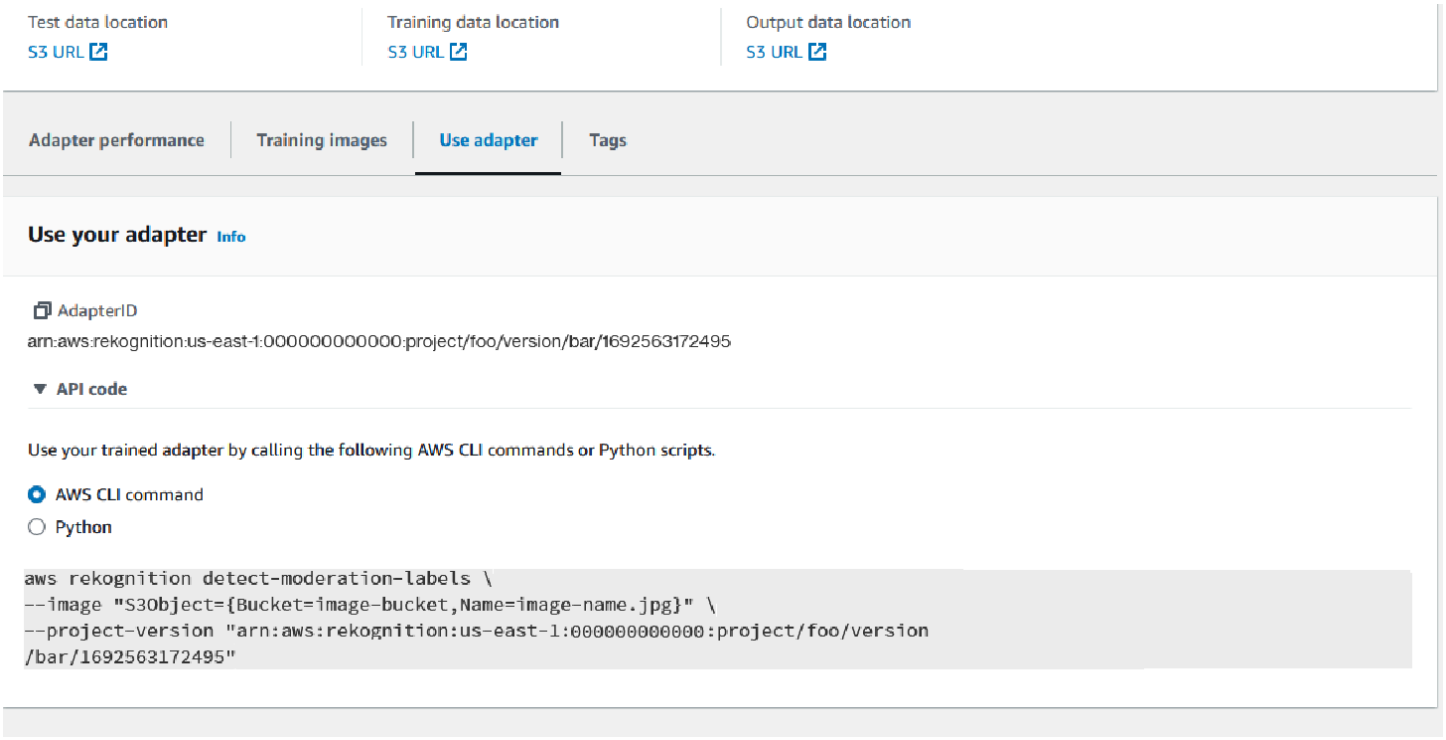
O painel de desempenho do adaptador mostra as taxas de melhoria de falsos positivos e de melhoria de falsos negativos do adaptador que você criou. A guia Desempenho por etiqueta pode ser usada para comparar o desempenho do adaptador e do modelo básico em cada categoria de etiqueta. Ele mostra contagens de previsões de falsos positivos e falsos negativos pelo modelo básico e pelo adaptador, estratificadas por categoria de rótulo. Ao analisar essas métricas, você pode determinar onde o adaptador precisa ser aprimorado. Para ter mais informações sobre essas métricas, consulte Avaliar e melhorar o adaptador.

Para melhorar o desempenho, você pode coletar mais imagens de treinamento e criar um novo adaptador baseado dentro do projeto. Basta retornar à página inicial da Moderação personalizada e criar um novo adaptador dentro do seu projeto, fornecendo mais imagens de treinamento para o adaptador a ser treinado. Desta vez, escolha a opção Adicionar a um projeto existente em vez de Criar um novo projeto e selecione o projeto no qual você deseja criar o novo adaptador no menu suspenso Nome do projeto. Como antes, anote suas imagens ou forneça um arquivo de manifesto com anotações.



Usar o adaptador

Depois de criar seu adaptador, você pode fornecê-lo a uma operação de Rekognition compatível, como. DetectModerationLabels Para ver exemplos de código que você pode usar para realizar inferências com seu adaptador, selecione a guia “Usar adaptador”, onde você pode ver exemplos de código para a AWS CLI e o Python. Você também pode visitar a seção respectiva da documentação da operação para a qual você criou um adaptador para ver mais exemplos de código, instruções de configuração e um exemplo de JSON.



Excluir adaptador e projeto

Você pode excluir adaptadores ou projeto individualmente. Você deve excluir cada adaptador associado ao seu projeto antes de excluir o projeto em si.

1. Para excluir um adaptador associado ao projeto, escolha o adaptador e escolha Excluir.
2. Para excluir um projeto, escolha o projeto que você deseja excluir e escolha Excluir.

Avaliar e melhorar o adaptador

Depois de cada rodada de treinamento do adaptador, você deve analisar as métricas de desempenho na ferramenta console do Rekognition para determinar o quão próximo o adaptador está do nível de desempenho desejado. Em seguida, você pode melhorar ainda mais a precisão de suas imagens do adaptador fazendo o upload de um novo lote de imagens de treinamento e treinando um novo adaptador dentro do seu projeto. Depois de criar uma versão aprimorada do adaptador, você pode usar o console para excluir qualquer versão antiga do adaptador que não seja mais necessária.

Você também pode recuperar métricas usando a operação da DescribeProjectVersionsAPI.

Métricas de performance

Depois de concluir o processo de treinamento e criar seu adaptador, é importante avaliar o quão bem o adaptador está extraindo informações de suas imagens.

Duas métricas são fornecidas no Console do Rekognition para ajudá-lo a analisar o desempenho do seu adaptador: melhoria de falsos positivos e melhoria de falsos negativos.

Você pode ver essas métricas para qualquer adaptador selecionando a guia "Desempenho do adaptador" na parte do adaptador do console. O painel de desempenho do adaptador mostra as taxas de melhoria de falsos positivos e de melhoria de falsos negativos do adaptador que você criou.

A melhoria de falsos positivos mede o quanto o reconhecimento de falsos positivos pelo adaptador melhorou em relação ao modelo básico. Se o valor de melhoria de falsos positivos for 25%, isso significa que o adaptador melhorou seu reconhecimento de falsos positivos em 25% no conjunto de dados de teste.

A melhoria de falsos negativos mede o quanto o reconhecimento de falsos negativos pelo adaptador melhorou em relação ao modelo básico. Se o valor de melhoria de falsos negativos for 25%, isso significa que o adaptador melhorou seu reconhecimento de falsos negativos em 25% no conjunto de dados de teste.

A guia Desempenho por etiqueta pode ser usada para comparar o desempenho do adaptador e do modelo básico em cada categoria de etiqueta. Ele mostra contagens de previsões de falsos positivos e falsos negativos pelo modelo básico e pelo adaptador, estratificadas por categoria de rótulo. Ao analisar essas métricas, você pode determinar onde o adaptador precisa ser aprimorado.

Por exemplo, se a taxa de falsos negativos do modelo básico para a categoria de rótulo de álcool for 15, enquanto a taxa de falsos negativos do adaptador for 15 ou superior, você sabe que deve se concentrar em adicionar mais imagens contendo o rótulo de álcool ao criar um novo adaptador.

Ao usar as operações da API Rekognition, a métrica F1-Score é retornada ao chamar a operação. DescribeProjectVersions

Melhorar o modelo

A implantação do adaptador é um processo iterativo, pois você provavelmente precisará treinar um adaptador várias vezes para atingir o nível de precisão desejado. Depois de criar e treinar seu adaptador, você deve testar e avaliar o desempenho do adaptador em vários tipos de etiquetas.

Se a precisão do adaptador estiver insuficiente em alguma área, adicione novos exemplos dessas imagens para aumentar o desempenho do adaptador para essas etiquetas. Tente fornecer ao adaptador exemplos adicionais e variados que reflitam os casos em que ele tem dificuldades. Fornecer ao seu adaptador imagens representativas e variadas permite que ele manipule diversos exemplos do mundo real.

Depois de adicionar novas imagens ao seu conjunto de treinamento, treine novamente o adaptador e reavalie-o no conjunto de teste e nas etiquetas. Repita esse processo até que o adaptador atinja o nível de desempenho desejado. Se você fornecer imagens e anotações mais representativas, as pontuações de falso positivo e falso negativo melhorarão gradualmente ao longo das sucessivas iterações de treinamento.

Formatos de arquivo manifesto

As seções a seguir mostram exemplos dos formatos de arquivo de manifesto para arquivos de entrada, saída e avaliação.

Manifesto de entrada

Um arquivo manifesto é um arquivo delimitado por linha json, com cada linha contendo um JSON que contém informações sobre uma única imagem.

Cada entrada no manifesto de entrada deve conter o campo `source-ref` com um caminho para a imagem no bucket do Amazon S3 e, para moderação personalizada, o campo `content-moderation-groundtruth` com anotações básicas. Espera-se que todas as imagens em um conjunto de dados estejam no mesmo bucket. A estrutura é comum aos arquivos de manifesto de treinamento e teste.

A operação `CreateProjectVersion` de moderação personalizada usa as informações fornecidas no manifesto de entrada para treinar um adaptador.

O exemplo a seguir é uma linha de um arquivo de manifesto para uma única imagem que contém uma única classe não segura:

```
{
   "source-ref": "s3://foo/bar/1.jpg",
   "content-moderation-groundtruth": {
        "ModerationLabels": [
            {
                "Name": "Rude Gesture"
            }
```

```
        ]
    }
}
```

O exemplo a seguir é uma linha de um arquivo de manifesto para uma única imagem insegura que contém várias classes inseguras, especificamente Nudez e Gesto Rude.

```
{
    "source-ref": "s3://foo/bar/1.jpg",
    "content-moderation-groundtruth": {
         "ModerationLabels": [
             {
                 "Name": "Rude Gesture"
             },
             {
                 "Name": "Nudity"
             }
         ]
    }
}
```

O exemplo a seguir é uma linha de um arquivo de manifesto para uma única imagem que não contém nenhuma classe insegura:

```
{
    "source-ref": "s3://foo/bar/1.jpg",
    "content-moderation-groundtruth": {
         "ModerationLabels": []
    }
}
```

Para ver a lista completa de rótulos compatíveis, consulte Moderação de conteúdo.

Manifesto de saída

Ao concluir um trabalho de treinamento, um arquivo de manifesto de saída é retornado. O arquivo de manifesto de saída é um arquivo delimitado por linha JSON com cada linha contendo um JSON que contém informações para uma única imagem. O caminho do Amazon S3 para o OutputManifest pode ser obtido a partir da `DescribeProjectVersion` resposta:

- `TrainingDataResult.Output.Assets[0].GroundTruthManifest.S3Object` para conjunto de dados de treinamento
- `TestingDataResult.Output.Assets[0].GroundTruthManifest.S3Object` para testar o conjunto de dados

As informações a seguir são retornadas para cada entrada no manifesto de saída:

Nome da chave	Descrição
`source-ref`	Referência a uma imagem em s3 que foi fornecida no manifesto de entrada
`content-moderation-groundtruth`	Anotações verídicas básicas que foram fornecidas no manifesto de entrada
`detect-moderation-labels`	Previsões do adaptador, apenas parte do conjunto de dados de teste
`detect-moderation-labels-base-model`	Previsões do modelo básico, apenas parte do conjunto de dados de teste

As previsões do adaptador e do modelo básico são retornadas em ConfidenceTrehsold 5.0 no formato semelhante à DetectModerationLabelsresposta.

O exemplo a seguir mostra a estrutura das previsões do modelo Adaptador e Base:

```
{
   "ModerationLabels": [
      {
         "Confidence": number,
         "Name": "string",
         "ParentName": "string"
      }
   ],
   "ModerationModelVersion": "string",
   "ProjectVersion": "string"
}
```

Para ver a lista completa de rótulos devolvidos, consulte Moderação de conteúdo.

Manifesto dos resultados da avaliação

Ao concluir um trabalho de treinamento, um arquivo de manifesto do resultado da avaliação é retornado. O manifesto dos resultados da avaliação é um arquivo JSON gerado pelo trabalho de treinamento e contém informações sobre o desempenho do adaptador nos dados de teste.

O caminho do Amazon S3 para o manifesto dos resultados da avaliação pode ser obtido no `EvaluationResult.Summary.S3Object` campo na DescribeProejctVersion resposta.

O exemplo a seguir mostra a estrutura do manifesto dos resultados da avaliação:

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

O arquivo de manifesto de avaliação contém:

- Resultados agregados conforme definido por `F1Score`
- Detalhes do trabalho de avaliação ProjectVersionArn, incluindo o número de imagens de treinamento, o número de imagens de teste e as etiquetas nas quais o adaptador foi treinado.
- FalseNegative Resultados agregados TruePositive, TrueNegative, FalsePositive, para o modelo básico e para o desempenho do adaptador.
- Por etiqueta TruePositive,, TrueNegative FalsePositive, e FalseNegative resultados para o modelo básico e o desempenho do adaptador, calculados no limite de confiança de entrada.
- FalseNegative Resultados agregados e por rótulo TruePositive, TrueNegative, FalsePositive, e para o desempenho do modelo básico e do adaptador em diferentes limites de confiança. O limite de confiança varia de 5 a 100 em etapas de 5.

Práticas recomendadas para adaptadores de treinamento

É sugerido que você siga as seguintes práticas recomendadas ao criar, treinar e usar seus adaptadores:

1. Os dados da imagem da amostra devem capturar os erros representativos que os clientes pretendem suprimir. Se o modelo estiver cometendo erros repetidos em imagens visualmente semelhantes, certifique-se de trazer muitas dessas imagens para treinamento.

2. Em vez de incluir apenas imagens de que a modelo cometeu erros em um rótulo de moderação específico, certifique-se de incluir imagens de que a modelo não esteja cometendo erros nesse rótulo de moderação.
3. Forneça no mínimo cinquenta amostras de falsos negativos OU vinte amostras de falsos positivos para treinamento e um mínimo de vinte amostras para teste. No entanto, forneça o máximo possível de imagens anotadas para melhorar o desempenho do adaptador.
4. Anotar todos os rótulos que são importantes para você em todas as imagens, se você decidir que precisa anotar a ocorrência de um rótulo em uma imagem, certifique-se de anotar a ocorrência desse rótulo em todas as outras imagens.
5. Os dados da imagem da amostra devem conter o máximo possível de variações na etiqueta, concentrando-se em instâncias representativas das imagens que serão analisadas em um ambiente de produção.

Configurando AutoUpdate permissões

O Rekognition suporta o recurso de adaptadores personalizados. AutoUpdate Isso significa que a reciclagem automatizada é feita da melhor maneira possível quando o AutoUpdate sinalizador está HABILITADO em um projeto. Essas atualizações automáticas exigem permissão para acessar seus conjuntos de dados de treinamento/teste e a AWS KMS chave com a qual você treina seu adaptador de cliente. Você pode fornecer essas permissões seguindo as etapas abaixo.

Permissões do bucket do Amazon S3

Por padrão, todos os buckets e objetos do Amazon S3 são privados. Somente o proprietário do recurso, a AWS conta que criou o bucket, pode acessar o bucket e quaisquer objetos que ele contenha. No entanto, o proprietário do recurso pode optar por conceder permissões de acesso a outros recursos e usuários escrevendo uma política de bucket.

Se você quiser criar ou modificar um bucket do Amazon S3 para ser usado como fonte de conjuntos de dados de entrada e destino dos resultados do treinamento em um treinamento de adaptador personalizado, você deve modificar ainda mais a política do bucket. Para ler ou gravar em um bucket do Amazon S3, o Rekognition deve ter as seguintes permissões.

Política de reconhecimento obrigatório do Amazon S3

O Rekognition exige uma política de permissão com os seguintes atributos:

- A declaração SID
- O nome do bucket
- O nome da entidade principal do serviço do Rekognition.
- Os recursos necessários para o Rekognition: o bucket e todo o seu conteúdo
- As ações necessárias que o Rekognition precisa realizar.

A política a seguir permite que o Rekognition acesse um bucket do Amazon S3 durante um novo treinamento automatizado.

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

Você pode seguir este guia para adicionar a política de bucket acima ao seu bucket do S3.

Veja mais informações sobre políticas de bucket aqui.

AWS KMS Permissões principais

O Rekognition permite que você forneça um opcional enquanto treina um adaptador personalizado. KmsKeyId Quando fornecida, o Rekognition usa essa chave para criptografar imagens de

treinamento e teste copiadas no serviço para treinamento de modelos. A chave também é usada para criptografar resultados de treinamento e arquivos de manifesto gravados no bucket OutputConfig de saída do Amazon S3 ().

Se você optar por fornecer uma chave do KMS como entrada para seu treinamento de adaptador personalizado (ou seja, `Rekognition:CreateProjectVersion`), deverá modificar a política de chave do KMS para permitir que a entidade principal de serviços do Rekognition use essa chave para reciclagem automatizada no futuro. O Rekognition deve ter as permissões a seguir.

Política de Chave Obrigatória do Rekognition AWS KMS

O Amazon Rekognition exige uma política de permissão com os seguintes atributos:

- A declaração SID
- O nome da entidade principal do serviço do Amazon Rekognition.
- As ações necessárias que o Amazon Rekognition precisa realizar.

A política de chave a seguir permite que o Amazon Rekognition acesse uma chave do Amazon KMS durante um novo treinamento automatizado:

```
{
  "Version": "2023-10-06",
  "Statement": [
    {
      "Sid": "KeyPermissions",
      "Effect": "Allow",
      "Principal": {
            "Service": "rekognition.amazonaws.com"
        },
      "Action": [
        "kms:DescribeKey",
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ]
      "Resource": "*"
    }
  ]
}
```

Você pode seguir este guia para adicionar a AWS KMS política acima à sua AWS KMS chave.

Veja mais informações sobre AWS KMS políticas aqui.

AWS Notificação do Health Dashboard para Rekognition

Seu AWS Health Dashboard fornece suporte para notificações provenientes do Rekognition. Essas notificações fornecem orientação de conscientização e remediação sobre mudanças programadas nos modelos do Rekognition que podem afetar seus aplicativos. Somente eventos específicos do recurso de moderação de conteúdo do Rekognition estão disponíveis atualmente.

O AWS Health Dashboard faz parte do serviço AWS Health. Ele não requer configuração e pode ser visualizado por qualquer usuário autenticado em sua conta. Para obter mais informações, consulte Conceitos básicos do AWS Health Dashboard.

Se você receber uma mensagem de notificação semelhante às mensagens a seguir, ela deverá ser tratada como um alarme para agir.

Exemplo de notificação: Uma nova versão do modelo está disponível para a moderação de conteúdo do Rekognition.

O Rekognition publica `AWS_MODERATION_MODEL_VERSION_UPDATE_NOTIFICATION` o evento no Health Dashboard para indicar que uma nova versão AWS do modelo de moderação foi lançada. Esse evento é importante se você estiver usando a DetectModerationLabels API e os adaptadores com essa API. Novos modelos podem afetar a qualidade dependendo do seu caso de uso e, eventualmente, substituirão as versões anteriores do modelo. É recomendável validar a qualidade do seu modelo e estar ciente dos cronogramas de atualização do modelo ao receber esse alerta.

Se você receber uma notificação de atualização da versão do modelo, deverá tratá-la como um alarme para agir. Se você não usa adaptadores, deve avaliar a qualidade do modelo atualizado em seu caso de uso existente. Se você usa adaptadores, deve treinar novos adaptadores com o modelo atualizado e avaliar sua qualidade. Se você tiver configurado o treinamento automático, os novos adaptadores serão treinados automaticamente e, em seguida, você poderá avaliar sua qualidade.


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

Consulte Monitoramento de eventos de saúde da AWS com EventBridge a Amazon para detectar e reagir aos eventos de AWS saúde usando EventBridge.

Revisão de conteúdo inadequado com o Amazon Augmented AI

O Amazon Augmented AI (Amazon A2I) permite construir os fluxos de trabalho necessários para a análise humana das previsões de machine learning.

O Amazon Rekognition está diretamente integrado no Amazon A2I para que você possa implementar facilmente a análise humana para o caso de uso na detecção de imagens não seguras. O Amazon A2I fornece um fluxo de trabalho de avaliação humana para moderação de imagens. Isto permite que você revise com facilidade as previsões do Amazon Rekognition. Você pode definir limiares de confiabilidade para seu caso de uso e ajustá-los ao longo do tempo. Com o Amazon A2I, você pode usar um grupo de revisores de sua própria empresa ou do Amazon Mechanical Turk. Você também pode usar fornecedores externos que já foram testados pela AWS em termos de qualidade e observância dos procedimentos de segurança.

As próximas etapas explicam como configurar o Amazon A2I com o Amazon Rekognition. Primeiro, você cria uma definição de fluxo com o Amazon A2I que tem as condições que acionam a revisão humana. Em seguida, você passa a definição de fluxo do Amazon Resource Name (ARN) para a operação `DetectModerationLabel` do Amazon Rekognition. Na resposta `DetectModerationLabel`, você pode ver se a revisão humana é necessária. Os resultados da revisão humana estão disponíveis em um bucket do Amazon S3 configurado na definição do fluxo.

Para ver uma end-to-end demonstração de como usar o Amazon A2I com o Amazon Rekognition, consulte um dos seguintes tutoriais no Amazon AI Developer Guide. SageMaker

- Demonstração: Introdução ao console do Amazon A2I
- Demonstração: Introdução à API do Amazon A2I

  Para começar a usar a API, você também pode executar um exemplo de caderno Jupyter. Consulte Usar uma instância de SageMaker notebook com o Amazon A2I Jupyter Notebook para usar a integração da Amazon Augmented AI (Amazon A2I) com o Amazon Rekognition [Exemplo] em uma instância de notebook de IA. SageMaker

Executando DetectModerationLabels com o Amazon A2I

> **Note**
>
> Crie todos os seus recursos do Amazon A2I e do Amazon Rekognition na mesma região da AWS.

1. Preencha os pré-requisitos listados em Getting Started with Amazon Augmented AI na documentação de IA. SageMaker

   Além disso, lembre-se de configurar suas permissões do IAM como na página Permissões e segurança na Amazon Augmented AI SageMaker na documentação de IA.
2. Siga as instruções para criar um fluxo de trabalho de revisão humana na documentação da SageMaker IA.

   O fluxo de trabalho de uma revisão humana gerencia o processamento de uma imagem. Ele contém as condições que acionam uma revisão humana, a equipe de trabalho que recebe a imagem, o modelo de interface do usuário usado pela equipe de trabalho e o bucket do Amazon S3 que recebe os resultados da equipe de trabalho.

Na sua `CreateFlowDefinition` chamada, você precisa definir o `HumanLoopRequestSource` como "AWS/Rekognition/DetectModerationLabels/Image/V3". Depois disso, precisa decidir como deseja configurar suas condições que acionam a revisão humana.

Com o Amazon Rekognition, você tem duas opções para `ConditionType`: `ModerationLabelConfidenceCheck` e `Sampling`.

`ModerationLabelConfidenceCheck` cria um loop humano quando a confiança de um rótulo de moderação está dentro de um intervalo. Por fim, `Sampling` envia uma porcentagem aleatória dos documentos processados para revisão humana. Cada `ConditionType` usa um conjunto diferente de `ConditionParameters` para definir o que resulta em revisão humana.

`ModerationLabelConfidenceCheck` tem o `ConditionParameters` `ModerationLableName`, que define a chave que precisa ser revisada por humanos. Além disso, possui confiança, que define a faixa percentual para envio à avaliação humana com LessThan GreaterThan, e Equals. `SamplingtemRandomSamplingPercentage`, que define uma porcentagem dos documentos que serão enviados para revisão humana.

O exemplo de código a seguir é uma chamada parcial de `CreateFlowDefinition`. Ele envia uma imagem para revisão humana se tiver classificação inferior a 98% no rótulo "Sugestivo", e mais de 95% no rótulo "Roupa de banho ou roupa interior feminina". Isto significa que se a imagem não é considerada sugestiva, mas tem uma mulher em roupa interior ou roupa de banho, você pode verificar novamente a imagem usando revisão humana.

```
def create_flow_definition():
'''
Creates a Flow Definition resource

Returns:
struct: FlowDefinitionArn
'''
humanLoopActivationConditions = json.dumps(
    {
        "Conditions": [
            {
              "And": [
                {
                    "ConditionType": "ModerationLabelConfidenceCheck",
```

```
                        "ConditionParameters": {
                            "ModerationLabelName": "Suggestive",
                            "ConfidenceLessThan": 98
                        }
                    },
                    {
                        "ConditionType": "ModerationLabelConfidenceCheck",
                        "ConditionParameters": {
                            "ModerationLabelName": "Female Swimwear Or Underwear",
                            "ConfidenceGreaterThan": 95
                        }
                    }
                ]
            }
        ]
    }
)
```

O `CreateFlowDefinition` retorna um `FlowDefinitionArn`, que é usado na próxima etapa, quando você chama `DetectModerationLabels`.

Para obter mais informações, consulte CreateFlowDefinitiona Referência da API de SageMaker IA.

3. Defina o parâmetro `HumanLoopConfig` ao chamar `DetectModerationLabels`, como em Detectando imagens inapropriadas. Consulte a etapa 4 para ver exemplos de uma chamada `DetectModerationLabels` com o `HumanLoopConfig` definido.

   a. Dentro do parâmetro `HumanLoopConfig`, defina o `FlowDefinitionArn` o como o ARN da definição de fluxo que você criou na etapa 2.

   b. Prepare o seu `HumanLoopName`. Isto deve ser exclusivo dentro de uma Região e estar em minúsculas.

   c. (Opcional) Você pode usar `DataAttributes` para definir se a imagem que você passou para o Amazon Rekognition nao contém informações pessoais identificáveis. Você deve definir esse parâmetro para enviar informações ao Amazon Mechanical Turk.

4. Executar `DetectModerationLabels`.

Os exemplos a seguir mostram como usar o AWS CLI e AWS SDK para Python (Boto3) para executar `DetectModerationLabels` com `HumanLoopConfig` set.

AWS SDK para Python (Boto3)

O exemplo de solicitação a seguir usa o SDK para Python (Boto3). Para obter mais informações, consulte detect_moderation_labels na referência da API AWS SDK para Python (Boto).

```
import boto3

rekognition = boto3.client("rekognition", aws-region)

response = rekognition.detect_moderation_labels( \
        Image={'S3Object': {'Bucket': bucket_name, 'Name': image_name}}, \
        HumanLoopConfig={ \
            'HumanLoopName': 'human_loop_name', \
            'FlowDefinitionArn': , "arn:aws:sagemaker:aws-
region:aws_account_number:flow-definition/flow_def_name" \
            'DataAttributes': {'ContentClassifiers':
 ['FreeOfPersonallyIdentifiableInformation','FreeOfAdultContent']}
         })
```

AWS CLI

O exemplo de solicitação a seguir usa a AWS CLI. Para obter mais informações, consulte detect-moderation-labels na Referência de comandos da 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 você executa `DetectModerationLabels` com `HumanLoopConfig` enabled, o Amazon SageMaker Rekognition chama a operação da API de IA. `StartHumanLoop` Esse comando obtém a resposta do `DetectModerationLabels` e verifica as condições da definição de fluxo no exemplo. Se atender às condições de revisão, ele retorna um `HumanLoopArn`. Isso significa que os membros da equipe de trabalho que você definiu em sua definição de fluxo agora podem revisar a imagem. Chamar a operação de runtime `DescribeHumanLoop` do Amazon Augmented AI fornece informações sobre o resultado do loop. Para obter mais informações, consulte a DescribeHumanLoopdocumentação de referência da API Amazon Augmented AI.

Depois que a imagem for revisada, você poderá ver os resultados no bucket especificado no caminho de saída da definição de fluxo. A Amazon A2I também notificará você com o Amazon CloudWatch Events quando a análise for concluída. Para ver quais eventos procurar, consulte CloudWatch Eventos na documentação da SageMaker IA.

Para obter mais informações, consulte Getting Started with Amazon Augmented AI SageMaker na documentação de IA.

Detectar texto

O Amazon Rekognition pode detectar texto em imagens e vídeos. Depois, é possível converter o texto detectado em texto legível por máquina. Você pode usar a detecção de texto legível por máquina em imagens para implementar soluções como:

- Pesquisa visual. Por exemplo, recuperar e exibir imagens que contêm o mesmo texto.
- Insights de conteúdo. Por exemplo, fornecer informações sobre temas que ocorrem em textos reconhecidos em quadros de vídeo extraídos. O aplicativo pode pesquisar conteúdo relevante em texto reconhecido, como notícias, placares esportivos, números de atletas e legendas.
- Navegação. Por exemplo, desenvolver um aplicativo móvel com recurso de fala para pessoas com deficiência visual que reconheça nomes de restaurantes, lojas ou placas de rua.
- Segurança pública e suporte de transporte. Por exemplo, detectar números de placas de carros a partir de imagens de câmeras de trânsito.
- Filtrando. Por exemplo, filtrar informações de identificação pessoal (PII) de imagens.

Para a detecção de texto em vídeos, é possível implementar soluções como:

- Pesquisar vídeos em busca de clipes com palavras-chave de texto específicas, como o nome de um convidado em um gráfico em um programa de notícias.
- Moderar o conteúdo para conformidade com os padrões organizacionais, detectando texto acidental, palavrões ou spam.
- Encontrar todas as sobreposições de texto na linha do tempo do vídeo para processamento adicional, como substituir texto por texto em outro idioma para internacionalização do conteúdo.
- Encontrar locais de texto, para que outros gráficos possam ser alinhados adequadamente.

Para detectar texto em imagens no formato JPEG ou PNG, use a DetectTextoperação. Para detectar texto em vídeo de forma assíncrona, use as operações e. StartTextDetectionGetTextDetection As operações de detecção de texto em imagem e vídeo suportam a maioria das fontes, inclusive as altamente estilizadas. Depois de detectar o texto, o Amazon Rekognition cria uma representação das palavras e linhas de texto detectadas, mostra a relação entre elas e informa onde o texto está em um quadro de imagem ou vídeo.

As operações `GetTextDetection` e `DetectText` detectam palavras e linhas. Uma palavra é um ou mais caracteres de script que não estão separados por espaços. `DetectText` pode detectar até

100 palavras em uma imagem. `GetTextDetection` também pode detectar até 100 palavras por quadro de vídeo.

Uma palavra é um ou mais caracteres de script que não estão separados por espaços. O Amazon Rekognition foi projetado para detectar palavras em inglês, árabe, russo, alemão, francês, italiano, português e espanhol.

Uma linha é uma sequência de palavras igualmente espaçadas. Uma linha não é necessariamente uma frase completa (os pontos não indicam o final de uma linha). Por exemplo, o Amazon Rekognition detecta o número da carteira de motorista como uma linha. Uma linha termina quando não há texto alinhado depois dela ou quando há uma grande lacuna entre as palavras, em relação ao comprimento das palavras. Dependendo da lacuna entre as palavras, o Amazon Rekognition pode detectar várias linhas no texto alinhadas na mesma direção. Se uma frase tiver várias linhas, a operação retornará várias linhas.

Considere a imagem a seguir.



As caixas azuis representam informações sobre o texto detectado e a localização do texto retornado pela operação `DetectText`. Neste exemplo, o Amazon Rekognition detecta "É", "SEGUNDA-FEIRA", "mas", "continue" e "Sorrindo" como palavras. O Amazon Rekognition detecta "É",

"SEGUNDA-FEIRA", "mas continue" e "Sorrindo" como linhas. Para ser detectado, o texto deve estar na orientação de +/- 90 graus do eixo horizontal.

Para obter um exemplo, consulte Detectar texto em uma imagem.

Tópicos

- Detectar texto em uma imagem
- Detectar texto em um vídeo armazenado

Detectar texto em uma imagem

Você pode fornecer uma imagem de entrada como uma matriz de bytes de imagem (bytes de imagem codificados em base64) ou como um objeto do Amazon S3. Neste procedimento, você carrega uma imagem JPEG ou PNG no bucket do S3 e especifica o nome do arquivo.

Para detectar texto em uma imagem (API)

1. Se você ainda não o fez, preencha os seguintes pré-requisitos.

   a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

   b. Instale e configure o AWS Command Line Interface e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Faça upload da imagem que contém texto para seu bucket do S3.

   Para obter instruções, consulte Como fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

3. Use os exemplos a seguir para chamar a operação `DetectText`.

   Java

   O código de exemplo a seguir exibe linhas e palavras que foram detectadas em uma imagem.

   Substitua os valores de `amzn-s3-demo-bucket` e `photo` pelos nomes do bucket S3 e da imagem usados na etapa 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

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
/**
*  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

Esse AWS CLI comando exibe a saída JSON para a operação da `detect-text` CLI.

Substitua os valores de `amzn-s3-demo-bucket` e `Name` pelos nomes do bucket S3 e da imagem usados na etapa 2.

Substitua o valor de `profile_name` com o nome do seu perfil de desenvolvedor.

```
aws rekognition detect-text  --image "{"S3Object":{"Bucket":"amzn-s3-demo-
bucket","Name":"image-name"}}" --profile default
```

Se você estiver acessando a CLI em um dispositivo Windows, use aspas duplas em vez de aspas simples e escape das aspas duplas internas com barra invertida (ou seja, \) para resolver quaisquer erros de analisador que você possa encontrar. Para obter um exemplo, veja o seguinte:

```
aws rekognition detect-text  --image "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --profile default
```

Python

O código de exemplo a seguir exibe linhas e palavras detectadas em uma imagem.

Substitua os valores de `amzn-s3-demo-bucket` e `photo` pelos nomes do bucket S3 e da imagem usados na etapa 2. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

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

O código de exemplo a seguir exibe linhas e palavras detectadas em uma imagem.

Substitua os valores de `amzn-s3-demo-bucket` e `photo` pelos nomes do bucket S3 e da imagem usados na etapa 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

O código de exemplo a seguir exibe linhas e palavras detectadas em uma imagem.

Substitua os valores de `amzn-s3-demo-bucket` e `photo` pelos nomes do bucket S3 e da imagem usados na etapa 2. Substitua o valor de `region` pela região encontrada em suas credenciais .aws. Substitua o valor de `profile_name` na linha que cria a sessão do Rekognition pelo nome do seu perfil de desenvolvedor.

```
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 solicitação de operação

Na operação `DetectText`, você fornece uma imagem de entrada como uma matriz de bytes codificada em base64 ou como uma imagem armazenada em um bucket do Amazon S3. O exemplo de solicitação JSON a seguir mostra a imagem carregada de um bucket do Amazon S3.

```
{
    "Image": {
        "S3Object": {
```

```
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "inputtext.jpg"
        }
    }
}
```

Filtros

A filtragem do texto por região, tamanho e pontuação de confiança oferece flexibilidade adicional para controlar a saída de detecção de seu texto. Usando regiões de interesse, é possível limitar facilmente a detecção de texto para as regiões relevantes para você, por exemplo, a parte superior direita da foto de perfil ou um local fixo em relação a um ponto de referência ao ler números de peças da imagem de uma máquina. O filtro de tamanho da caixa delimitadora de palavras pode ser usado para evitar texto pequeno em segundo plano que pode ser ruído ou irrelevante. O filtro de confiança de palavra permite que você remova resultados que podem não ser confiáveis devido a estarem embaçados ou borrados.

Para obter informações sobre valores de filtro, consulte `DetectTextFilters`.

É possível usar os seguintes filtros:

- MinConfidence—Define o nível de confiança da detecção de palavras. Palavras com confiança de detecção abaixo desse nível são excluídas do resultado. Os valores devem estar entre 0 e 100.
- MinBoundingBoxWidth— Define a largura mínima da caixa delimitadora da palavra. Palavras com caixas delimitadoras menores que esse valor são excluídas do resultado. O valor é relativo à largura do quadro da imagem.
- MinBoundingBoxHeight— Define a altura mínima da caixa delimitadora da palavra. Palavras com alturas de caixa delimitadora menores que este valor são excluídas do resultado. O valor é relativo à altura do quadro da imagem.
- RegionsOfInterest— Limita a detecção a uma região específica do quadro da imagem. Os valores são relativos às dimensões do quadro. Para texto apenas parcialmente dentro de uma região, a resposta é indefinida.

DetectText resposta da operação

A `DetectText` operação analisa a imagem e retorna uma matriz TextDetections, em que cada elemento (`TextDetection`) representa uma linha ou palavra detectada na imagem. Para cada elemento, o `DetectText` retorna as seguintes informações:

- O texto detectado (`DetectedText`)
- As relações entre palavras e linhas (`Id` e `ParentId`)
- A localização do texto na imagem (`Geometry`)
- A confiança que o Amazon Rekognition tem na precisão do texto detectado e da caixa delimitadora (`Confidence`)
- O tipo do texto detectado (`Type`)

Texto detectado

Cada elemento de `TextDetection` contém texto reconhecido (palavras ou linhas) no campo `DetectedText`. Uma palavra é um ou mais caracteres de script não separados por espaços. `DetectText` pode detectar até 100 palavras em uma imagem. O texto retornado pode incluir caracteres que tornam uma palavra irreconhecível. Por exemplo, G@to em vez de Gato. Para determinar se um elemento do `TextDetection` representa uma linha de texto ou uma palavra, use o campo `Type`.

Cada elemento `TextDetection` inclui um valor percentual que representa o grau de confiança que o Amazon Rekognition tem na precisão do texto detectado e da caixa delimitadora que envolve o texto.

Relações entre palavras e linhas

Cada elemento de `TextDetection` tem um identificador campo, `Id`. O `Id` mostra a posição da palavra em uma linha. Se o elemento for uma palavra, o campo do identificador pai, `ParentId`, identificará a linha onde a palavra foi detectada. O `ParentId` de uma linha é nulo. Por exemplo, a linha "mas continue" na imagem anterior tem os seguintes valores de `Id` e `ParentId`:

Texto	ID	ID do pai
but keep	3	
but	8	3
keep	9	3

Localização do texto em uma imagem

Para determinar onde o texto reconhecido está em uma imagem, use as informações da caixa delimitadora (Geometria) que são retornadas por `DetectText`. O objeto `Geometry` contém dois tipos de informações de caixa delimitadora para linhas e palavras detectadas:

- Um contorno retangular grosseiro alinhado ao eixo em um objeto BoundingBox
- Um polígono mais refinado que é composto de várias coordenadas X e Y em uma matriz de pontos

As coordenadas da caixa delimitadora e do polígono mostram onde o texto está localizado na imagem de origem. Os valores das coordenadas são uma proporção do tamanho geral da imagem. Para obter mais informações, consulte BoundingBox.

A seguinte resposta do JSON da operação `DetectText` mostra as palavras e linhas detectadas na imagem a seguir.




```
{
 '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'}
```

Detectar texto em um vídeo armazenado

A detecção de texto em vídeos armazenados pelo Amazon Rekognition Video é uma operação assíncrona. Para começar a detectar texto, ligue StartTextDetection. O Amazon Rekognition Video publica o status de conclusão da análise de vídeo em um tópico do Amazon SNS. Se a análise do vídeo for bem-sucedida, chame GetTextDetection para obter os resultados da análise. Para obter mais informações sobre como iniciar uma análise de vídeo e obter os resultados, consulte Chamando as operações de vídeo do Amazon Rekognition Video.

Esse procedimento expande o código em Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK). Ele usa uma fila do Amazon SQS para obter o status de conclusão de uma solicitação de análise de vídeo.

Para detectar texto em um vídeo armazenado em um bucket do Amazon S3 (SDK)

1. Siga as etapas em Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK).
2. Adicione o código a seguir à classe `VideoDetect` na etapa 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);


}
```

Na função `main`, substitua as linhas:

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

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

por:

```
        StartTextDetection(amzn-s3-demo-bucket, video);

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

Java V2

Esse código foi retirado do GitHub repositório de exemplos do SDK de AWS documentação. Veja o exemplo completo aqui.

```
//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
```

Na função `main`, substitua as linhas:

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

por:

```
    analyzer.StartTextDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetTextDetectionResults()
```

CLI

Execute o AWS CLI comando a seguir para começar a detectar texto em um vídeo.

```
 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
```

Atualize os seguintes valores:

- Mude `amzn-s3-demo-bucket` e `video-name` para o nome do bucket do Amazon S3 e o nome do arquivo que você especificou na etapa 2.
- Altere `region-name` para a região da AWS que você está usando.
- Substitua o valor de `profile-name` com o nome do seu perfil de desenvolvedor.
- Mude `topic-ARN` para o ARN do tópico do Amazon SNS que você criou na etapa 3 do Configuração do Amazon Rekognition Video.
- Mude `role-ARN` para o ARN do perfil de serviço do IAM que você criou na etapa 7 do Configuração do Amazon Rekognition Video.

Se você estiver acessando a CLI em um dispositivo Windows, use aspas duplas em vez de aspas simples e escape das aspas duplas internas com barra invertida (ou seja, \) para resolver quaisquer erros de analisador que você possa encontrar. Para ver um exemplo, veja abaixo:

```
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
```

Depois de executar o exemplo do código de procedimento, copie o `jobID` retornado e forneça-o ao seguinte comando `GetTextDetection` abaixo para obter seus resultados, substitua `job-id-number` pelo `jobID` que você recebeu anteriormente:

```
aws rekognition get-text-detection --job-id job-id-number --profile profile-name
```

> **Note**
>
> Se você já tiver executado um exemplo de vídeo diferente de Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK), o código a ser substituído poderá ser diferente.

3. Execute o código. O texto detectado no vídeo é mostrado em uma lista.

Filtros

Os filtros são parâmetros de solicitação opcionais que podem ser usados ao chamar `StartTextDetection`. A filtragem do texto por região, tamanho e pontuação de confiança oferece flexibilidade adicional para controlar a saída de detecção de seu texto. Usando regiões de interesse, é possível limitar facilmente a detecção de texto a regiões que sejam relevantes para você. Por exemplo, uma região no terço inferior referente a elementos gráficos ou um canto superior esquerdo para ler placares em um jogo de futebol. O filtro de tamanho da caixa delimitadora de palavras pode ser usado para evitar texto pequeno em segundo plano que pode ser ruído ou irrelevante. Por fim, o filtro de confiança de palavras permite remover resultados que podem não ser confiáveis por estarem borrados ou manchados.

Para obter informações sobre valores de filtro, consulte `DetectTextFilters`.

É possível usar os seguintes filtros:

- MinConfidence—Define o nível de confiança da detecção de palavras. Palavras com confiança de detecção abaixo desse nível são excluídas do resultado. Os valores devem estar entre 0 e 100.
- MinBoundingBoxWidth— Define a largura mínima da caixa delimitadora da palavra. Palavras com caixas delimitadoras menores que esse valor são excluídas do resultado. O valor é relativo à largura do quadro de vídeo.
- MinBoundingBoxHeight— Define a altura mínima da caixa delimitadora da palavra. Palavras com alturas de caixa delimitadora menores que este valor são excluídas do resultado. O valor é relativo à altura do quadro de vídeo.
- RegionsOfInterest— Limita a detecção a uma região específica do quadro. Os valores são relativos às dimensões do quadro. Para objetos apenas parcialmente dentro das regiões, a resposta é indefinida.

GetTextDetection resposta

`GetTextDetection` retorna uma matriz (`TextDetectionResults`) que contém informações sobre as faces detectadas no vídeo. Existe um elemento da matriz, TextDetection, para cada vez que uma palavra ou linha é detectada no vídeo. Os elementos da matriz são classificados por tempo, em milissegundos desde o início do vídeo.

O exemplo a seguir é a resposta parcial do JSON de `GetTextDetection`. Na resposta, observe o seguinte:

- Informações de texto — O elemento da `TextDetectionResult` matriz contém informações sobre o texto detectado (TextDetection) e a hora em que o texto foi detectado no vídeo (`Timestamp`).
- Informações de paginação: o exemplo mostra uma página de informações de detecção de texto. É possível especificar quantos elementos de texto retornar no parâmetro de entrada `MaxResults` para `GetTextDetection`. Se houver mais resultados do que `MaxResults` ou se houver mais resultados do que o máximo padrão, `GetTextDetection` retornará um token (`NextToken`) que será usado para obter a próxima página de resultados. Para obter mais informações, consulte Obter os resultados da análise do Amazon Rekognition Video.
- Informações de vídeo – A resposta inclui informações sobre o formato do vídeo (`VideoMetadata`) em cada página de informações retornada pelo `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"
}
```

Detectando segmentos de vídeo em vídeo armazenado

O Amazon Rekognition Video fornece uma API que identifica segmentos úteis de vídeo, como quadros pretos e créditos finais.

Os espectadores estão assistindo a mais conteúdo do que nunca. Em particular, as plataformas Over-The-Top (OTT) e Video-On-Demand (VOD) oferecem uma ampla seleção de opções de conteúdo a qualquer hora, em qualquer lugar e em qualquer tela. Com o aumento dos volumes de conteúdo, as empresas de mídia estão enfrentando desafios na preparação e no gerenciamento de seu conteúdo. Isso é crucial para proporcionar uma experiência de visualização de alta qualidade e melhor monetização do conteúdo. Atualmente, as empresas usam grandes equipes com forças de trabalho humanas treinadas para executar tarefas como as seguintes:

- Descobrir onde estão os créditos iniciais e finais em um conteúdo
- Escolher os locais certos para inserir anúncios, como em sequências silenciosas de molduras pretas
- Dividindo vídeos em clipes menores para uma melhor indexação

Esses processos manuais são caros, lentos e não podem ser dimensionados para acompanhar o volume de conteúdo produzido, licenciado e recuperado dos arquivos diariamente.

Você pode usar o Amazon Rekognition Video para automatizar tarefas operacionais de análise de mídia usando a detecção de segmentos de vídeo totalmente gerenciada e desenvolvida especificamente com base no aprendizado de máquina (ML). APIs Ao usar o segmento Amazon Rekognition APIs Video, você pode analisar facilmente grandes volumes de vídeos e detectar marcadores, como quadros pretos ou alterações na captura. Você obtém códigos de tempo, carimbos de data/hora e números de quadros da SMPTE (Society of Motion Picture and Television Engineers) de cada detecção. Nenhuma experiência em ML é necessária.

O Amazon Rekognition Video analisa vídeos armazenados em um bucket do Amazon Simple Storage Service (Amazon S3). Os timecodes SMPTE retornados têm precisão de quadros — o Amazon Rekognition Video fornece o número exato de quadros de um segmento de vídeo detectado e processa vários formatos de taxa de quadros de vídeo automaticamente. Você pode usar os metadados com precisão de quadros do Amazon Rekognition Video para automatizar completamente determinadas tarefas ou reduzir significativamente a workload de revisão de operadores humanos treinados, para que eles possam se concentrar em um trabalho mais criativo.

Você pode realizar tarefas como preparar conteúdo, inserir anúncios e adicionar "marcadores compulsivos" ao conteúdo em grande escala na nuvem.

Para obter informações sobre preços, consulte os preços do Amazon Rekognition.

A detecção de segmentos do Amazon Rekognition Video oferece suporte a dois tipos de tarefas de segmentação: detecção Dicas técnicas e Detecção de captura.

Tópicos

- Dicas técnicas
- Detecção de captura
- Sobre a API de detecção de segmentos do Amazon Rekognition Video
- Usar a API Amazon Rekognition Segment
- Exemplo: Detectando segmentos em um vídeo armazenado

Dicas técnicas

Uma dica técnica identifica molduras pretas, barras coloridas, créditos de abertura, créditos finais, logotipos de estúdio e conteúdo principal do programa em um vídeo.

Molduras pretas

Os vídeos geralmente contêm molduras pretas vazias sem áudio que são usadas como sinais para inserir anúncios ou marcar o final de um segmento do programa, como uma cena ou créditos de abertura. Com o Amazon Rekognition Video, você pode detectar sequências de quadros pretos para automatizar a inserção de anúncios, empacotar conteúdo para VOD e demarcar vários segmentos ou cenas do programa. Os quadros pretos com áudio (como desaparecimentos graduais ou narrações) são considerados como conteúdo e não são retornados.

Créditos

O Amazon Rekognition Video pode identificar automaticamente os quadros exatos em que os créditos de abertura e encerramento de um filme ou programa de TV começam e terminam. Com essas informações, você pode gerar "marcadores compulsivos" ou solicitações interativas para o espectador, como "Próximo episódio" ou "Skip Intro", em aplicativos de vídeo sob demanda (VOD). Você também pode detectar o primeiro e o último quadro do conteúdo do programa em um vídeo.

O Amazon Rekognition Video é treinado para lidar com uma grande variedade de estilos de crédito inicial e final, desde créditos contínuos simples até créditos mais desafiadores junto com o conteúdo.

Barras de cores

O Amazon Rekognition Video permite que você detecte seções de vídeo que exibem barras de cores SMPTE, que são um conjunto de cores exibido em padrões específicos para garantir que a cor seja calibrada corretamente em monitores de transmissão, programas e câmeras. Para obter mais informações sobre barras de cores SMPTE, consulte Barra de cores SMPTE. Esses metadados são úteis para preparar o conteúdo para aplicativos de VOD, removendo segmentos da barra de cores do conteúdo ou para detectar problemas como perda de sinais de transmissão em uma gravação, quando as barras de cores são mostradas continuamente como um sinal padrão em vez de conteúdo.

Barreiras

As barreiras são seções do vídeo, geralmente próximas ao início, que contêm metadados de texto sobre o episódio, o estúdio, o formato do vídeo, os canais de áudio e muito mais. O Amazon Rekognition Video pode identificar o início e o fim das listas, facilitando o uso dos metadados de texto ou a remoção da chapa ao preparar o conteúdo para visualização final.

Logotipos de estúdio

Os logotipos do estúdio são sequências que mostram os logotipos ou emblemas do estúdio de produção envolvido na criação do programa. O Amazon Rekognition Video pode detectar essas sequências para que os usuários possam analisá-las para identificar estúdios.

Conteúdo

O conteúdo são as partes do programa de TV ou filme que contêm o programa ou elementos relacionados. Molduras pretas, créditos, barras coloridas, ardósias e logotipos de estúdio não são considerados conteúdo. O Amazon Rekognition Video pode detectar o início e o fim de cada segmento de conteúdo no vídeo, para que você possa encontrar o tempo de execução do programa ou segmentos específicos.

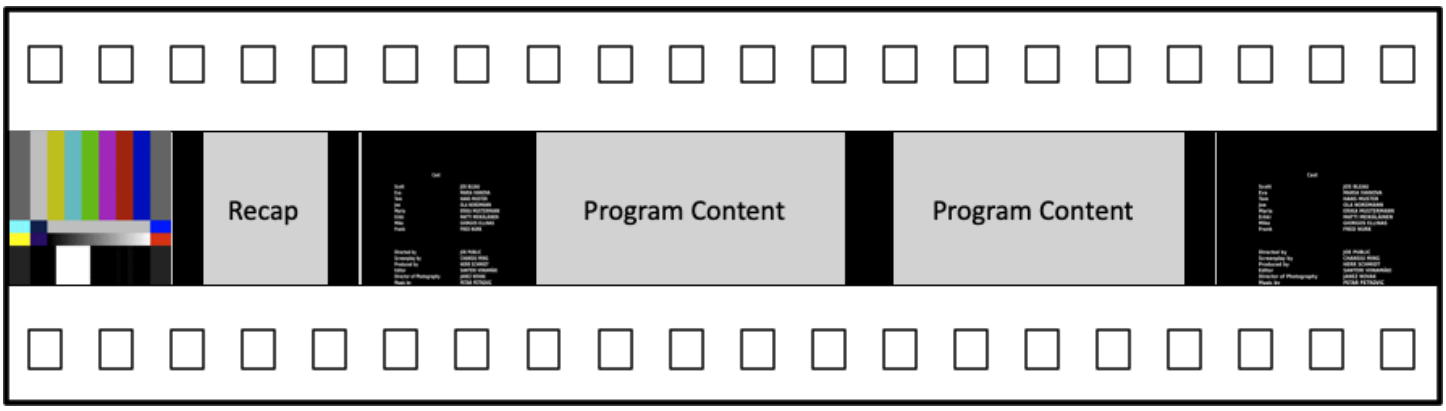
Os segmentos de conteúdo incluem, mas não estão limitados ao seguinte:

- Programe cenas entre dois intervalos publicitários
- Uma rápida recapitulação do episódio anterior no início do vídeo

- Conteúdo bônus pós-crédito
- Conteúdo "sem texto", como um conjunto de todas as cenas do programa que originalmente continham texto sobreposto, mas em que o texto foi removido para oferecer suporte à tradução para outros idiomas.

Depois que o Amazon Rekognition Video terminar de detectar todos os segmentos de conteúdo, você poderá aplicar o conhecimento do domínio ou enviá-los para análise humana para categorizar ainda mais cada segmento. Por exemplo, se você usa vídeos que sempre começam com uma recapitulação, você pode categorizar o primeiro segmento de conteúdo como uma recapitulação.

O diagrama a seguir mostra segmentos de dicas técnicas na linha do tempo de um programa ou filme. Observe as barras coloridas e os créditos de abertura, os segmentos de conteúdo, como o resumo e o programa principal, as molduras pretas em todo o vídeo e os créditos finais.



Detecção de captura

Uma tomada é uma série de fotos consecutivas inter-relacionadas, tiradas de forma contígua por uma única câmera e representando uma ação contínua no tempo e no espaço. Com o Amazon Rekognition Video, você pode detectar o início, o fim e a duração de cada captura, bem como a contagem de todas as fotos em um conteúdo. É possível usar metadados de tomada para tarefas como as seguintes:

- Criar vídeos promocionais usando tomadas selecionadas.
- Inserir anúncios em locais que não atrapalhem a experiência do espectador, como no meio de uma tomada quando alguém estiver falando.
- Gerar um conjunto de miniaturas de visualização que evitem o conteúdo de transição entre as tomadas.

Uma detecção de tomada é marcada no quadro exato em que há um corte definido para uma câmera diferente. Se houver uma transição suave de uma câmera para outra, o Amazon Rekognition Video omite a transição. Isso garante que os momentos de início e término da tomada não incluam seções sem conteúdo real.

O diagrama a seguir ilustra segmentos de detecção de tomada em um filme cinematográfico. Observe que cada tomada é identificada por um corte de um ângulo ou localização da câmera para o próximo.



Sobre a API de detecção de segmentos do Amazon Rekognition Video

Para segmentar um vídeo armazenado, você usa as operações assíncronas StartSegmentDetectione de GetSegmentDetectionAPI para iniciar um trabalho de segmentação e buscar os resultados. A detecção de segmentos aceita vídeos armazenados em um bucket do Amazon S3 e retorna uma saída JSON. Você pode optar por detectar somente sinais técnicos, somente alterações de captura ou ambas juntas configurando a solicitação da API `StartSegmentdetection`. Também é possível filtrar segmentos detectados definindo limites para uma confiança mínima de previsão. Para obter mais informações, consulte Usar a API Amazon Rekognition Segment . Para obter um código de exemplo, consulte Exemplo: Detectando segmentos em um vídeo armazenado.

Usar a API Amazon Rekognition Segment

A detecção de segmentos do Amazon Rekognition Video em vídeos armazenados é uma operação assíncrona do Amazon Rekognition Video. A API Amazon Rekognition Segment é uma API composta na qual você escolhe o tipo de análise (dicas técnicas ou detecção de disparos) em uma única chamada de API. Para obter informações sobre como chamar operações assíncronas, consulte Chamando as operações de vídeo do Amazon Rekognition Video.

Tópicos

- Iniciando a análise do segmento
- Obtendo resultados de análise de segmentos

Iniciando a análise do segmento

Como iniciar a detecção de segmentos em uma chamada de vídeo armazenada StartSegmentDetection. Os parâmetros de entrada são os mesmos de outras operações de vídeo do Amazon Rekognition Video com a adição da seleção do tipo de segmento e da filtragem de resultados. Para obter mais informações, consulte Iniciar a análise de vídeo.

A seguir está um exemplo de JSON passado pela `StartSegmentDetection`. A solicitação especifica que os segmentos de detecção de tomada e sinais técnicos sejam detectados. Diferentes filtros para a confiança mínima de detecção são necessários para os segmentos de sinais técnicos (90%) e os segmentos de detecção de tomada (80%).

```
{
  "Video": {
    "S3Object": {
      "Bucket": "test_files",
      "Name": "test_file.mp4"
    }
    "SegmentTypes":["TECHNICAL_CUES", "SHOT"]
    "Filters": {
      "TechnicalCueFilter": {
         "MinSegmentConfidence": 90,
         "BlackFrame" : {
            "MaxPixelThreshold": 0.1,
            "MinCoveragePercentage": 95
         }
      },
      "ShotFilter" : {
          "MinSegmentConfidence": 60
      }
  }
}
```

Escolher um tipo de segmento

Use o parâmetro de entrada da matriz `SegmentTypes` para detectar segmentos de sinais técnicos e/ou de detecção de tomada no vídeo de entrada.

- TECHNICAL_CUE: identifica registros de data e hora com precisão de quadros para o início, o fim e a duração das dicas técnicas (molduras pretas, barras coloridas, créditos de abertura, créditos finais, logotipos de estúdio e conteúdo principal do programa) detectadas em um vídeo. Por exemplo, é possível usar sinais técnicos para encontrar o início dos créditos finais. Para obter mais informações, consulte Dicas técnicas.
- SHOT: identifica o início, o fim e a duração de uma captura. Por exemplo, é possível usar a detecção de tomada para identificar tomadas candidatas para a edição final de um vídeo. Para obter mais informações, consulte Detecção de captura.

Filtrar os resultados da análise

É possível usar o parâmetro de entrada `Filters` (StartSegmentDetectionFilters) para especificar a confiança mínima de detecção retornada na resposta. Em `Filters`, use `ShotFilter` (StartShotDetectionFilter) para filtrar as tomadas detectadas. Use `TechnicalCueFilter` (StartTechnicalCueDetectionFilter) para filtrar dicas técnicas.

Para obter um código de exemplo, consulte Exemplo: Detectando segmentos em um vídeo armazenado.

Obtendo resultados de análise de segmentos

O Amazon Rekognition Video publica o status de conclusão da análise de vídeo em um tópico do Amazon Simple Notification Service. Se a análise do vídeo for bem-sucedida, ligue GetSegmentDetectionpara obter os resultados da análise do vídeo.

Veja a seguir uma solicitação `GetSegmentDetection` de exemplo. O `JobId` é o identificador do trabalho retornado da chamada para `StartSegmentDetection`. Para obter informações sobre outros parâmetros de entrada, consulte Obter os resultados da análise do Amazon Rekognition Video.

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

```
    "NextToken": "XfXnZKiyMOGDhzBzYUhS5puM+g1IgezqFeYpv/H/+5noP/LmM57FitUAwSQ5D6G4AB/
PNwolrw=="
}
```

`GetSegmentDetection` retorna os resultados da análise solicitada e informações gerais sobre o vídeo armazenado.

Informações gerais

`GetSegmentDection` retorna as seguintes informações gerais:

- Informações de áudio — A resposta inclui metadados de áudio em uma matriz,`AudioMetadata`, de AudioMetadataobjetos. Pode haver vários streams de áudio. Cada objeto `AudioMetadata` contém metadados para um único stream de áudio. As informações de áudio em um objeto `AudioMetadata` incluem o codec de áudio, o número de canais de áudio, a duração do stream de áudio e a taxa de amostragem. Metadados de áudio são retornados em cada página de informações retornadas por `GetSegmentDetection`.
- Informações de vídeo — Atualmente, o Amazon Rekognition Video retorna um único objeto na matriz. VideoMetadata`VideoMetadata` O objeto contém informações sobre o stream de vídeo no arquivo de entrada que o Amazon Rekognition Video escolheu analisar. O objeto `VideoMetadata` inclui o codec de vídeo, o formato de vídeo e outras informações. Metadados de vídeo são retornados em cada página de informações retornadas por `GetSegmentDetection`.
- Informações de paginação — O exemplo mostra uma página de informações do segmento. É possível especificar quantos elementos de texto a serem retornados no parâmetro de entrada `MaxResults` para `GetSegmentDetection`. Se existirem mais resultados além de `MaxResults`, o `GetSegmentDetection` retornará um token (`NextToken`) usado para obter a próxima página de resultados. Para obter mais informações, consulte Obter os resultados da análise do Amazon Rekognition Video.
- Solicitar informações: o tipo de análise solicitado na chamada para `StartSegmentDetection` é retornado no campo `SelectedSegmentTypes`.

Segmentos

Os sinais técnicos e as informações de tomada detectados em um vídeo são retornados em uma matriz, `Segments`, de objetos SegmentDetection. A matriz é classificada pelos tipos de segmento (TECHNICAL_CUE ou SHOT) especificados no parâmetro de entrada `SegmentTypes` de `StartSegmentDetection`. Em cada tipo de segmento, a matriz é classificada por valores de time

stamp. Cada objeto `SegmentDetection` inclui informações sobre o tipo de segmento detectado (sinal técnico ou detecção de tomada) e informações gerais, como a hora de início, a hora de término e a duração do segmento.

As informações de horário são retornadas em três formatos.

- Milissegundos

  O número de milissegundos desde o início do vídeo. Os campos `DurationMillis`, `StartTimestampMillis`, e `EndTimestampMillis` estão no formato de milissegundos.

- Timecode

  Os timecodes do Amazon Rekognition Video estão no formato SMPTE, em que cada quadro de vídeo tem um valor de timecode exclusivo. O formato é hh:mm:ss:quadro. Por exemplo, um valor de código de tempo de 01:05:40:07, seria lido como uma hora, cinco minutos, quarenta segundos e sete quadros. Os casos de uso de taxa de quadros reduzida são compatíveis com o Amazon Rekognition Video. O formato do código de tempo com taxa de descarte de quadro é hh:mm:ss;quadro. Os campos `DurationSMPTE`, `StartTimecodeSMPTE` e `EndTimecodeSMPTE` estão no formato de código de tempo.

- Contadores de quadros

  A duração de cada segmento de vídeo também é expressa com o número de quadros. O campo `StartFrameNumber` fornece o número do quadro no início de um segmento de vídeo e `EndFrameNumber` fornece o número do quadro no final de um segmento de vídeo. `DurationFrames` fornece o número total de quadros em um segmento de vídeo. Esses valores são calculados usando um índice de quadros que começa com 0.

Você pode usar o campo `SegmentType` para determinar o tipo de segmento retornado pelo Amazon Rekognition Video.

- Dicas técnicas — o `TechnicalCueSegment` campo é um TechnicalCueSegmentobjeto que contém a confiança na detecção e o tipo de uma dica técnica. Os tipos de dicas técnicas são `ColorBars`, `EndCredits`, `BlackFrames`, `OpeningCredits`, `StudioLogo`, `Slate` e `Content`.
- Foto — o `ShotSegment` campo é um ShotSegmentobjeto que contém a confiança de detecção e um identificador para o segmento de captura dentro do vídeo.

O exemplo a seguir é a resposta do JSON em `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
        }
    ]
}
```

Para obter um código de exemplo, consulte Exemplo: Detectando segmentos em um vídeo armazenado.

Exemplo: Detectando segmentos em um vídeo armazenado

O procedimento a seguir mostra como detectar segmentos de sinalização técnica e segmentos de detecção de captura em um vídeo armazenado em um bucket do Amazon S3. O procedimento também mostra como filtrar segmentos detectados com base na confiança que o Amazon Rekognition Video tem na precisão da detecção.

O exemplo expande o código em Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK) no qual usa uma fila do Amazon Simple Queue Service para obter o status de conclusão de uma solicitação de análise de vídeo.

Para detectar segmentos em um vídeo armazenado em um bucket (SDK) do Amazon S3

1. Execute Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK).
2. Adicione o seguinte ao código que você usou na etapa 1.

   Java

   1. Adicione as seguintes importações.

      ```
      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. Adicione o código a seguir à 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. Na função `main`, substitua as linhas:

```
       StartLabelDetection(amzn-s3-demo-bucket, video);

       if (GetSQSMessageSuccess()==true)
```

```
         GetLabelDetectionResults();
```

por:

```
        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. Adicione o código a seguir à classe `VideoDetect` criada por você na etapa 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. Na função `main`, substitua as linhas:

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

por:

```
    analyzer.StartSegmentDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetSegmentDetectionResults()
```

> **Note**
>
> Se você já tiver executado um exemplo de vídeo diferente de Análise de um vídeo armazenado em um bucket do Amazon S3 com Java ou Python (SDK), o código a ser substituído poderá ser diferente.

3. Execute o código. São exibidas informações sobre os segmentos detectados no vídeo de entrada.

Detectando a vivacidade da face

O Amazon Rekognition Face Liveness ajuda você a verificar se um usuário que está passando pela verificação facial está fisicamente presente na frente de uma câmera. Ele detecta ataques falsos apresentados a uma câmera ou tentando contornar uma câmera. Os usuários podem concluir uma verificação de Face Liveness tirando uma pequena selfie em vídeo, seguindo uma série de instruções destinadas a verificar sua presença.

A vivacidade facial é determinada com um cálculo probabilístico e, em seguida, uma pontuação de confiança (entre 0 e 100) é retornada após a verificação. Quanto maior a pontuação, maior a confiança de que a pessoa que recebe o cheque está viva. O Face Liveness também retorna uma moldura, chamada de imagem de referência, que pode ser usada para comparação e pesquisa de faces. Como acontece com qualquer sistema baseado em probabilidade, o Face Liveness não pode garantir resultados perfeitos. Use-o com outros fatores para tomar uma decisão baseada em riscos sobre a identidade pessoal dos usuários.

O Face Liveness usa vários componentes:

- AWS Amplifique o SDK (React, Swift (iOS) e Android) com componente FaceLivenessDetector
- AWS SDKs
- AWS Nuvem APIs

Quando você configura seu aplicativo para se integrar ao recurso Face Liveness, ele usa as seguintes operações de API:

- CreateFaceLivenessSession- Inicia uma sessão de Face Liveness, permitindo que o modelo de detecção de Face Liveness seja usado em seu aplicativo. Retorna um SessionId para a sessão criada. Também permite que você defina seu ChallengePrefrence, para que você possa usar a FaceMovementChallenge opção.
- StartFaceLivenessSession- Chamado pelo AWS Amplify FaceLivenessDetector. Inicia um fluxo de eventos contendo informações sobre eventos e atributos relevantes na sessão atual.
- GetFaceLivenessSessionResults- Recupera os resultados de uma sessão específica do Face Liveness, incluindo uma pontuação de confiança do Face Liveness, imagem de referência e imagens de auditoria.

Você usará o SDK do AWS Amplify para integrar o recurso Face Liveness aos seus fluxos de trabalho de verificação baseados em face para aplicativos da web. Quando os usuários embarcarem ou se autenticarem por meio de seu aplicativo, envie-os para o fluxo de trabalho de verificação do Face Liveness no SDK do Amplify. O SDK do Amplify gerencia a interface do usuário e o feedback em tempo real dos usuários enquanto eles capturam sua selfie em vídeo.

Ao usar FaceMovementAndLightChallenge o rosto do usuário em direção ao oval exibido em seu dispositivo, o SDK do Amplify exibe uma sequência de luzes coloridas na tela. Em seguida, ele transmite com segurança o vídeo da selfie para a nuvem. APIs Como alternativa, ao usar FaceMovementChallenge, o rosto do usuário se move para o oval exibido no dispositivo, mas não há uma sequência de luzes coloridas. Embora 'FaceMovementAndLightChallenge' continue sendo a melhor configuração para maximizar a precisão, 'FaceMovementChallenge' permite que os clientes priorizem a velocidade mais rápida das verificações de atividade em vez da precisão. Ao selecionar entre essas configurações, os clientes devem considerar seus requisitos de caso de uso, incluindo os tipos de ataque esperados, as taxas desejadas de aceitação falsa e rejeição e verificações adicionais, como geolocalização (por exemplo, com base em IP), códigos One Time Pass (OTPs) etc. Os clientes devem tomar essa decisão depois de testar o desempenho do Liveness com vários limites de pontuação de confiança em seu conteúdo específico. Além disso, com os dois tipos de verificação de atividade, os proprietários do aplicativo devem implementar controles para proteger o dispositivo do qual o stream de vídeo é enviado. Depois que a análise for concluída, você receberá o seguinte no back-end:

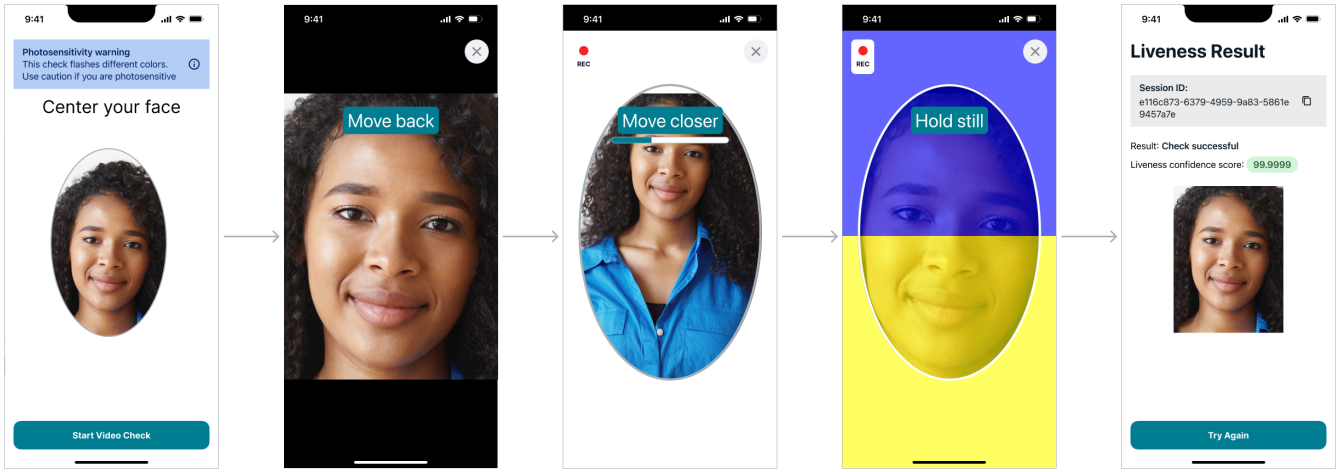
- Uma pontuação de confiança do Face Liveness (entre 0 e 100)
- Uma imagem de alta qualidade chamada imagem de referência que pode ser usada para correspondência facial ou pesquisa facial
- Um conjunto de até quatro imagens, chamadas de imagens de auditoria, selecionadas do vídeo selfie

O Face Liveness pode ser aproveitado para uma variedade de casos de uso. Por exemplo, o Face Liveness pode ser usado junto com a correspondência facial (com CompareFacese SearchFacesByImage) para verificação de identidade, para estimativa de idade em plataformas com restrição de acesso com base na idade e para detectar usuários humanos reais e, ao mesmo tempo, deter bots.

Você pode aprender mais sobre os casos de uso aos quais o serviço se destina, como o machine learning (ML) é usado pelo serviço e as principais considerações sobre o design e o uso responsáveis do serviço no cartão de serviço Rekognition Face Liveness AI.

Você pode definir limites para as pontuações de confiança de Face Liveness e Face Match. Os limites escolhidos devem refletir seu caso de uso. Em seguida, você envia uma verificação de identidade approval/denial ao usuário com base na pontuação acima ou abaixo dos limites. Se negado, peça ao usuário que tente novamente ou o envie para outro método.

O gráfico a seguir demonstra o fluxo do usuário, das instruções à verificação de vivacidade e ao resultado retornado:



Requisitos de vivacidade facial do lado do usuário

O Amazon Rekognition Face Liveness exige as seguintes especificações mínimas:

Dispositivos:

- O dispositivo deve ter uma câmera frontal
- Taxa mínima de atualização da tela do dispositivo: 60 Hz
- Tamanho mínimo de exibição ou tela: 4 polegadas
- O dispositivo não deve ser desbloqueado ou enraizado

Especificações da câmera:

- Câmera colorida: a câmera frontal deve ser capaz de gravar cores.
- Sem câmera virtual ou software de câmera.
- Capacidade mínima de gravação: 15 quadros por segundo.

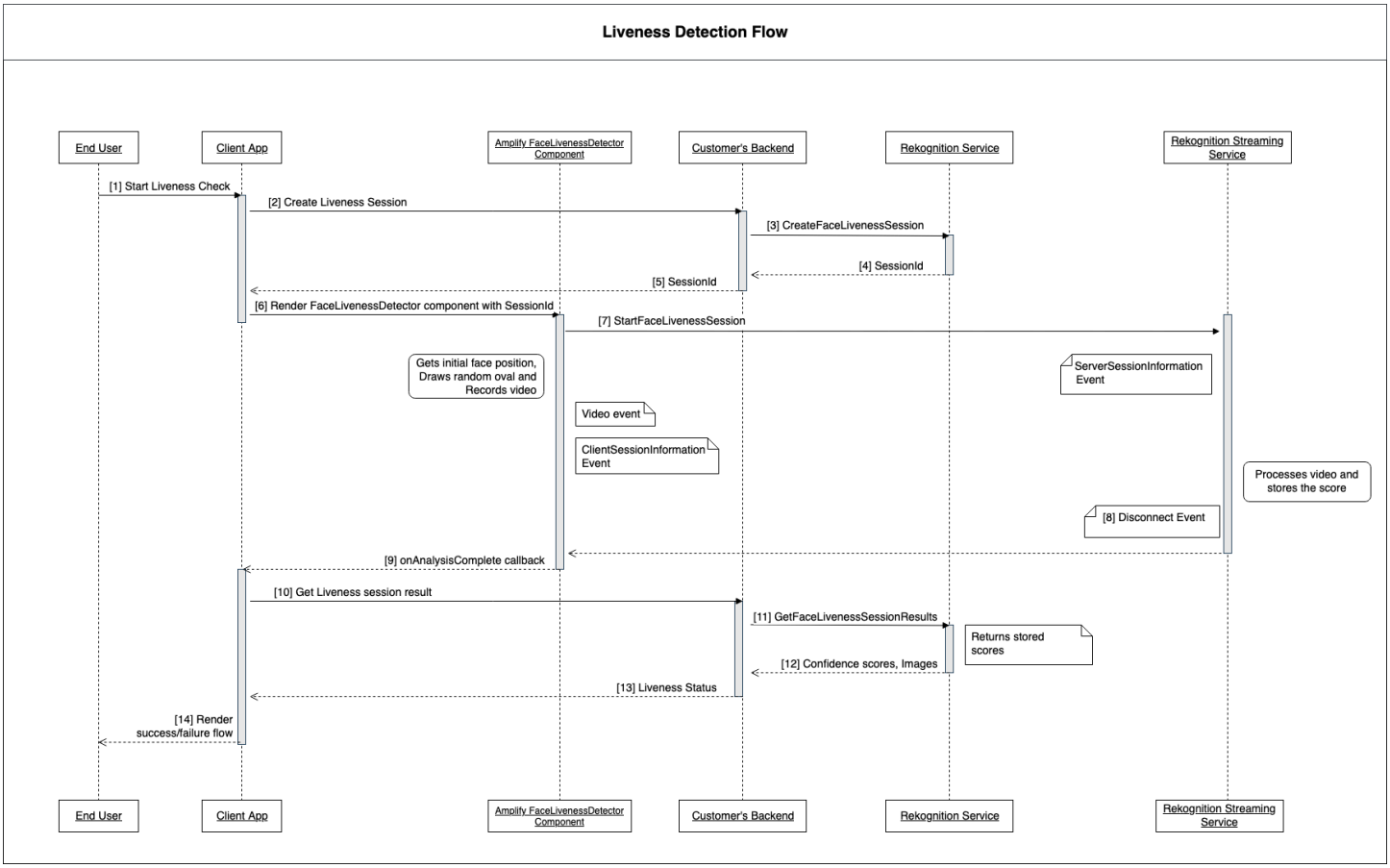
- Resolução mínima de gravação de vídeo: 480x640px.
- Quando os usuários usam uma webcam com um desktop para uma verificação de vivacidade facial, é importante montar a webcam em cima da mesma tela em que a verificação de vida facial começa.

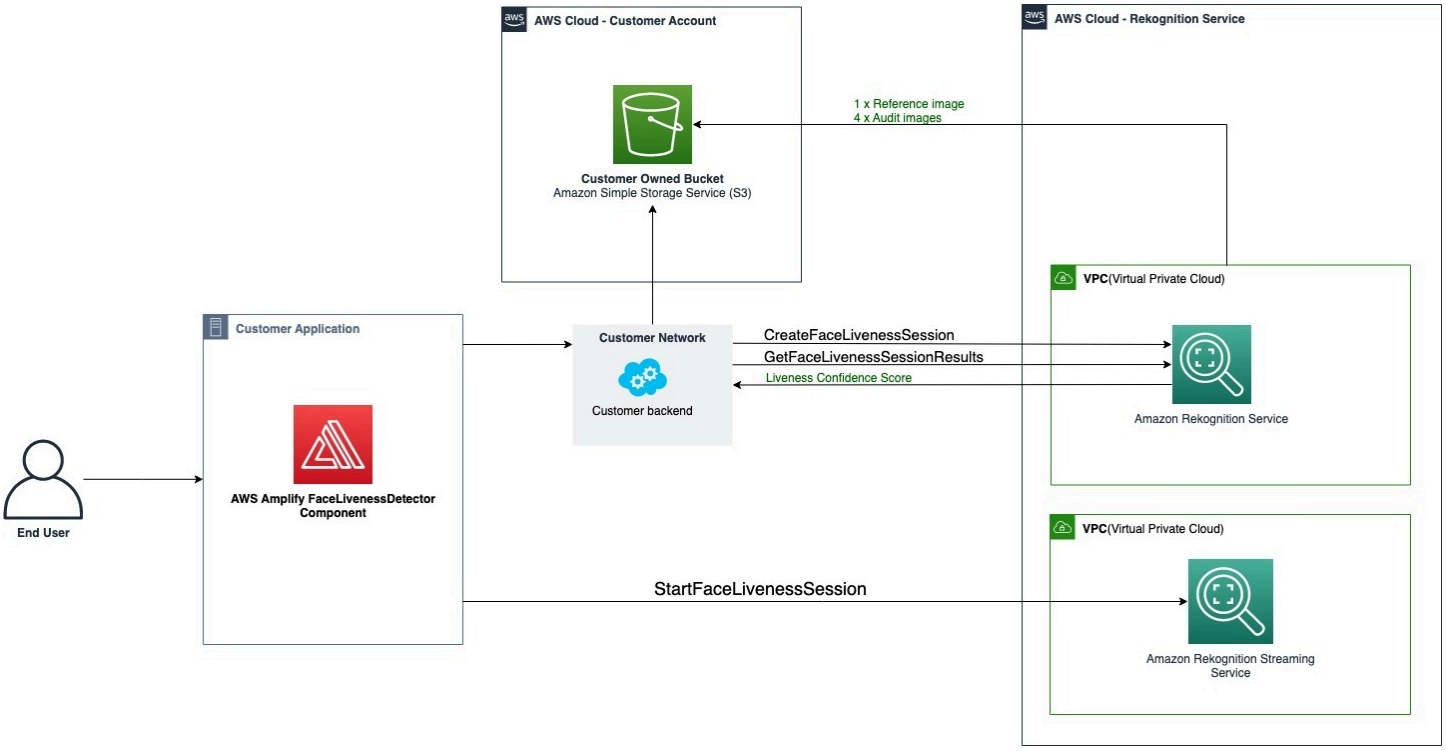
Requisito mínimo de largura de banda: 100 kbps

Navegadores compatíveis: as três versões mais recentes dos principais navegadores, como Google Chrome, Mozilla Firefox, Apple Safari e Microsoft Edge. Para obter mais informações sobre o suporte de navegadores, consulte Quais navegadores são compatíveis com o Console de Gerenciamento da AWS?

Diagramas de arquitetura e sequência

Os diagramas a seguir detalham como o Amazon Rekognition Face Liveness opera em relação à arquitetura e à sequência de operações do recurso:





O processo de verificação do Face Liveness envolve várias etapas, conforme descrito a seguir:

1. O usuário inicia uma verificação do Face Liveness no aplicativo cliente.
2. O aplicativo do cliente liga para o back-end do cliente, que por sua vez chama o serviço Amazon Rekognition. O serviço cria uma sessão de vivacidade facial e retorna uma sessão exclusiva SessionId. Nota: Depois que um SessionId é enviado, ele expira em 3 minutos, portanto, há apenas uma janela de 3 minutos para concluir as etapas 3 a 7 abaixo. Um novo SessionID deve ser usado para cada verificação de vivacidade facial. Se um determinado SessionID for usado para verificações subsequentes do Face Liveness, as verificações falharão. Além disso, a SessionId expira 3 minutos após o envio, tornando todos os dados do Liveness associados à sessão (por exemplo, ID da sessão, imagem de referência, imagens de auditoria etc.) indisponíveis.
3. O aplicativo cliente renderiza o componente FaceLivenessDetector Amplify usando os retornos de chamada SessionId obtidos e apropriados.
4. O FaceLivenessDetector componente estabelece uma conexão com o serviço de streaming Amazon Rekognition, renderiza um oval na tela do usuário e exibe uma sequência de luzes coloridas. FaceLivenessDetector grava e transmite vídeo em tempo real para o serviço de streaming Amazon Rekognition.

5. O serviço de streaming Amazon Rekognition processa o vídeo em tempo real, armazena os resultados e DisconnectEvent retorna a FaceLivenessDetector para o componente quando o streaming é concluído.
6. O FaceLivenessDetector componente chama o `onAnalysisComplete` retorno de chamada para sinalizar ao aplicativo cliente que o streaming foi concluído e que as pontuações estão prontas para recuperação.
7. O aplicativo cliente chama o back-end do cliente para obter um sinalizador booleano indicando se o usuário estava ativo ou não. O back-end do cliente faz a solicitação ao serviço Amazon Rekognition para obter a pontuação de confiança, a referência e as imagens de auditoria. O back-end do cliente usa esses atributos para determinar se o usuário está ativo e retorna uma resposta apropriada ao aplicativo do cliente.
8. Por fim, o aplicativo cliente passa a resposta para o FaceLivenessDetector componente, que renderiza adequadamente a success/failure mensagem para concluir o fluxo.

Pré-requisitos

Os pré-requisitos para usar o Amazon Rekognition Face Liveness incluem o seguinte:

1. Configurar uma AWS conta
2. Configure o Face Liveness AWS SDKs
3. Configurar recursos do AWS Amplify

Etapa 1: Configurar uma conta da AWS

Se você ainda não tiver uma AWS conta, conclua as etapas descritas em Crie uma AWS conta e um usuário para criar uma.

Etapa 2: configurar o Face Liveness AWS SDKs

Se você ainda não o fez, instale AWS CLI e configure o AWS SDKs e. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

Há várias maneiras de autenticar chamadas do AWS SDK. Os exemplos neste guia pressupõem que você esteja usando um perfil de credenciais padrão para chamar comandos da AWS CLI AWS e operações da API do SDK.

Consulte a página Como conceder acesso programático para obter mais informações sobre como conceder acesso à sua conta de usuário ao SDK escolhido. AWS A página também explica como usar um perfil em seu computador local e como executar o código de amostra em AWS ambientes.

Certifique-se de que o usuário que está chamando as operações do Face Liveness tenha as permissões corretas para chamar as operações, como as permissões `AmazonS3FullAccess` e `AmazonRekognitionFullAccess`.

Etapa 3: configurar os recursos do AWS Amplify

Para integrar o Amazon Rekognition Face Liveness ao seu aplicativo, você deve configurar o SDK do Amplify para usar o componente AWS Amplify. FaceLivenessDetector

Se ainda não o fez, siga as instruções para configurar a AWS Command Line Interface (AWS CLI) em Conceitos básicos da AWS CLI. Depois que a CLI for instalada, conclua as etapas de configuração da autenticação, vistas no site de documentos da interface do usuário do Amplify, para configurar seus recursos do Amplify. AWS

Melhores práticas para detectar a vivacidade facial

Recomendamos que você siga várias práticas recomendadas ao usar o Amazon Rekognition Face Liveness. As melhores práticas do Face Liveness incluem diretrizes sobre onde as verificações do Face Liveness devem ser conduzidas, o uso de imagens de auditoria e a escolha de limites de pontuação de confiança.

Consulte Recomendações para o uso do Face Liveness para ver a lista completa das melhores práticas.

Programando o Amazon Rekognition Face Liveness APIs

Para usar a API Amazon Rekognition Face Liveness, você deve criar um back-end que execute as seguintes etapas:

1. Ligue CreateFaceLivenessSessionpara iniciar uma sessão de Face Liveness. Quando a operação `CreateFaceLivenessSession` for concluída, a interface do usuário solicita que o usuário envie uma selfie em vídeo. O FaceLivenessDetector componente do AWS Amplify então liga StartFaceLivenessSessionpara realizar a detecção de Liveness.
2. Ligue GetFaceLivenessSessionResultspara retornar os resultados da detecção associados a uma sessão de Face Liveness.

3. Continue configurando seu aplicativo React para usar o FaceLivenessDetector componente seguindo as etapas no guia Amplify Liveness.

Antes de usar o Face Liveness, certifique-se de ter criado uma conta da AWS, configurado a AWS CLI e a AWS e configurado o SDKs AWS Amplify. Você também deve garantir que a política do IAM para sua API de back-end tenha permissões que abranjam o seguinte: `GetFaceLivenessSessionResults`, e `CreateFaceLivenessSession`. Consulte a seção Pré-requisitos para obter mais informações.

Etapa 1: CreateFaceLivenessSession

CreateFaceLivenessSession A operação da API cria uma sessão de Face Liveness e retorna uma única`SessionId`.

Como parte da entrada para essa operação, também é possível especificar uma localização de bucket do Amazon S3. Isso permite o armazenamento de uma imagem de referência e imagens de auditoria geradas durante a sessão Face Liveness. O bucket do Amazon S3 deve estar localizado na conta da AWS do chamador e na mesma região do endpoint Face Liveness. Além disso, as chaves de objeto do S3 são geradas pelo sistema Face Liveness.

Também é possível fornecer um `AuditImagesLimit`, que é um número entre 0 e 4. Por padrão, ele é definido como 0. O número de imagens retornadas é o melhor esforço e é baseado na duração do vídeo de selfie.

Por fim, você pode fornecer um ChallengePreference, incluindo o tipo e a versão do desafio que você deseja que sua sessão use como padrão. Por padrão, isso será usado FaceLivenessMovementAndLightChallenge, mas pode ser definido como FaceMovementChallenge.

Exemplo de solicitação

```
{
   "ClientRequestToken": "string",
   "KmsKeyId": "string",
   "Settings": {
      "AuditImagesLimit": number,
      "ChallengePreferences": [
         {
            "Type": "string",
            "Versions": {
```

```
                "Maximum": "string",
                "Minimum": "string"
             }
          }
       ],
       "OutputConfig": {
          "S3Bucket": "string",
          "S3KeyPrefix": "string"
       }
    }
}
```

Exemplo de resposta

```
{
    {"SessionId": "0f959dbb-37cc-45d8-a08d-dc42cce85fa8"}
}
```

Etapa 2: StartFaceLivenessSession

Quando a operação CreateFaceLivenessSession da API termina, o componente AWS Amplify executa a operação StartFaceLivenessSession da API. O usuário é solicitado a capturar uma selfie em vídeo. Para uma verificação bem-sucedida, o usuário deve posicionar a face dentro do oval na tela, mantendo uma boa iluminação. Para obter mais informações, consulte Recomendações para o uso do Face Liveness.

Essa operação de API requer o vídeo capturado durante a sessão do Face Liveness, o sessionID obtido da operação CreateFaceLivenessSession da API e um retorno de chamada. `onAnalysisComplete` O retorno de chamada pode ser usado para sinalizar o back-end para chamar a operação da GetFaceLivenessSessionResults API, que retorna uma pontuação de confiança, referência e imagens de auditoria.

Observe que essa etapa é executada pelo FaceLivenessDetector componente AWS Amplify no aplicativo cliente. Você não precisa fazer configurações adicionais para ligar `StartFaceLivenessSession`.

Etapa 3: GetFaceLivenessSessionResults

A operação GetFaceLivenessSessionResults da API recupera os resultados de uma sessão específica do Face Liveness. Ele requer o SessionID como entrada e retorna a pontuação de

confiança correspondente do Face Liveness. Ele também fornece uma imagem de referência que inclui uma caixa delimitadora de face e imagens de auditoria que também contêm caixas delimitadoras de face. A pontuação de confiança do Face Liveness varia de 0 a 100.

Exemplo de solicitação

```
{"SessionId": "0f959dbb-37cc-45d8-a08d-dc42cce85fa8"}
```

Exemplo de resposta

```
{
    "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"
}
```

Etapa 4: Responda aos resultados

Após a sessão Face Liveness, compare a pontuação de confiança da verificação com o limite especificado. Se a pontuação for maior que o limite, o usuário poderá ir para a próxima tela ou tarefa. Se a verificação falhar, o usuário será notificado e solicitado a tentar novamente.

Chamando o rosto de vivacidade APIs

Você pode testar o Amazon Rekognition Face Liveness com qualquer SDK compatível, como o AWS Python AWS SDK Boto3 ou o AWS SDK for Java. Você pode chamar o `CreateFaceLivenessSession` e `GetFaceLivenessSessionResults` APIs com o SDK escolhido. A seção a seguir demonstra como chamá-los APIs com o Python e o Java. SDKs

Para chamar o Face Liveness APIs:

- Se você ainda não o fez, crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess`. Para obter mais informações, consulte a Etapa 1: Configure uma conta da AWS e crie um usuário.
- Se você ainda não tiver feito isso, instale e configure a AWS CLI e a AWS. SDKs Para obter mais informações, consulte Etapa 2: Configurar a AWS CLI e a AWS. SDKs

Python

O trecho a seguir mostra como você pode chamá-los APIs em seus aplicativos Python. Observe que, para executar este exemplo, você precisará usar pelo menos a versão 1.26.110 do SDK do Boto3, embora a versão mais recente do SDK seja recomendada.

```
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

O trecho a seguir mostra como você pode chamá-los APIs em seus aplicativos 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

O trecho a seguir demonstra como chamar o Face Liveness APIs com o SDK Java V2: 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

O trecho a seguir demonstra como chamar o Face Liveness APIs com o SDK do Node.Js: 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)

O trecho a seguir demonstra como chamar o Face Liveness APIs com o SDK do AWS Node.Js para Javascript v3:

```
import { RekognitionClient, CreateFaceLivenessSessionCommand } from "@aws-sdk/
client-rekognition"; // ES Modules
import const { RekognitionClient, CreateFaceLivenessSessionCommand } =
 require("@aws-sdk/client-rekognition"); // CommonJS import
const client = new RekognitionClient(config);
const input = {
  KmsKeyId: "STRING_VALUE",
  Settings: {
    OutputConfig: { // LivenessOutputConfig
      S3Bucket: "STRING_VALUE", // required
      S3KeyPrefix: "STRING_VALUE",
    },
    AuditImagesLimit: Number("int"),
  },
  ClientRequestToken: "STRING_VALUE",
};
const command = new CreateFaceLivenessSessionCommand(input);
const response = await client.send(command);
// { // CreateFaceLivenessSessionResponse
//   SessionId: "STRING_VALUE", // required
// };
```

Configurando e personalizando seu aplicativo

Configurar seu aplicativo

Seu aplicativo Face Liveness pode operar em dispositivos móveis ou navegadores de desktop. Você desejará configurar os componentes do Face Liveness para se integrarem à solução escolhida. Você também deve garantir que seu aplicativo tenha permissão para usar a câmera de um dispositivo. O guia Amplify Liveness fornece instruções detalhadas sobre como:

- Instalar e configurar o AWS Amplify
- Importe e renderize o FaceLivenessDetector componente
- Ouvir os retornos de chamada
- Exemplo de mensagem de erro do Render Amplify

Personalize seu aplicativo

Você pode personalizar determinados componentes do seu aplicativo Liveness usando o AWS Amplify.

Para obter informações sobre tradução, consulte a documentação do Amplify Authenticator.

Para obter informações sobre como personalizar os componentes e temas do Amplify, consulte a documentação do Amplify sobre temas.

Modelo de responsabilidade compartilhada Face Liveness

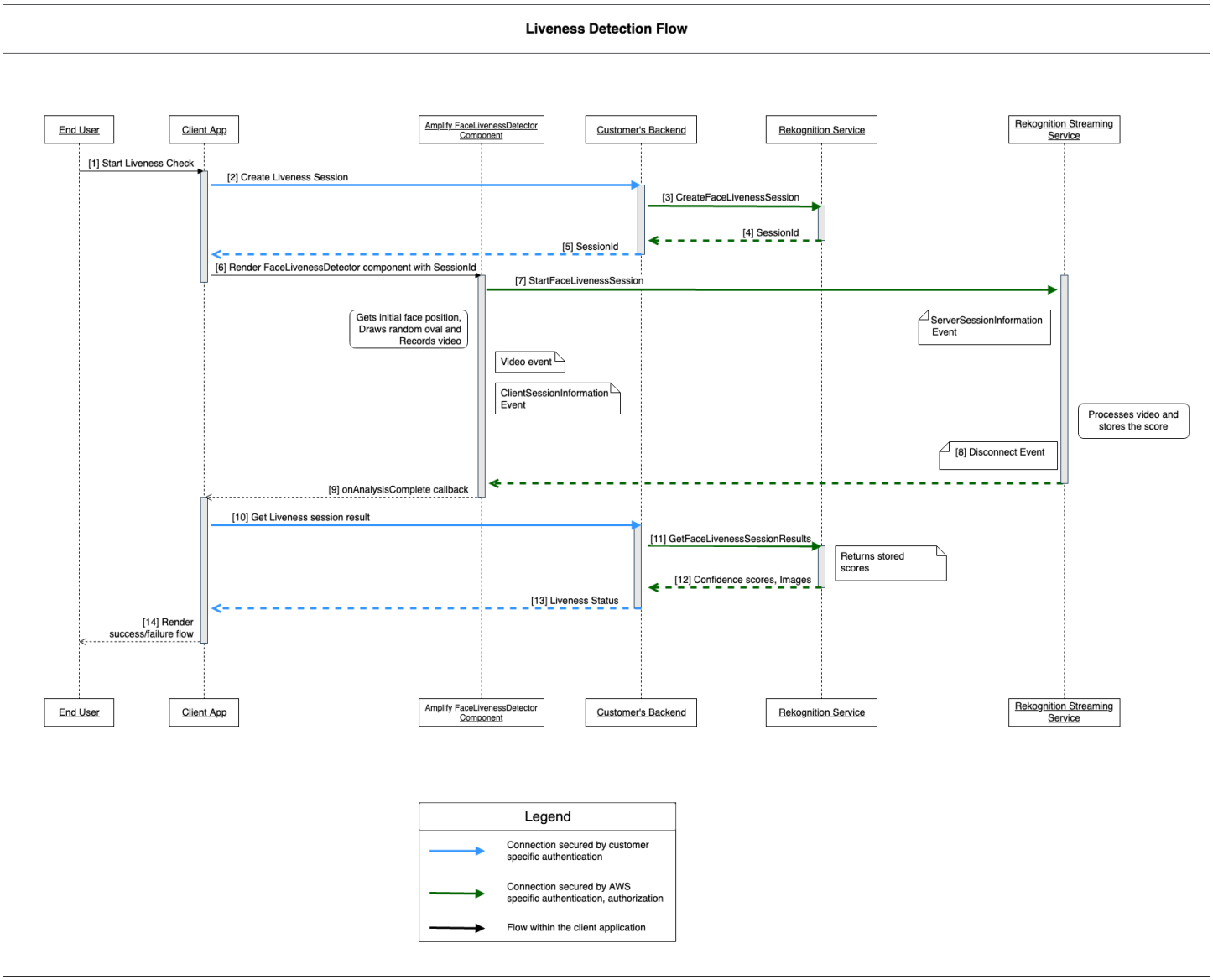
Segurança e conformidade são uma responsabilidade compartilhada entre você AWS e você, o cliente. Leia mais sobre o modelo de responsabilidade AWS compartilhada aqui.

1. Todas as chamadas para o AWS serviço (via aplicativo do cliente ou back-end do cliente) são autenticadas e autorizadas com AWS Auth (AWS Autenticação). É responsabilidade dos proprietários do serviço Face Liveness garantir que isso aconteça.
2. Todas as chamadas para o back-end do cliente (a partir do aplicativo do cliente) são autenticadas e autorizadas pelo cliente. Essa responsabilidade recai sobre o cliente. O cliente deve garantir que as chamadas do aplicativo cliente sejam autenticadas e não tenham sido manipuladas de forma alguma.
3. O back-end do cliente deve identificar o usuário final que está executando o desafio Face Liveness. É responsabilidade do cliente vincular o usuário final a uma sessão do Face Liveness. O serviço Face Liveness não faz distinção entre usuários finais. Ele só é capaz de identificar a AWS identidade da chamada (que o cliente manipula).
4. A AWS recomenda que os clientes apliquem verificações de validação adicionais, como localização geográfica (por exemplo, com base em IP), códigos One Time Pass (OTPs) etc. além do Face Liveness, que se alinha aos requisitos de seu caso de uso e postura de segurança.

A configuração 'FaceMovementAndLightChallenge' oferece a mais alta precisão para o Rekognition Liveness, exigindo que os usuários movam o rosto em direção à tela e fiquem parados por uma série de luzes piscando. Recomendamos que os clientes usem essa configuração padrão. Como alternativa, os clientes podem ativar a configuração FaceMovementChallenge '', que reduz o tempo de verificação em vários segundos ao eliminar as luzes piscantes. Embora 'FaceMovementAndLightChallenge' continue sendo a melhor configuração para maximizar a precisão, 'FaceMovementChallenge' permite que os clientes priorizem verificações de atividade mais

rápidas. Ao selecionar entre essas configurações, os clientes devem considerar seus requisitos de caso de uso, incluindo os tipos de ataque esperados, as taxas desejadas de aceitação falsa e rejeição, e também implementar verificações adicionais, como geolocalização (por exemplo, com base em IP), códigos One Time Pass (OTPs) etc. Os clientes devem tomar essa decisão depois de testar o desempenho do Liveness com vários limites de pontuação de confiança, dependendo do caso de uso. Os clientes são responsáveis por implementar controles para proteger o dispositivo do qual o vídeo é enviado

O diagrama de fluxo a seguir mostra quais chamadas são autenticadas pelo serviço da AWS ou pelo cliente:



Todas as chamadas para o serviço Amazon Rekognition Face Liveness são AWS protegidas pelo Auth (usando o mecanismo de assinatura). AWS Isso inclui as seguintes chamadas:

- [3] Chamada de CreateFaceLivenessSessionAPI (do back-end do cliente)
- [7] Chamada de StartFaceLivenessSessionAPI (do aplicativo cliente)
- [11] Chamada de GetFaceLivenessSessionResultsAPI (do back-end do cliente)

Todas as chamadas para o back-end do cliente precisam ter um mecanismo de autenticação e autorização. Os clientes precisam garantir que o usuário code/library/etc terceirizado esteja sendo mantido e desenvolvido ativamente. Os clientes também precisam garantir que o usuário final correto esteja fazendo chamadas para a sessão correta do Face Liveness. Os clientes devem autenticar e autorizar os seguintes fluxos:

- [2] Criar sessão Face Liveness (a partir do aplicativo cliente)
- [10] Obtenha o resultado da sessão Face Liveness (do aplicativo cliente)

Os clientes podem seguir o modelo de segurança STRIDE para garantir que suas chamadas de API estejam protegidas.

Tipo	Descrição	Controle de segurança
Falsificação	Ação de ameaça que visa acessar e usar as credenciais de outro usuário, como nome de usuário e senha.	Autenticação
Adulteração	Ação de ameaça com a intenção de alterar ou modificar dados persisten tes de forma maliciosa. Os exemplos incluem registros em um banco de dados e a alteração de dados em trânsito entre dois computadores em uma rede aberta, como a Internet.	Integridade
Repúdio	Ação de ameaça destinada a realizar operações proibidas	Não repúdio

	em um sistema que não tem a capacidade de rastrear as operações.	
Divulgação de informações	Ação de ameaça com a intenção de ler um arquivo ao qual não foi concedido acesso ou ler dados em trânsito.	Confidencialidade
Negação de serviço	Ação de ameaça que tenta negar acesso a usuários válidos, por exemplo, tornando um servidor web temporari amente indisponível ou inutilizável.	Disponibilidade
Elevação do privilégio	Ação de ameaça com a intenção de obter acesso privilegiado aos recursos para obter acesso não autorizado às informações ou compromet er um sistema.	Autorização

AWS protege suas conexões das seguintes maneiras:

1. Calcular a assinatura da solicitação e, em seguida, verificar a assinatura no lado do serviço. As solicitações são autenticadas usando essa assinatura.
2. AWS os clientes precisam configurar funções adequadas do IAM para autorizar determinadas ações/operações. Esses perfis do IAM são necessárias para fazer chamadas ao serviço da AWS.
3. Somente solicitações HTTPS para o AWS serviço são permitidas. As solicitações são criptografadas na rede aberta usando TLS. Isso protege a confidencialidade das solicitações e mantém a integridade da solicitação.
4. AWS o serviço registra dados suficientes para identificar as chamadas feitas pelos clientes. Isso evita ataques de repúdio .
5. AWS o serviço possui a manutenção de disponibilidade suficiente

O cliente é responsável por proteger seu serviço e suas chamadas de API das seguintes formas:

1. O cliente deve garantir que siga um mecanismo adequado de autenticação. Há vários mecanismos de autenticação que podem ser usados para autenticar uma solicitação. Os clientes podem explorar a autenticação baseada em resumos _OAuth , o OpenID Connect e outros mecanismos.
2. Os clientes devem garantir que seu serviço ofereça suporte aos canais de criptografia adequados (como TLS/HTTPS) para fazer chamadas de API de serviço.
3. Os clientes devem se certificar de registrar os dados necessários para identificar de forma exclusiva uma chamada de API e o chamador. Eles devem ser capazes de identificar o cliente que está chamando sua API com parâmetros definidos e a hora das chamadas.
4. Os clientes devem garantir que seu sistema esteja disponível e protegido contra ataques DDo S. Aqui estão alguns exemplos de técnicas de defesa contra ataques DDo S.

Os clientes são responsáveis por manter seus aplicativos up-to-date. Para obter mais informações, consulte Diretrizes de atualização do Face Liveness.

Diretrizes de atualização do Face Liveness

AWS atualiza regularmente o Face Liveness AWS SDKs (usado no back-end do cliente) e FaceLivenessDetector os componentes do AWS Amplify SDKs (usados em aplicativos do cliente) para fornecer novos recursos, segurança atualizada e aprimorada APIs, correções de bugs, melhorias de usabilidade e muito mais. Recomendamos que você mantenha o SDKs up-to-date para garantir o funcionamento ideal do recurso. Se você continuar usando versões mais antigas do SDKs, as solicitações poderão ser bloqueadas por motivos de manutenção e segurança.

O Face Liveness exige que você use o FaceLivenessDetector componente incluído no AWS SDKs Amplify (React, iOS, Android).

Versionamento e prazos

Estamos fazendo o versionamento dos seguintes componentes principais do recurso Face Liveness. Seguimos um formato de versionamento semântico. Por exemplo, um formato de versão de X.Y.Z em que X representa a versão principal, Y representa a versão secundária e Z representa a versão do patch.

- Os desafios do usuário do Face Liveness (por exemplo, FaceMovementAndLightChallenge desafio) fazem parte da API StartFaceLivenessSession
- FaceLivenessDetector componentes fornecidos por meio do AWS Amplify SDKs são usados em aplicativos clientes

Versões principais: Reservamos as principais atualizações de versão para atualizações críticas de segurança, API inovadora e usabilidade espetaculares. Os aplicativos e o back-end do cliente devem ser atualizados o mais rápido possível para que você continue usando os recursos do Face Liveness. Depois de lançarmos uma nova versão principal, oferecemos suporte à versão principal anterior por 120 dias a partir do dia da nova versão. Podemos bloquear as solicitações provenientes da versão principal anterior após 120 dias.

Versões secundárias: Reservamos pequenas atualizações de versão para recursos e melhorias importantes de segurança e usabilidade. É altamente recomendável aplicar essas atualizações. Embora nos esforcemos para garantir que pequenas atualizações sejam compatíveis com versões anteriores pelo maior tempo possível, podemos anunciar end-of-support uma versão secundária anterior 180 dias após o lançamento de uma nova versão secundária.

Versões de correção: Reservamos atualizações da versão do patch para correções e melhorias opcionais. Embora recomendemos que você mantenha sua versão up-to-date para obter a melhor segurança e experiência do usuário, nos esforçamos para garantir que as atualizações de patch sejam totalmente compatíveis com versões anteriores até lançarmos uma nova versão principal ou secundária.

A janela de tempo de versionamento (120 dias para versões principais e 180 dias para menores) se aplica à atualização do SDK em seu aplicativo, ao upload do aplicativo para a loja de aplicativos ou ao site e aos usuários que baixam a versão mais recente do aplicativo.

Lançamento da versão e matriz de compatibilidade

O lançamento de uma versão principal para desafio de FaceLivenessDetector componente ou usuário geralmente coincide. Para ajudar você a acompanhar as dependências de versão, consulte os recursos vinculados nas tabelas a seguir.

Versões do SDK e registros de alterações:

FaceLivenessDetector para web SDK	FaceLivenessDetector para iOS SDK	FaceLivenessDetector para Android SDK

Versão atual	Registro de alterações	Versão atual/Changelog	Versão atual/Changelog

Desafios do usuário:

Nome do desafio	Versão	Data de lançamento	Data de aposentadoria
FaceMovementAndLightChallenge	v1.0.0	4/10/2023	N/D
FaceMovementChallenge	v1.0.0	30/04/2025	N/D

Comunicação de novos lançamentos

AWS comunica novos lançamentos por meio dos seguintes canais:

- Notificações por e-mail de atualização da integridade do serviço enviadas para o e-mail da conta associado ao ID da conta Face Liveness.
- Atualizações publicadas AWS SDKs e notificações associadas nos respectivos GitHub repositórios.
- Atualizações publicadas para o AWS Amplify SDKs e notificações associadas nos respectivos GitHub repositórios.

Recomendamos que você se inscreva nesses canais para ficar up-to-date.

Perguntas frequentes sobre o Face Liveness

Use os itens de perguntas frequentes a seguir para encontrar respostas às perguntas mais frequentes sobre o Rekognition Face Liveness.

- Quais são os resultados de uma verificação de vivacidade facial?

  O Rekognition Face Liveness fornece os seguintes resultados para cada verificação de vivacidade:

  - Pontuação de confiança: Uma pontuação numérica que varia de 0 a 100 é retornada. Essa pontuação indica a probabilidade de o vídeo de selfie ser de uma pessoa real e não de um malfeitor usando paródia.
  - Imagem de alta qualidade: Uma única imagem de alta qualidade é extraída do vídeo selfie. Esse quadro pode ser utilizado para vários fins, como comparação facial, estimativa de idade ou pesquisa facial.
  - Imagens de auditoria: Até quatro imagens são retornadas do vídeo de selfie, que podem ser usadas para fins de trilha de auditoria.
- O Rekognition Face Liveness está em conformidade com os testes iBeta Presentation Attack Detection (PAD)?

  O teste de Detecção de Ataques de Apresentação (PAD) do iBeta Quality Assurance é conduzido de acordo com ISO/IEC 30107-3. O iBeta é credenciado pelo NIST/NVLAP para testar e fornecer resultados de acordo com esse padrão PAD. O Rekognition Face Liveness passou nos testes de conformidade iBeta Presentation Attack Detection (PAD) de nível 1 e 2 com uma pontuação PAD perfeita. O relatório pode ser encontrado na página da iBeta aqui.
- Como posso obter uma moldura de alta qualidade e molduras adicionais?

  O quadro de alta qualidade e os quadros adicionais podem ser retornados como bytes brutos ou carregados em um bucket do Amazon S3 que você especificar, dependendo das configurações da sua CreateFaceLivenessSessionsolicitação de API.
- Posso alterar a localização das luzes ovais e coloridas?

  Não. A localização oval e as luzes coloridas são recursos que permitem maior precisão e, portanto, não podem ser personalizados.
- Posso personalizar a interface do usuário de acordo com nosso aplicativo?

  Sim, você pode personalizar a maioria dos componentes da tela, como tema, cor, idioma, conteúdo de texto e fonte, para alinhá-los ao seu aplicativo. Detalhes sobre como personalizar esses componentes podem ser encontrados na documentação de nossos componentes de interface do usuário React, Swift e Android.
- Posso personalizar o tempo e o tempo da contagem regressiva para ajustar um rosto em formato oval?

Não, o tempo de contagem regressiva e o tempo de ajuste facial foram predeterminados com base em estudos internos em grande escala com milhares de usuários, com o objetivo de fornecer um equilíbrio ideal entre eles, permitindo maior precisão e latência. Por esse motivo, essas configurações de horário não podem ser personalizadas.

- As luzes de cores diferentes atendem às diretrizes de acessibilidade?

  Sim, as luzes de cores diferentes em nosso produto seguem as diretrizes de acessibilidade descritas nas WCAG 2.1. Conforme verificado em mais de 1000 verificações de usuários, a experiência do usuário exibe aproximadamente duas cores por segundo, o que está em conformidade com a recomendação de limitar as cores a três por segundo. Isso reduz a probabilidade de desencadear crises epilépticas na maioria da população.

- O SDK ajusta o brilho da tela para obter os melhores resultados?

  O celular Face Liveness SDKs (para Android e iOS) ajusta automaticamente o brilho quando a verificação é iniciada. No entanto, para o SDK da Web, há limitações nas páginas da Web que impedem o ajuste automático do brilho. Nesses casos, esperamos que o aplicativo web instrua os usuários finais a aumentarem manualmente o brilho da tela para obter os melhores resultados.

- Precisa ser oval? Poderíamos usar outras formas semelhantes?

  Não, o tamanho, a forma e a localização do oval não são personalizáveis. O design oval específico foi cuidadosamente escolhido por sua eficácia em capturar e analisar com precisão os movimentos faciais. Portanto, a forma oval não pode ser modificada.

- Qual é a end-to-end latência?

  Medimos a end-to-end latência desde o momento em que o usuário inicia a ação necessária para concluir a verificação de atividade até o momento em que o usuário obtém o resultado (aprovação ou falha). Na melhor das hipóteses, a latência é de 5 s. Em média, esperamos que seja cerca de 7 s. Na melhor das hipóteses, a latência é de 11 s. Vemos uma variação na end-to-end latência, pois ela depende: do tempo em que o usuário conclui a ação necessária (ou seja, mover o rosto para o oval), da conectividade de rede, da latência do aplicativo etc.

- Posso usar o recurso Face Liveness sem o SDK do Amplify?

  Não, o SDK do Amplify é necessário para usar o recurso Rekognition Face Liveness.

- Onde posso encontrar os estados de erro associados ao Face Liveness?

  Você pode ver os diferentes estados de erro do Face Liveness aqui.
- O Face Liveness não está disponível na minha região. Como posso usar o recurso?

  Você pode optar por ligar para o Face Liveness em qualquer uma das regiões onde ele estiver disponível, dependendo da carga de tráfego e da proximidade. Atualmente, a vivacidade facial está disponível nas seguintes AWS regiões:
  - Leste dos EUA (N. da Virgínia)
  - Oeste dos EUA (Oregon)
  - Europa (Irlanda)
  - Ásia-Pacífico (Tóquio, Mumbai)

  Mesmo que sua AWS conta esteja localizada em uma região diferente, não se espera que a diferença de latência seja significativa. Você pode obter molduras de selfie e imagens de auditoria de alta qualidade por meio da localização do Amazon S3 ou como bytes brutos, mas seu bucket do Amazon S3 deve corresponder à região AWS do Face Liveness. Se forem diferentes, você deverá receber as imagens como bytes brutos.
- O Amazon Rekognition Liveness Detection usa o conteúdo do cliente para melhorar o serviço?

  Você pode optar por não permitir que suas entradas de imagem e vídeo sejam usadas para melhorar ou desenvolver a qualidade do Rekognition e de outras tecnologias de machine learning/inteligência artificial da Amazon usando uma política de exclusão das Organizações da AWS . Para obter informações sobre como desativar, consulte a Política de exclusão do Managing AI Services.

Análise em massa

O Amazon Rekognition Bulk Analysis permite processar uma grande coleção de imagens de forma assíncrona usando um arquivo manifesto com a operação. StartMediaAnalysisJob A saída de cada imagem individual corresponde à saída retornada pela operação que você usa para análise.

Atualmente, o Rekognition suporta a análise com a operação. DetectModerationLabels

Você será cobrado pelo número de imagens que foram processadas com sucesso pelo trabalho. Os resultados de um trabalho concluído são enviados para um bucket do Amazon S3 especificado.

Observe que o Bulk Analysis não é compatível com a integração do Amazon A2I.

A API pode detectar tipos de conteúdo animado ou ilustrado, e as informações sobre o tipo de conteúdo detectado são retornadas como parte da resposta.

Tópicos

- Processamento de imagens em massa
- Análise em massa de manifestos de saída
- Tipo de conteúdo
- Verificar previsões e adaptadores de treinamento

Processamento de imagens em massa

Você pode iniciar um novo trabalho de análise em massa enviando um arquivo de manifesto e chamando a StartMediaAnalysisJob operação. O arquivo manifesto de entrada contém referências a imagens em um bucket do Amazon S3 e está formatado da seguinte forma:

```
{"source-ref": "s3://foo/bar/1.jpg"}
```

Criar uma tarefa de análise em massa (CLI)

1. Se ainda não tiver feito isso:

    a. Crie ou atualize um usuário com permissões `AmazonRekognitionFullAccess` e `AmazonS3ReadOnlyAccess`. Para obter mais informações, consulte Etapa 1: Configure uma conta da AWS e crie um usuário.

b. Instale e configure o AWS CLI e AWS SDKs o. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Faça upload de imagens para seu bucket S3.

   Para obter instruções, consulte Como fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

3. Use os comandos a seguir para criar e recuperar trabalhos de análise em massa.

CLI

Use o comando a seguir para chamar a StartMediaAnalysisJoboperação para análise com a DetectModerationLabels operação:

```
# 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"
```

Você pode obter informações sobre um determinado trabalho, como o caminho do Amazon S3 do bucket em que os resultados e os arquivos de resumo são armazenados, usando a GetMediaAnalysisJoboperação. Você fornece a ele uma ID de trabalho retornada por StartMediaAnalysisJob ou ListMediaAnalysisJob. Detalhes sobre trabalhos individuais são mantidos por apenas um ano.

```
# Request
aws rekognition get-media-analysis-job \
--job-id customer-job-id
```

Você pode listar todas as suas análises em massa usando a operação de ListMediaAnalysisJobstrabalho, que retorna páginas de trabalhos. Com o argumento `max-results`, você pode especificar o número máximo de trabalhos a serem retornados por página, limitado ao valor de `max-results`. No máximo 100 resultados são retornados por página. Detalhes sobre trabalhos individuais são mantidos por apenas um ano.

```
# Request
# Specify number of jobs to return per page, limited to max-results.
```

```
aws rekognition list-media-analysis-jobs --max-results 1
```

Análise em massa de manifestos de saída

O trabalho de análise em massa gera um arquivo manifesto de saída que contém os resultados do trabalho e um resumo do manifesto que contém estatísticas e detalhes sobre quaisquer erros ao processar as entradas do manifesto de entrada.

Se forem incluídas entradas duplicadas no manifesto de entrada, o trabalho não tentará filtrar entradas exclusivas e, em vez disso, processará todas as entradas fornecidas.

O arquivo manifesto de saída está formatado da seguinte forma:

```
// 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"}]}}
```

O resumo do manifesto de saída é formatado da seguinte forma:

```
{
    "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 de conteúdo

As informações sobre o tipo de conteúdo de mídia analisado pela StartMediaAnalysisJob operação são retornadas pela GetMediaAnalysisJob operação. ContentType pode ser uma das duas categorias diferentes:

- Conteúdo animado, que inclui videogame e animação (por exemplo, desenho animado, quadrinhos, mangá, anime).
- Conteúdo ilustrado, que inclui desenho, pintura e esboços.

Verificar previsões e adaptadores de treinamento

A análise em massa também pode ser utilizada por meio do console do Rekognition para obter previsões para um lote de imagens, verificar essas previsões e, em seguida, criar um adaptador usando as previsões verificadas. Os adaptadores permitem aumentar a precisão de qualquer operação compatível do Rekognition.

Atualmente, você pode criar adaptadores para usá-los com o recurso de moderação personalizada do Rekognition. Ao criar um adaptador e fornecê-lo para a DetectModerationLabelsoperação, você pode obter maior precisão nas tarefas de moderação de conteúdo relacionadas ao seu caso de uso específico.

Para obter mais informações sobre a moderação personalizada, consulte Aprimorando a precisão com moderação personalizada. Consulte Análise e verificação em massa para obter uma explicação sobre como verificar as previsões feitas com a análise em massa. Para ver um tutorial sobre como usar o console do Rekognition para verificar previsões e criar um adaptador, consulte Tutorial de adaptador de moderação personalizado.

Tutoriais

Esses tutoriais de vários serviços demonstram como usar as operações de API do Rekognition junto com outros AWS serviços para criar aplicativos de amostra e realizar uma variedade de tarefas. A maioria desses tutoriais usa o Amazon S3 para armazenar imagens ou vídeos. Outros serviços comumente usados incluem AWS Lambda.

Tópicos

- Armazenamento de dados do Amazon Rekognition com Amazon RDS e DynamoDB
- Usando o Amazon Rekognition e o Lambda para marcar ativos em um bucket do Amazon S3
- Criação AWS de aplicativos de análise de vídeo
- Criar uma função do Amazon Rekognition Lambda
- Como usar o Amazon Rekognition para verificação de identidade
- Usar o Amazon Rekognition (REK) para detectar rótulos para aplicativos de marketing
- Detectando rótulos em uma imagem usando Lambda e Python

Armazenamento de dados do Amazon Rekognition com Amazon RDS e DynamoDB

Ao usar o Amazon Rekognition APIs, é importante lembrar que as operações da API não salvam nenhum dos rótulos gerados. Você pode salvar esses rótulos colocando-os no banco de dados, junto com identificadores para as respectivas imagens.

Este tutorial demonstra a detecção de rótulos e o salvamento desses rótulos detectados em um banco de dados. O aplicativo de amostra desenvolvido neste tutorial lerá imagens de um bucket do Amazon S3, chamará a DetectLabelsoperação nessas imagens e armazenará os rótulos resultantes em um banco de dados. O aplicativo armazenará dados em uma instância de banco de dados do Amazon RDS ou em um banco de dados do DynamoDB, dependendo do tipo de banco de dados que você gostaria de usar.

Você usará o AWS SDK para Python ou este tutorial. Você também pode ver o GitHub repositório de exemplos do SDK de AWS documentação para ver mais tutoriais em Python.

Tópicos

- Pré-requisitos

- Obter rótulos para imagens em um bucket do Amazon S3
- Criação de uma tabela do Amazon DynamoDB
- Carregar dados para o DynamoDB
- Criação de um banco de dados MySQL no Amazon RDS
- Upload de dados para uma tabela MySQL do Amazon RDS

Pré-requisitos

Antes de começar este tutorial, você precisará instalar o Python e concluir as etapas necessárias para configurar o SDK do Python. AWS Além disso, confira se você:

Criou uma conta na AWS e um perfil do IAM

Instalou o SDK para Python (Boto3)

Configurou corretamente suas credenciais de AWS acesso

O bucket Amazon S3 criado o encheu de imagens

Criou uma instância de banco de dados RDS, se estiver usando o RDS para armazenar dados

Obter rótulos para imagens em um bucket do Amazon S3

Comece escrevendo uma função que pegue o nome de uma imagem em seu bucket do Amazon S3 e recupere essa imagem. Essa imagem será exibida para confirmar que as imagens corretas estão sendo passadas para uma chamada para a DetectLabelsqual também está na função.

1. Encontre o bucket do Amazon S3 que você gostaria de usar e anote seu nome. Você fará chamadas para esse bucket do Amazon S3 e lerá as imagens dentro dele. Certifique-se de que seu bucket contenha algumas imagens para passar para a DetectLabelsoperação.
2. Escreva o código para se conectar ao seu bucket Amazon S3. Você pode se conectar ao recurso Amazon S3 com o Boto3 para recuperar uma imagem de um bucket do Amazon S3. Uma vez conectado ao recurso Amazon S3, você pode acessar seu bucket fornecendo ao método Bucket o nome do seu bucket Amazon S3. Depois de se conectar ao bucket do Amazon S3, você recupera imagens do bucket usando o método Object. Ao usar o Matplotlib, você pode usar essa conexão para visualizar suas imagens à medida que elas são processadas. O Boto3 também é usado para se conectar ao cliente Rekognition.

No código a seguir, forneça sua região ao parâmetro region_name. Você passará o nome do bucket do Amazon S3 e o nome da imagem para DetectLabels, que retornará os rótulos da imagem correspondente. Depois de selecionar apenas os rótulos da resposta, o nome da imagem e os rótulos são retornados.

```
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. Salve esse código em um arquivo chamado get_images.py.

Criação de uma tabela do Amazon DynamoDB

O código a seguir usa o Boto3 para se conectar ao DynamoDB e usa o método `CreateTable` do DynamoDB para criar uma tabela chamada Images. A tabela tem uma chave primária composta que consiste em uma chave de partição chamada Imagem e uma chave de classificação chamada Rótulos. A chave Image contém o nome da imagem, enquanto a tecla Labels armazena os rótulos atribuídos a essa imagem.

```
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)
```

Salve esse código em um editor e execute-o uma vez para criar uma tabela do DynamoDB.

Carregar dados para o DynamoDB

Agora que o banco de dados do DynamoDB foi criado e você tem uma função para obter rótulos para imagens, você pode armazená-los no DynamoDB. O código a seguir recupera todas as imagens em um bucket do S3, obtém rótulos para elas e armazena os dados no DynamoDB.

1. Você precisará escrever o código para fazer o upload dos dados para o DynamoDB. Uma função chamada `get_image_names` é usada para se conectar ao seu bucket do Amazon S3 e retorna os nomes de todas as imagens no bucket como uma lista. Você passará essa lista para a função `read_image_from_S3`, que é importada do arquivo `get_images.py` que você criou.

   ```
   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. A função `read_image_from_S3` que criamos anteriormente retornará o nome da imagem que está sendo processada e o dicionário de rótulos associados a essa imagem. Uma função chamada `find_values` é usada para obter apenas os rótulos da resposta. O nome da imagem e seus rótulos estão então prontos para serem enviados para sua tabela do 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. Você usará uma terceira função, chame `load_data`, para realmente carregar as imagens e os rótulos na tabela do DynamoDB que você criou.

```
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. É aqui que as três funções que definimos anteriormente são chamadas e as operações são realizadas. Adicione as três funções definidas acima, junto com o código abaixo, a um arquivo Python. Execute o código.

```
bucket = "bucket_name"
file_list = get_image_names(bucket)

for file in file_list:
    file_name = file
    print("Getting labels for " + file_name)
    image_name, image_labels = read_image_from_s3(bucket, file_name)
    image_json_string = json.dumps(image_labels, indent=4)
    labels=set(find_values("Name", image_json_string))
    print("Labels found: " + str(labels))
    labels_dict = {}
    print("Saving label data to database")
    labels_dict["Image"] = str(image_name)
```

```
    labels_dict["Labels"] = str(labels)
    print(labels_dict)
    load_data(labels_dict)
    print("Success!")
```

Você acabou de gerar rótulos para suas imagens e armazená-los em uma instância do DynamoDB. DetectLabels Certifique-se de eliminar todos os recursos que você criou ao ler este tutorial. Isso evitará que você seja cobrado por recursos que não está usando.

Criação de um banco de dados MySQL no Amazon RDS

Antes de prosseguir, certifique-se de ter concluído o procedimento de configuração do Amazon RDS e criado uma instância de banco de dados MySQL usando o Amazon RDS.

O código a seguir usa a biblioteca PyMySQL e sua instância de banco de dados Amazon RDS. Ele cria uma tabela para conter os nomes de suas imagens e os rótulos associados a essas imagens. O Amazon RDS recebe comandos para criar tabelas e inserir dados nas tabelas. Para usar o Amazon RDS, você deve se conectar ao host do Amazon RDS usando seu nome de host, nome de usuário e senha. Você se conectará ao Amazon RDS fornecendo esses argumentos à `connect` função do PyMy SQL e criando uma instância de um cursor.

1. No código a seguir, substitua o valor do host pelo endpoint do host do Amazon RDS e substitua o valor do usuário pelo nome de usuário principal associado à sua instância do Amazon RDS. Você também precisará substituir a senha pela senha mestra do seu usuário principal.

   ```
   import pymysql

   host = "host-endpoint"
   user = "username"
   password = "master-password"
   ```

2. Crie um banco de dados e uma tabela para inserir seus dados de imagem e etiqueta. Faça isso executando e confirmando uma consulta de criação. O código a seguir cria um banco de dados. Execute esse código somente uma vez.

   ```
   conn = pymysql.connect(host=host, user=user, passwd=password)
   print(conn)
   cursor = conn.cursor()
   print("Connection successful")
   ```

```
# run once
create_query = "create database rekogDB1"
print("Creation successful!")
cursor.execute(create_query)
cursor.connection.commit()
```

3. Depois que o banco de dados for criado, você deverá criar uma tabela para inserir os nomes e rótulos das imagens. Para criar uma tabela, primeiro você passará o comando use SQL, junto com o nome do seu banco de dados, para a função `execute`. Depois que a conexão é feita, uma consulta para criar uma tabela é executada. O código a seguir se conecta ao banco de dados e cria uma tabela com uma chave primária, chamada `image_id`, e um atributo de texto armazenando os rótulos. Use as importações e variáveis que você definiu anteriormente e execute esse código para criar uma tabela em seu banco de dados.

```
# 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!")
```

Upload de dados para uma tabela MySQL do Amazon RDS

Depois de criar o banco de dados do Amazon RDS e uma tabela no banco de dados, você pode obter rótulos para suas imagens e armazená-los no banco de dados do Amazon RDS.

1. Conecte-se ao seu bucket do Amazon S3 e recupere os nomes de todas as imagens no bucket. Esses nomes de imagem serão passados para a função `read_image_from_s3` que você criou anteriormente para obter os rótulos de todas as suas imagens. O código a seguir se conecta ao seu bucket do Amazon S3 e retorna uma lista de todas as imagens em seu 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. A resposta da DetectLabelsAPI contém mais do que apenas os rótulos, então escreva uma função para extrair somente os valores dos rótulos. A função a seguir retorna uma lista cheia apenas dos rótulos.

```
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. Você precisará de uma função para inserir os nomes e rótulos das imagens em sua tabela. A função a seguir executa uma consulta de inserção e insere qualquer par de nomes e rótulos de imagem.

```
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. Finalmente, você deve executar as funções que você definiu acima. No código a seguir, os nomes de todas as imagens em seu bucket são coletados e fornecidos para a função que chama DetectLabels. Depois disso, os rótulos e o nome da imagem à qual eles se aplicam são enviados para seu banco de dados do Amazon RDS. Copie as três funções definidas acima com o código abaixo, em um arquivo Python. Execute o arquivo 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!")
```

Você usou com sucesso DetectLabels para gerar rótulos para suas imagens e armazenou esses rótulos em um banco de dados MySQL usando o Amazon RDS. Certifique-se de eliminar todos os recursos que você criou ao ler este tutorial. Isso evitará que você seja cobrado por recursos que não está usando.

Para ver mais exemplos AWS de vários serviços, consulte o repositório de exemplos GitHub do SDK de AWS documentação.

Usando o Amazon Rekognition e o Lambda para marcar ativos em um bucket do Amazon S3

Neste tutorial, você cria uma AWS Lambda função que marca automaticamente ativos digitais localizados em um bucket do Amazon S3. A função do Lambda lê todos os objetos em um determinado bucket do Amazon S3. Para cada objeto no bucket, ele passa a imagem ao serviço do

Amazon Rekognition para gerar uma série de rótulos. Cada rótulo é usado para criar uma tag que é aplicada à imagem. Depois de executar a função do Lambda, ela cria automaticamente tags com base em todas as imagens em um determinado bucket do Amazon S3 e as aplica às imagens.

Por exemplo, suponha que você execute a função do Lambda e tenha essa imagem em um bucket do Amazon S3.



Em seguida, o aplicativo cria automaticamente as tags e as aplica à imagem.

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**
>
> Os serviços que você usa neste tutorial fazem parte do nível AWS gratuito. Quando você terminar o tutorial, recomendamos encerrar todos os recursos que você criou durante o tutorial para que você não seja cobrado.

Este tutorial usa o AWS SDK for Java versão 2. Consulte o GitHub repositório de exemplos do SDK de AWS documentação para obter tutoriais adicionais do Java V2.

Tópicos

- Pré-requisitos
- Configurar o perfil IAM do Lambda
- Criar o projeto
- Escreva o código
- Empacote o projeto
- Implante a função do Lambda
- Teste o método Lambda

Pré-requisitos

Antes de começar, você precisa concluir as etapas em Configurar o AWS SDK for Java. Em seguida, verifique se você tem o seguinte:

- JDK Java 1.8.
- Maven 3.6 ou superior.
- Um bucket do Amazon S3 com 5 a 7 imagens da natureza. Essas imagens são lidas pela função do Lambda.

Configurar o perfil IAM do Lambda

Este tutorial usa os serviços Amazon Rekognition e Amazon S3. Configure a função do Lambda-support para ter políticas que permitam invocar esses serviços a partir de uma função do Lambda.

Para configurar a função

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Funções e, em seguida, escolha Criar função.
3. Escolha AWS serviço e, em seguida, escolha Lambda.
4. Escolha a aba Permissões.
5. Pesquise por AWSLambdaBasicExecutionRole.
6. Escolha Próximas tags.
7. Escolha Revisar.

8. Nomeie a função do Lambda-support.
9. Selecione Criar perfil.
10. Escolha lambda-support para visualizar a página de visão geral.
11. Escolha Anexar políticas.
12. AmazonRekognitionFullAccessEscolha na lista de políticas.
13. Escolha Anexar política.
14. Pesquise o AmazonS3 e escolha FullAccess Anexar política.

Criar o projeto

Crie um novo projeto Java e, em seguida, configure o pom.xml do Maven com as configurações e dependências necessárias. Certifique-se de que seu arquivo pom.xml tenha a seguinte aparência:

```
<?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>
```

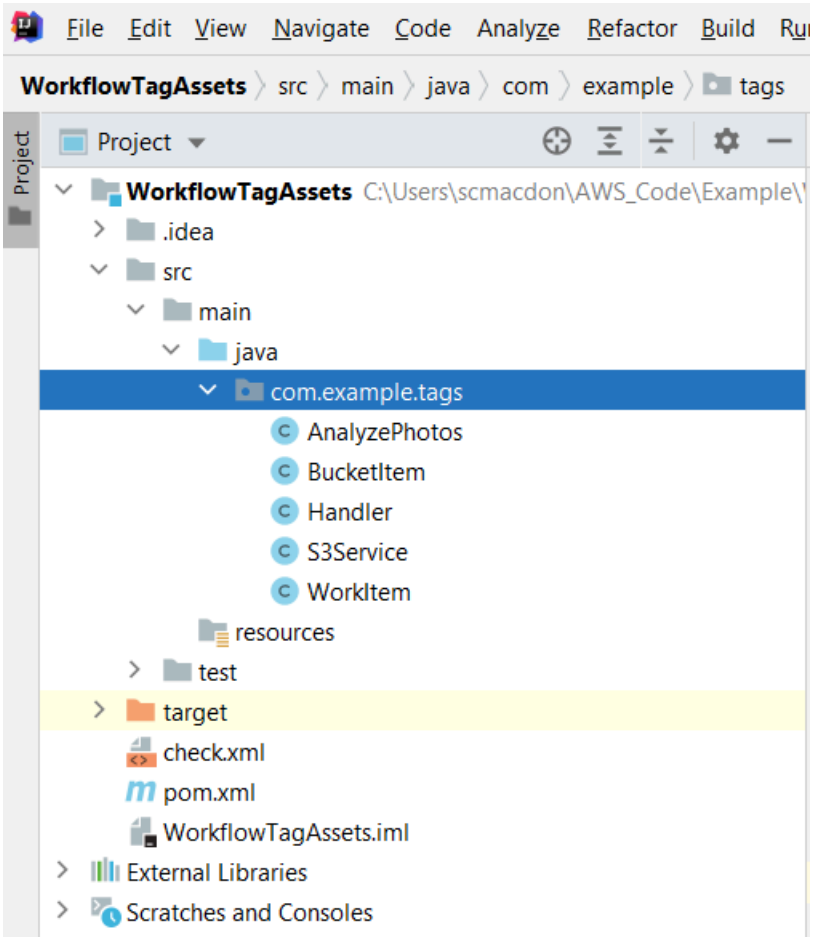
```
<dependencies>
   <dependency>
        <groupId>com.amazonaws</groupId>
        <artifactId>aws-lambda-java-core</artifactId>
        <version>1.2.1</version>
    </dependency>
    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
        <version>2.8.6</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>
```

Escreva o código

Use a API Java AWS Lambda de tempo de execução para criar a classe Java que define a função Lambda. Neste exemplo, há uma classe Java para a função do Lambda chamada Manipulador e classes adicionais necessárias para esse caso de uso. A figura a seguir mostra as classes Java no projeto. Observe que todas as classes Java estão localizadas em um pacote chamado com.example.tags.



Crie as seguintes classes Java para o código:

- O Handler usa a API de tempo de execução Lambda Java e executa o caso de uso descrito neste tutorial. AWS A lógica do aplicativo que é executada está localizada no método handleRequest.
- O S3Service usa a API Amazon S3 para realizar operações do S3.
- AnalyzePhotosusa a API Amazon Rekognition para analisar as imagens.
- BucketItemdefine um modelo que armazena informações do bucket do Amazon S3.

- WorkItemdefine um modelo que armazena dados do Amazon Rekognition.

Classe Handler

Esse código Java representa a classe Handler. A classe lê um sinalizador que é passado para a função do Lambda. O serviço S3. ListBucketObjectsO método retorna um objeto List em que cada elemento é um valor de string que representa a chave do objeto. Se o valor do sinalizador for verdadeiro, as tags serão aplicadas iterando pela lista e aplicando tags a cada objeto chamando o método s3Service.tagAssets . Se o valor do sinalizador for falso, então o S3Service. deleteTagFromÉ invocado um método de objeto que exclui as tags. Além disso, observe que você pode registrar mensagens CloudWatch nos registros da Amazon usando um LambdaLoggerobjeto.

> **Note**
>
> Certifique-se de atribuir o nome do seu bucket à variável 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

A classe a seguir usa a API do Amazon S3 para realizar operações do S3. Por exemplo, o getObjectBytesmétodo retorna uma matriz de bytes que representa a imagem. Da mesma forma, o listBucketObjectsmétodo retorna um objeto List em que cada elemento é um valor de string que especifica o nome da chave.

```
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

O código Java a seguir representa a AnalyzePhotosclasse. Essa classe usa a API Amazon Rekognition para analisar as imagens.

```
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

O código Java a seguir representa a BucketItemclasse que armazena dados de objetos do 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

O código Java a seguir representa a 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;
}

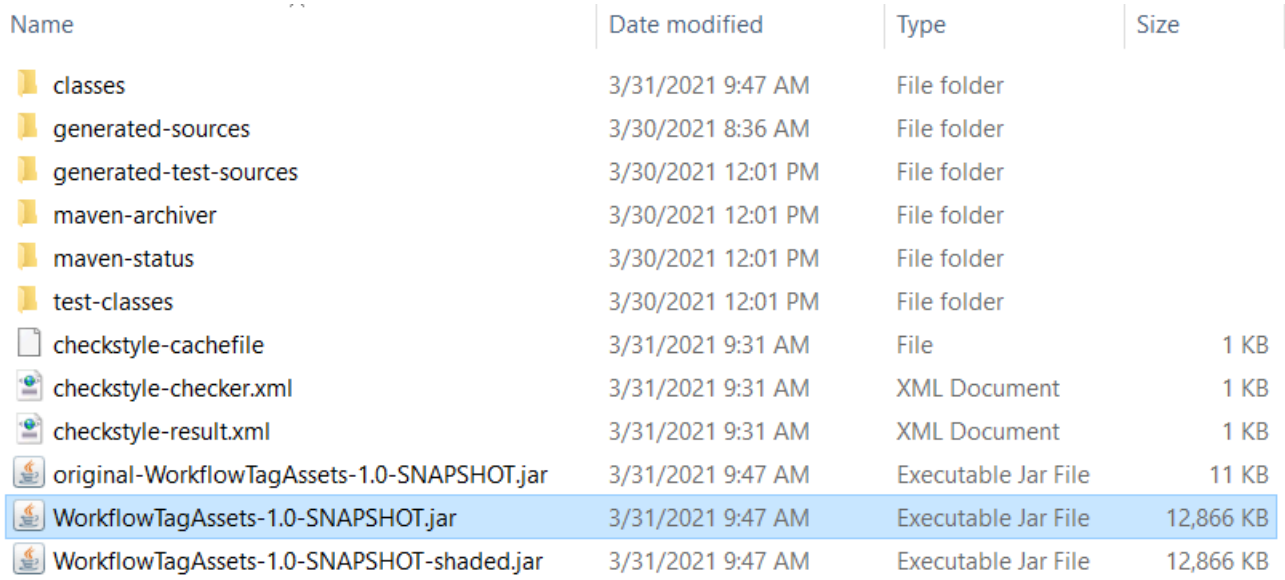
}
```

Empacote o projeto

Empacote o projeto em um arquivo.jar (JAR) usando o seguinte comando Maven.

```
mvn package
```

O arquivo JAR está localizado na pasta de destino (que é uma pasta secundária da pasta do projeto).



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**
>
> Observe o uso do maven-shade-pluginno arquivo POM do projeto. Esse plugin é responsável por criar um JAR que contém as dependências necessárias. Se você tentar empacotar o projeto sem esse plug-in, as dependências necessárias não serão incluídas no arquivo JAR e você encontrará um ClassNotFoundException.

Implante a função do Lambda

1. Abra o console do lambda.
2. Escolha Criar função.
3. Escolha Author from scratch (Criar do zero).
4. Na seção Informações básicas, insira cron como nome.
5. No Runtime, escolha Java 8.
6. Escolha Usar um perfil existente e, em seguida, escolha lambda-support (o perfil do IAM que você criou).
7. Escolha a opção Criar função.
8. Em Tipo de entrada de código, escolha Fazer upload de um arquivo.zip ou.jar.
9. Escolha Upload e, em seguida, navegue até o arquivo JAR que você criou.

10 Para Handler, digite o nome totalmente qualificado da função, por exemplo, com.example.tags.Handler:handleRequest (com.example.tags especifica o pacote, Handler é a classe seguida de :: e o nome do método).

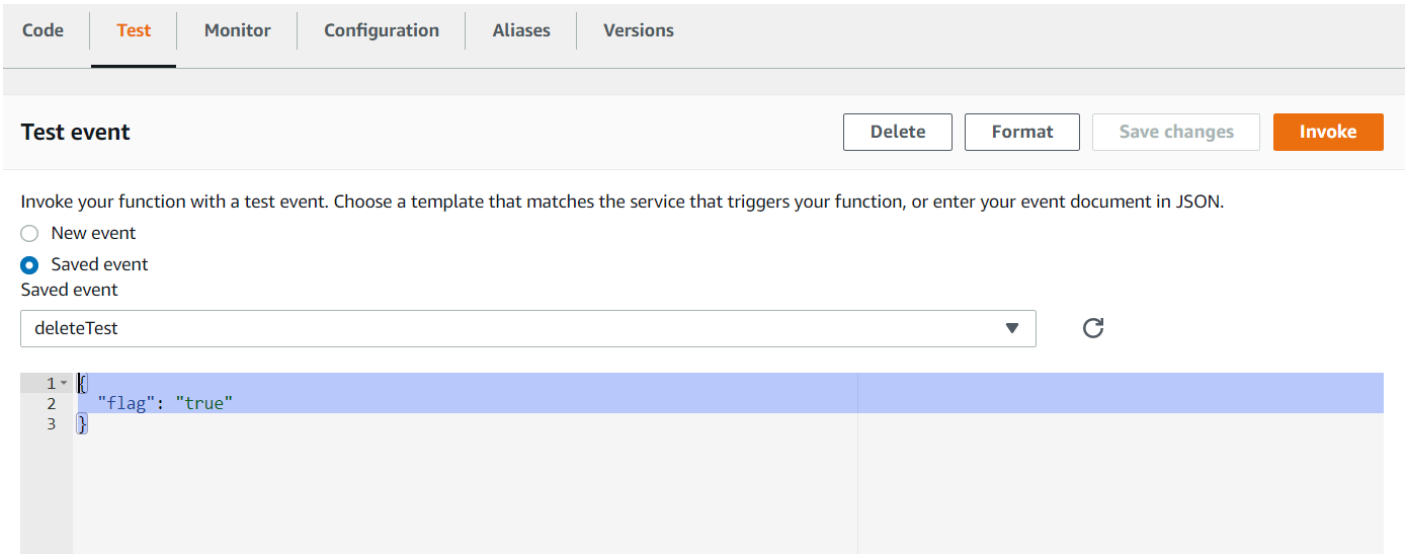
11 Escolha Salvar.

Teste o método Lambda

Neste ponto do tutorial, você pode testar a função do Lambda.

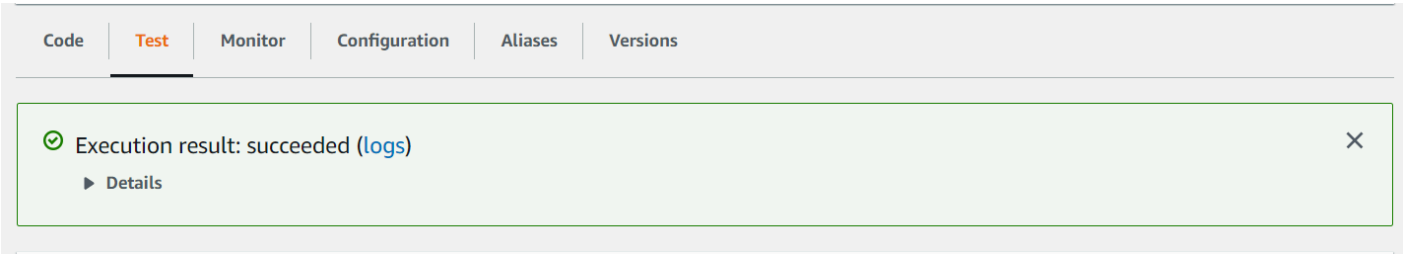
1. No console do Lambda, clique na guia Testar e insira o seguinte JSON.

```
                {
"flag": "true"
 }
```



> **Note**
>
> Passar tags verdadeiras para os ativos digitais e passar falsas exclui as tags.

2. Escolha o botão Invocar. Depois que a função do Lambda é invocada, você vê uma mensagem de sucesso.



Parabéns, você criou uma AWS Lambda função que aplica automaticamente tags a ativos digitais localizados em um bucket do Amazon S3. Conforme declarado no início deste tutorial, não se esqueça de encerrar todos os recursos que você criou ao passar por este tutorial para garantir que você não seja cobrado.

Para ver mais exemplos AWS de vários serviços, consulte o repositório de exemplos GitHub do SDK de AWS documentação.

Criação AWS de aplicativos de análise de vídeo

Você pode criar um aplicativo Web Java que analisa vídeos para detecção de rótulos usando o AWS SDK for Java versão 2. O aplicativo criado neste AWS tutorial permite que você envie um vídeo (MP4 arquivo) para um bucket do Amazon S3. Em seguida, o aplicativo usa o serviço Amazon Rekognition para analisar o vídeo. Os resultados são usados para preencher um modelo de dados e, em seguida, um relatório é gerado e enviado por e-mail para um usuário específico usando o Amazon Simple Email Service.

A ilustração a seguir mostra um relatório que é gerado após a conclusão da análise do vídeo pelo aplicativo. As colunas na tabela abaixo mostram faixa etária, barba, óculos e olhos abertos, bem como valores de confiança para diferentes previsões de atributos.

	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				

Neste tutorial, você cria um aplicativo Spring Boot que invoca vários AWS serviços. O Spring Boot APIs é usado para criar um modelo, visualizações diferentes e um controlador. Para obter mais informações, consulte Spring Boot.

Esse serviço usa os seguintes AWS serviços:

- Amazon Rekognition
- Amazon S3
- Amazon SES
- AWS Elastic Beanstalk

Os AWS serviços incluídos neste tutorial estão incluídos no nível AWS gratuito. Recomendamos que você encerre todos os recursos criados no tutorial ao terminar de usá-los para evitar cobranças.

Pré-requisitos

Antes de começar, você precisa concluir as etapas em Configurar o AWS SDK for Java. Em seguida, verifique se você tem o seguinte:

- JDK Java 1.8.
- Maven 3.6 ou superior.
- Um bucket do Amazon S3 chamado video[somevalue]. Certifique-se de usar esse nome de bucket em seu código Java do Amazon S3. Para mais informações, consulte Criar um bucket.
- Um perfil do IAM. Você precisa disso para a VideoDetectFacesclasse que você criará. Para obter mais informações, consulte Configurar o Amazon Rekognition Video.
- Um tópico válido do Amazon SNS. Você precisa disso para a VideoDetectFacesclasse que você criará. Para obter mais informações, consulte Configurar o Amazon Rekognition Video.

Procedimento

No decorrer do tutorial, você fará o seguinte:

1. Criar um projeto
2. Adicione as dependências POM ao seu projeto.
3. Crie as classes Java
4. Crie os arquivos HTML
5. Crie os arquivos de script
6. Empacote o projeto em um arquivo JAR
7. Implante o aplicativo em AWS Elastic Beanstalk

Para continuar com o tutorial, siga as instruções detalhadas no GitHub repositório de exemplos do SDK de AWS documentação.

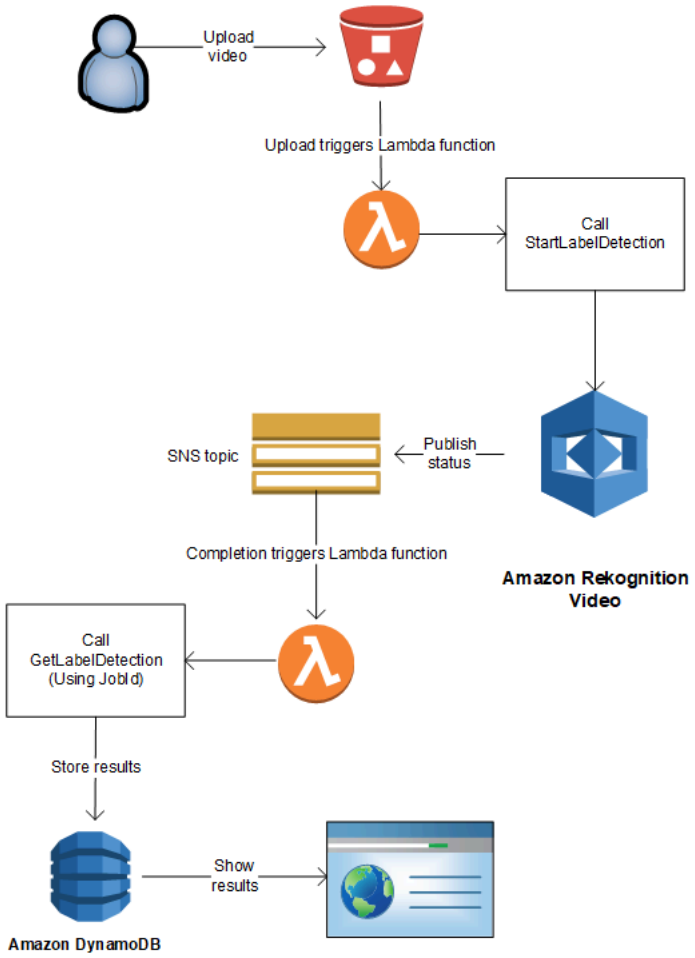
Criar uma função do Amazon Rekognition Lambda

Este tutorial mostra como obter os resultados de uma operação de análise de vídeo para detecção de rótulo usando uma função Java Lambda.

> **Note**
>
> Este tutorial usa o AWS SDK for Java 1.x. Para ver um tutorial usando o Rekognition e o AWS SDK for Java versão 2, consulte o repositório de exemplos do SDK de documentação.AWS GitHub

Você pode usar as funções do Lambda com as operações de vídeo do Amazon Rekognition Video. Por exemplo, o diagrama a seguir mostra um site que usa uma função do Lambda para iniciar automaticamente a análise de um vídeo durante o upload dele em um bucket do Amazon S3. Quando a função Lambda é acionada, ela chama StartLabelDetectionpara começar a detectar rótulos no vídeo enviado. Para obter informações sobre como usar o a fim de processar notificações de eventos de um bucket do Amazon S3, consulte Usar o AWS Lambda com eventos do Amazon S3.

Uma segunda função do Lambda é acionada quando o status da conclusão da análise é enviado ao tópico do Amazon SNS registrado. A segunda função Lambda é chamada GetLabelDetectionpara obter os resultados da análise. Os resultados são então armazenados em um banco de dados em preparação para exibição em uma página da web. Essa segunda função do Lambda é o foco deste tutorial.



Neste tutorial, a função do Lambda é acionada quando o Amazon Rekognition Video envia o status de conclusão da análise de vídeo para o tópico Amazon SNS registrado. Em seguida, ele coleta os resultados da análise do vídeo chamando GetLabelDetection. Para fins de demonstração, este tutorial grava os resultados da detecção de rótulos em um CloudWatch registro. Na função do Lambda do seu aplicativo, você deve armazenar os resultados da análise para uso posterior. Por exemplo, você pode usar o Amazon DynamoDB para salvar os resultados da análise. Para obter mais informações, consulte Como trabalhar com o DynamoDB.

Os procedimentos a seguir mostram como:

- Criar o tópico do Amazon SNS e configurar as permissões.
- Crie a função Lambda usando o tópico do Amazon SNS AWS Management Console e inscreva-a no tópico do Amazon SNS.
- Configurar a função do Lambda usando o AWS Management Console.
- Adicione um código de amostra a um AWS Toolkit for Eclipse projeto e faça o upload para a função Lambda.

- Teste a função do Lambda usando a AWS CLI.

> **Note**
>
> Use a mesma AWS região em todo o tutorial.

Pré-requisitos

Este tutorial pressupõe que você conhece o AWS Toolkit for Eclipse. Para obter mais informações, consulte AWS Toolkit for Eclipse.

Criar o tópico do SNS

O status de conclusão de uma operação de análise do Amazon Rekognition Video Video é enviado para um tópico do Amazon SNS. Esse procedimento cria o tópico do Amazon SNS e perfil de serviço IAM que dá ao Amazon Rekognition Video acesso aos seus tópicos do Amazon SNS. Para obter mais informações, consulte Chamando as operações de vídeo do Amazon Rekognition Video.

Para criar um tópico do Amazon SNS

1. Caso ainda não o tenha feito, crie um perfil de serviço IAM para conceder ao Amazon Rekognition Video acesso aos seus tópicos do Amazon SNS. Anote o nome de recurso da Amazon (ARN). Para obter mais informações, consulte Conceder acesso a vários tópicos do Amazon SNS.
2. Criar um tópico do Amazon SNS usando o console do Amazon SNS. Você só precisa especificar o nome do tópico. Prefixe o nome do tópico com. AmazonRekognition Anote o ARN do tópico.

Criar a função do Lambda

Você cria a função do Lambda usando o AWS Management Console. Em seguida, você usa um projeto AWS Toolkit for Eclipse para carregar o pacote da função do Lambda no AWS Lambda. Também é possível criar a função do Lambda com o AWS Toolkit for Eclipse. Para obter mais informações, consulte Tutorial: Como criar, fazer upload e invocar uma função do AWS Lambda.

Para criar a função do Lambda

1. Faça login no Console de Gerenciamento da AWS e abra o console do AWS Lambda em https://console.aws.amazon.com/lambda/.
2. Escolha Create function (Criar função).
3. Escolha Author from scratch (Criar do zero).
4. Em Function name (Nome da função), digite um nome para a função.
5. Em Runtime (Runtime), selecione Java 8.
6. Selecione Choose or create an execution role (Escolher ou criar uma função de execução).
7. Em Execution role (Função de execução), escolha Create a new role with basic Lambda permissions (Criar uma função com permissões básicas do Lambda).
8. Anote o nome da nova função que é exibido na parte inferior da seção Basic information (Informações básicas).
9. Escolha a opção Criar função.

Configurar a função do Lambda

Depois de criar a função do Lambda, configure-a para ser acionada pelo tópico do Amazon SNS criado em Criar o tópico do SNS. Ajuste também os requisitos de memória e o tempo limite da função do Lambda.

Configurar a função do Lambda

1. Em Function Code (Código da função), digite `com.amazonaws.lambda.demo.JobCompletionHandler` para o Handler (Manipulador).
2. Em Basic settings (Configurações básicas), selecione Edit (Editar). A caixa de diálogo Edit basic settings (Editar configurações básicas) é exibida.
   a. Selecione 1024 para Memory (Memória).
   b. Selecione 10 segundos para Timeout (Tempo limite).
   c. Escolha Salvar.
3. Em Designer, selecione + Add trigger (+ Adicionar trigger). A caixa de diálogo Add trigger (Adicionar trigger) é exibida.
4. Em Trigger configuration (Configuração de trigger), selecione SNS.

No tópico SNS, escolha o tópico do Amazon SNS que você criou em Criar o tópico do SNS.

5. Selecione Enable trigger (Habilitar trigger).
6. Para adicionar o trigger, selecione Add (Adicionar).
7. Escolha Salvar para salvar a função do Lambda.

Configurar o perfil IAM do Lambda

Para chamar as operações do Amazon Rekognition Video, você adiciona a política gerenciada da AWS à função AmazonRekognitionFullAccessdo IAM Lambda. Operações iniciais, como StartLabelDetection, também exigem permissões de função de aprovação para a função de serviço do IAM que o Amazon Rekognition Video usa para acessar o tópico do Amazon SNS.

Para configurar a função

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Perfis.
3. Na lista, selecione o nome da função de execução criada em Criar a função do Lambda.
4. Escolha a aba Permissões.
5. Escolha Anexar políticas.
6. AmazonRekognitionFullAccessEscolha na lista de políticas.
7. Escolha Anexar política.
8. Novamente, selecione a função de execução.
9. Escolha Add inline policy (Adicionar política em linha).
10. Selecione a guia JSON.
11. Substitua a política existente pela política a seguir. Substituir `servicerole` com o perfil de serviço do IAM que você criou em Criar o tópico do SNS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "mysid",
            "Effect": "Allow",
            "Action": "iam:PassRole",
```

```
            "Resource": "arn:servicerole"
        }
    ]
}
```

12. Selecione Revisar política.
13. Em Name* (Nome*), digite um nome para a política.
14. Selecione Criar política.

Crie o projeto AWS Toolkit for Eclipse Lambda

Quando a função Lambda é acionada, o código a seguir obtém o status de conclusão do tópico do Amazon SNS e GetLabelDetectionchama para obter os resultados da análise. Uma contagem de rótulos detectados e uma lista de rótulos detectados são gravados em um CloudWatch registro. Sua função do Lambda deve armazenar os resultados da análise de vídeo para uso posterior.

Para criar o projeto AWS Toolkit for Eclipse Lambda

1. Crie um projeto AWS Toolkit for EclipseAWS Lambda.
   - Em Project name: (Nome do projeto:), digite um nome de sua escolha para o projeto.
   - Em Nome da classe:, insira JobCompletionHandler.
   - Em Input type: (Tipo de entrada:), selecione SNS Event (Evento do SNS).
   - Deixe os outros campos inalterados.
2. No Eclipse Project Explorer, abra o método do manipulador Lambda gerado JobCompletionHandler (.java) e substitua o conteúdo pelo seguinte:

```
//Copyright 2018 Amazon.com, Inc. or its 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. Os namespaces do Rekognition não foram resolvidos. Para corrigir isso:
   - Pause o mouse sobre a parte sublinhada da linha `import com.amazonaws.services.rekognition.AmazonRekognition;`.
   - Selecione Fix project set up... (Corrigir a configuração do projeto...)
   - Escolha a versão mais recente do arquivo Amazon Rekognition.
   - Selecione OK para adicionar o arquivo morto ao projeto.
4. Salve o arquivo.
5. Clique com o botão direito na janela de código do Eclipse e selecione AWS Lambda e, então, Upload function to AWS Lambda.
6. Na página Select Target Lambda Function, escolha a região da AWS para usar.
7. Selecione Choose an existing lambda function (Selecionar um função existente do Lambda) e selecione a função do Lambda criada em Criar a função do Lambda.
8. Escolha Próximo. A caixa de diálogo Function Configuration (Configuração da função) é exibida.
9. Em IAM Role (Função do IAM), escolha a função do IAM criada em Criar a função do Lambda.
10. Escolha Concluir e a função do Lambda será enviada para AWS.

Testar a função do Lambda

Use o AWS CLI comando a seguir para testar a função Lambda iniciando a análise de detecção de rótulos de um vídeo. Após a conclusão da análise, a função do Lambda é acionada. Confirme se a análise foi bem-sucedida verificando os CloudWatch registros de registros.

Para testar a função do Lambda

1. Faça upload de um arquivo de vídeo no formato MOV ou MPEG-4 no bucket do S3. Para fins de teste, faça upload de um vídeo com até 30 segundos de duração.

   Para obter instruções, consulte Como fazer upload de objetos no Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

2. Execute o AWS CLI comando a seguir para começar a detectar rótulos em um vídeo.

   ```
   aws rekognition start-label-detection --video
    "S3Object={Bucket="bucketname",Name="videofile"}" \
   --notification-channel "SNSTopicArn=TopicARN,RoleArn=RoleARN" \
   --region Region
   ```

   Atualize os seguintes valores:

   - Mude `bucketname` e `videofile` para o nome do bucket do Amazon S3 e o nome do arquivo do vídeo no qual você deseja detectar rótulos.
   - Mude `TopicARN` para o ARN do tópico do Amazon SNS que você criou em Criar o tópico do SNS.
   - Mude `RoleARN` para o ARN do perfil do IAM que você criou em Criar o tópico do SNS.
   - `Region`Mude para a AWS região que você está usando.

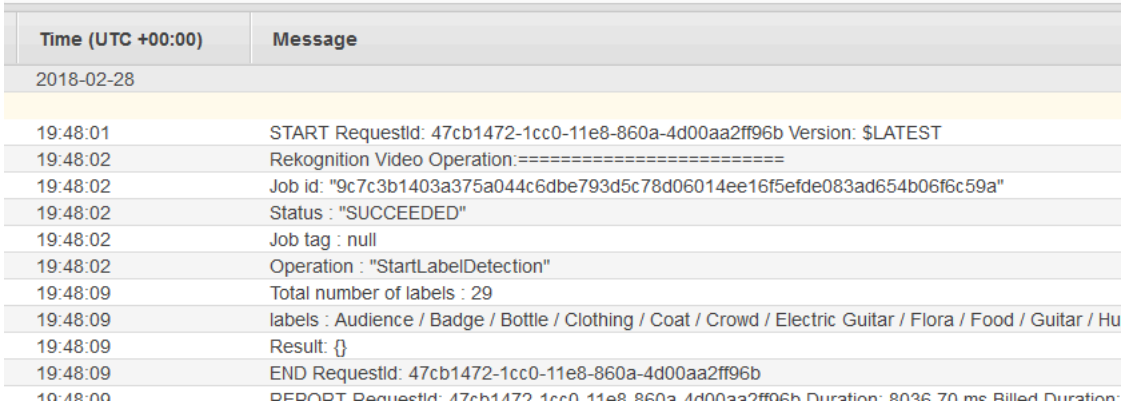
3. Anote o valor do `JobId` na resposta. A resposta é semelhante ao seguinte exemplo JSON.

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

4. Abra o console do https://console.aws.amazon.com/cloudwatch/.

5. Quando a análise é concluída, uma entrada de registro para a função do Lambda aparece no Grupo de logs.

6. Escolha a função do Lambda para ver os fluxos de log.
7. Selecione o fluxo de log mais recente para ver as entradas de log feitas pela função do Lambda. Se a operação for bem-sucedida, será semelhante à saída a seguir, que mostra os detalhes da operação de reconhecimento de vídeo, incluindo a ID do trabalho, o tipo de operação "StartLabelDetection“e uma lista das categorias de rótulos detectadas, como Garrafa, Roupa, Multidão e Comida:



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:

   O valor de Job id (ID do trabalho) deve corresponder ao valor de `JobId` anotado na etapa 3.

Como usar o Amazon Rekognition para verificação de identidade

O Amazon Rekognition oferece aos usuários várias operações que permitem a criação simples de sistemas de verificação de identidade. O Amazon Rekognition permite que o usuário detecte faces em uma imagem e compare as faces detectadas com outras, comparando dados de faces. Esses dados faciais são armazenados em contêineres ao lado do servidor chamados Coleções. Ao usar as operações de detecção facial, comparação facial e gerenciamento de coleções do Amazon Rekognition, você pode criar um aplicativo com uma solução de verificação de identidade.

Este tutorial demonstrará dois fluxos de trabalho comuns para a criação de aplicativos que exigem verificação de identidade.

O primeiro fluxo de trabalho envolve o registro de um novo usuário em uma coleção. O segundo fluxo de trabalho envolve a pesquisa em uma coleção que já existe para registrar um usuário retornado.

Você usará o AWS SDK para Python neste tutorial. Você também pode ver o GitHub repositório de exemplos do SDK de AWS documentação para ver mais tutoriais em Python.

Tópicos

- Pré-requisitos
- Criar uma coleção
- Registro do novo usuário
- Login de usuário existente

Pré-requisitos

Antes de começar este tutorial, você precisará instalar o Python e concluir as etapas necessárias para configurar o SDK do Python. AWS Além disso, confira se você:

- Criou uma conta na AWS e um perfil do IAM
- Instalou o SDK para Python (Boto3)
- Configure corretamente suas credenciais de AWS acesso
- Criou um bucket do Amazon Simple Storage Service e fez o upload da imagem que você quer usar como ID para a verificação de identidade.
- Selecionei uma segunda imagem como imagem de destino para a verificação de identidade.

Criar uma coleção

Antes de registrar ou pesquisar um usuário em uma Coleção, você precisa ter uma Coleção para trabalhar. Uma coleção do Amazon Rekognition é um contêiner do lado do servidor usado para armazenar informações sobre as faces detectadas.

Criar a coleção

Você começará escrevendo uma função que cria uma coleção para ser usada pelo seu aplicativo. O Amazon Rekognition armazena informações sobre as faces detectadas em contêineres do lado do servidor chamados Coleções. Você pode pesquisar informações armazenadas em uma Coleção para encontrar as faces conhecidas. Para armazenar informações de daces, primeiro é necessário criar uma coleção usando a operação `CreateCollection`.

1. Selecione um nome para a coleção que você deseja criar. No código a seguir, substitua o valor de `collection_id` pelo nome da coleção que você deseja criar e substitua o valor de `region` pelo nome da região definida nas suas credenciais de usuário. Você pode usar o argumento `Tags` para aplicar as tags que quiser à coleção, embora isso não seja obrigatório. A operação

`CreateCollection` retornará informações sobre a coleção que você criou, incluindo o ARN da coleção. Anote o Arn recebido como resultado da execução do código.

```
import boto3

def create_collection(collection_id, region):
    client = boto3.client('rekognition', region_name=region)

    # Create a collection
    print('Creating collection:' + collection_id)
    response = client.create_collection(CollectionId=collection_id,
    Tags={"SampleKey1":"SampleValue1"})
    print('Collection ARN: ' + response['CollectionArn'])
    print('Status code: ' + str(response['StatusCode']))
    print('Done...')

collection_id = 'collection-id-name'
region = "region-name"
create_collection(collection_id, region)
```

2. Salve e execute o código. Copie o Arn da coleção.

   Agora que a Coleção do Rekognition foi criada, você pode armazenar informações e identificadores faciais nessa Coleção. Você também poderá comparar as faces com as informações armazenadas para verificação.

Registro do novo usuário

Agora, você pode registrar novos usuários e adicionar suas informações a uma coleção. O processo de registro de um novo usuário normalmente envolve as seguintes etapas:

Chame a operação **`DetectFaces`**

Escreva o código para verificar a qualidade da imagem da face por meio da operação `DetectFaces`. Você usará a operação `DetectFaces` para determinar se uma imagem capturada pela câmera é adequada para ser processada pela operação `SearchFacesByImage`. A imagem deve conter apenas uma face. Você fornecerá um arquivo de imagem de entrada local para a operação `DetectFaces` e receberá detalhes das faces detectadas na imagem. O código de exemplo a seguir indica a imagem de entrada para `DetectFaces` e, em seguida, verifica se apenas uma face foi detectada na imagem.

1. No exemplo de código a seguir, substitua `photo` pelo nome da imagem de destino que detectará as faces. Você também precisará substituir o valor de `region` pelo nome da região associada à sua conta.

```
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. Salve o código e o execute.

Chame a operação **`CompareFaces`**

Seu aplicativo precisará registrar novos usuários em uma coleção e confirmar a identidade dos usuários retornados. Primeiro, você criará as funções usadas para registrar um novo usuário. Você começará usando a `CompareFaces` operação para comparar uma input/target image of the user and a ID/stored imagem local. Se houver uma correspondência entre a face detectada nas duas imagens, você pode pesquisar na Coleção para ver se o usuário foi registrado com ela.

Comece escrevendo uma função que compara uma imagem de entrada com a imagem de ID que você armazenou em seu bucket do Amazon S3. No exemplo de código a seguir, você mesmo precisará fornecer a imagem de entrada, que deve ser capturada após o uso de alguma forma de detector de vivacidade. Você também precisará passar o nome de uma imagem armazenada no bucket do Amazon S3.

1. Substitua o valor de `bucket` pelo nome do bucket do Amazon S3 que contém o arquivo de origem. Você também precisará substituir o valor de `source_file` pelo nome da imagem de origem que você está usando. Substitua o valor de `target_file` pelo nome do arquivo de destino que você informou. Substitua o valor de `region` pelo nome de `region` definido nas credenciais de usuário.

   Você também deve especificar um nível mínimo de confiança na correspondência que é retornada na resposta, usando o argumento `similarityThreshold`. As faces detectadas só serão retornadas na matriz `FaceMatches` se a confiança estiver acima desse limite. Sua escolha de `similarityThreshold` deve refletir a natureza de seu caso de uso específico. Qualquer caso de uso que envolva aplicativos críticos de segurança deve usar 99 como o limite selecionado.

```
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. Salve o código e o execute.

   Você receberá um objeto de resposta contendo informações sobre a face correspondente e o nível de confiança.

Chame a operação **`SearchFacesByImage`**

Se o nível de confiança da operação `CompareFaces` estiver acima do `SimilarityThreshold` escolhido, você deve pesquisar em sua coleção um rosto que possa corresponder à imagem de entrada. Se uma correspondência for encontrada em sua coleção, isso significa que o usuário provavelmente já está registrado na Coleção e não há necessidade de registrar um novo usuário. Se não houver uma correspondência, você pode registrar o novo usuário em sua Coleção.

1. Comece escrevendo o código que invocará a operação `SearchFacesByImage`. A operação receberá um arquivo de imagem local como argumento e, em seguida, pesquisará em `Collection` uma face que corresponda às principais faces detectadas na imagem fornecida.

   No exemplo de código a seguir, altere o valor de `collectionId` para a Coleção que você deseja pesquisar. Substitua o valor de `region` pelo nome da região associada à sua conta.

Você também precisará substituir o valor de `photo` pelo nome do seu arquivo de entrada. Especifique um limite de similaridade substituindo o valor de por uma porcentagem do `threshold` escolhido.

```
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. Salve o código e o execute. Se houver uma correspondência, isso significa que a pessoa reconhecida na imagem já faz parte da Coleção e não há necessidade de passar para as próximas etapas. Nesse caso, você pode simplesmente permitir que o usuário acesse o aplicativo.

Chame a operação **`IndexFaces`**

Supondo que nenhuma correspondência tenha sido encontrada na Coleção pesquisada, você precisará adicionar a face do usuário à sua coleção. Faça isso chamando a operação `IndexFaces`. Quando você liga IndexFaces, o Amazon Rekognition extrai as características faciais de um rosto identificado na sua imagem de entrada, armazenando os dados na coleção especificada.

1. Comece escrevendo o código para chamar a operação `IndexFaces`. Substitua o valor de `image` pelo nome do arquivo local que você deseja usar como imagem de entrada para a IndexFaces operação. Você também precisará substituir o valor de `photo_name` pelo nome do arquivo de entrada. Substitua o valor de `collection_id` pelo ID da coleção que você criou anteriormente. Em seguida, substitua o valor de `region` pelo nome da região associada à sua conta. Especifique um valor para o parâmetro de entrada `MaxFaces`, que define o número máximo de faces que devem ser indexadas em uma imagem. O valor padrão desse parâmetro é 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. Salve o código e o execute. Determine se você gostaria de salvar algum dos dados retornados pela operação `IndexFaces`, como o FaceID atribuído à pessoa na imagem. A próxima seção explicará como salvar esses dados. Antes de continuar, copie os valores `FaceId`, `ImageId` e `Confidence` retornados.

Armazene dados de imagem e FaceID no Amazon S3 e no Amazon DynamoDB

Após obter o Face ID para a imagem de entrada, os dados da imagem podem ser salvos no Amazon S3, enquanto os dados da face e o URL da imagem podem ser inseridos em um banco de dados como o DynamoDB.

1. Escreva o código para fazer upload da imagem de entrada para o banco de dados do Amazon S3. No exemplo de código a seguir, substitua o valor de `bucket` pelo nome do bucket em que será feito o upload do arquivo e, em seguida, substitua o valor de `file_name` pelo nome do arquivo local que armazenará o bucket do Amazon S3. Forneça um nome de chave que identificará o arquivo no bucket do Amazon S3 substituindo o valor de `key_name` por um nome que você gostaria de dar ao arquivo de imagem. O arquivo que você deseja carregar é o mesmo que foi definido nas amostras de código anteriores, que é o arquivo de entrada usado IndexFaces. Por fim, substitua o valor de `region` pelo nome da região associada à sua conta.

```
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. Salve e execute o exemplo de código a seguir para fazer upload da sua imagem de entrada no Amazon S3.

3. Salve o Face ID retornado em um banco de dados. Isso pode ser feito criando uma tabela de banco de dados DynamoDB e, em seguida, fazendo o upload do Face ID para essa tabela. O exemplo de código a seguir cria uma tabela do DynamoDB. Observe que você só precisa executar o código que cria essa tabela uma vez. No código a seguir, substitua o valor de `region` pelo valor da região associada à sua conta. Você também precisará substituir o valor de `database_name` pelo nome que gostaria de dar à tabela do 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. Salve e execute o código a seguir para criar sua tabela.

5. Depois de criar a tabela, você pode fazer o upload da tabela retornada FaceId para ela. Para fazer isso, você estabelecerá uma conexão com a tabela usando a função Table e, em seguida, usará a função `put_item` para fazer upload dos dados.

   No exemplo de código a seguir, substitua o valor de `bucket` pelo nome do bucket que contém a imagem de entrada que você fez upload no Amazon S3. Substitua o valor de `file_name` pelo nome do arquivo de entrada carregado no bucket do Amazon S3 e o valor de pela `key_name` chave usada anteriormente para identificar o arquivo de entrada. Por fim, substitua o valor de `region` pelo nome da região associada à sua conta. Esses valores devem corresponder aos informados na etapa 1.

   `AddDBEntry`Armazena os valores FaceId, ImageId, e de confiança atribuídos a um rosto em uma coleção. Informe os valores que você salvou na Etapa 2 da seção `IndexFaces` do processo na função abaixo.

```
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. Salve e execute o exemplo de código a seguir para armazenar os dados do Face ID retornados em uma tabela.

Login de usuário existente

Depois que um usuário é registrado em uma Coleção, é possível autenticá-lo usando a operação `SearchFacesByImage` quando ele retornar. Para isso, basta obter uma imagem de entrada e verificar a qualidade dela usando `DetectFaces`. Esse procedimento verifica se uma imagem adequada foi usada antes de executar a operação `SearchFacesbyImage`.

Ligue para a DetectFaces operação

1. Você usará a operação `DetectFaces` para definir se uma imagem capturada pela câmera é adequada para ser processada pela operação `SearchFacesByImage`. A imagem de entrada deve conter apenas uma face. O exemplo de código a seguir recebe uma imagem de entrada e a envia para a operação `DetectFaces`.

   No exemplo de código a seguir, substitua o valor de `photo` pelo nome da imagem de destino local e substitua o valor de `region` pelo nome da região associada à sua conta.

```
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. Salve e execute o código.

Ligue para a SearchFacesByImage operação

1. Escreva o código para comparar a face detectada com as faces da Coleção com `SearchFacesByImage`. A seguir, você usará o código mostrado na seção Registro de novo usuário, indicando a imagem de entrada para a operação `SearchFacesByImage`.

   No exemplo de código a seguir, altere o valor de `collectionId` para a Coleção que deverá ser pesquisada. Altere o valor de `bucket` para o nome de um bucket do Amazon S3 e o valor de `fileName` para um arquivo de imagem nesse bucket. Substitua o valor de `region` pelo nome da região associada à sua conta. Especifique um limite de similaridade substituindo o valor de por uma porcentagem do `threshold` escolhido.

   ```
   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. Salve e execute o código.

Verificar o FaceID retornado e o nível de confiança

Agora você pode verificar as informações sobre a correspondência FaceId imprimindo elementos de resposta FaceId, como os atributos Similaridade e Confiança.

```
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']))
```

Usar o Amazon Rekognition (REK) para detectar rótulos para aplicativos de marketing

Este tutorial orienta você na criação de uma aplicação do Python de amostra que pode ser usada para enviar e-mails para pessoas com base nas imagens enviadas para um site. A aplicação de amostra foi projetada para envolver os usuários em uma campanha de marketing em rede social enviando e-mails personalizados sobre ofertas de viagens caso pontos turísticos sejam reconhecidos em suas fotos.

A solução integra vários AWS serviços, incluindo Amazon Rekognition, Amazon S3, DynamoDB e Amazon SES. CloudWatch A aplicação usa o Amazon Rekognition para detectar rótulos em imagens carregadas por contas reconhecidas em um banco de dados do DynamoDB, então o titular da conta recebe um e-mail de marketing com base nos rótulos detectados na imagem. A arquitetura completa da solução é a seguinte:

- Armazenar dados do usuário no banco de dados do DynamoDB.
- O usuário carrega dados de imagem e metadados (número da conta do usuário) para o Amazon S3.
- Uma função Lambda é invocada DetectLabelspara identificar e registrar os rótulos na imagem carregada, pesquisar o e-mail do usuário no DynamoDB e enviar um e-mail de marketing para o usuário que fez o upload da imagem.
- Os resultados da função Lambda são registrados CloudWatch para análise posterior.

Veja uma visão geral de todas as etapas do tutorial:

1. Crie e preencha uma tabela do DynamoDB.
2. Escreva uma função do Lambda com registro e notificação.
3. Configure permissões e a função do Lambda.
4. Faça o upload de imagens e metadados com o Amazon S3.
5. CloudWatch Registros de pesquisas.

Tópicos

- Pré-requisitos
- Criação e preenchimento de uma tabela do DynamoDB
- Criar uma função do Lambda com registro em log e notificação.
- Configurar permissões e a função do Lambda.
- Fazer o upload de imagens e metadados com o Amazon S3.
- CloudWatch Registros de pesquisa do cliente

Pré-requisitos

Antes de começar este tutorial, você precisará de:

- Uma AWS conta e as permissões apropriadas do IAM.
- Python e Boto3 instalados em seu ambiente de desenvolvimento.
- Compreensão básica do Lambda, Amazon S3, DynamoDB, Amazon SES e. CloudWatch
- Familiaridade básica com o Lambda e o Amazon SES.

Criação e preenchimento de uma tabela do DynamoDB

Para começar este tutorial, criaremos uma tabela do DynamoDB para armazenar dados do cliente, como: endereço de e-mail, idade, número de telefone e status da associação, usando uma variável AccountNumber "" como chave primária. Também inseriremos alguns dados de amostra na tabela.

Criar a tabela do DynamoDB

Primeiro, criaremos uma tabela do DynamoDB. Veja como configurá-la usando Python e Boto3. Criamos uma função chamada `create_user_table` e conectamo-nos ao recurso do DynamoDB nela. No exemplo de código abaixo, substitua o valor de “region_name” pelo nome da região em que sua conta opera e execute a célula de código para criar sua tabela.

```
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()
```

A execução desse script configurou uma tabela do DynamoDB chamada `CustomerDataTable` com `AccountNumber` como chave primária.

Inserção de dados de amostra

Agora, o ideal é inserir alguns dados de amostra na tabela. Esses dados de amostra ajudarão a testar a funcionalidade completa do aplicativo.

Criaremos uma segunda função que adiciona dados de amostra à mesma `CustomerDataTable` que criamos antes. O código abaixo cria três exemplos de entradas, e cada entrada inclui um número de conta, endereço de e-mail, idade, número de telefone e status de membro. No exemplo de código abaixo, substitua o valor de `region_name` pelo nome da região em que sua conta opera e execute a célula de código para criar sua tabela. Se você quiser testar a parte de entrega de e-mail do aplicativo, substitua o valor `EmailAddress` na primeira entrada do cliente abaixo por um endereço de e-mail no qual você possa receber e-mails. Salve e execute o código.

```
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()
```

Com a tabela do DynamoDB configurada e preenchida, agora podemos continuar integrando essa recuperação de dados à função do Lambda. Isso permite que nosso aplicativo recupere os detalhes do usuário com base no número da conta que será identificado no próximo fluxo de trabalho de processamento de imagens.

Criar uma função do Lambda com registro em log e notificação.

Agora, podemos criar uma função do Lambda. Queremos garantir que a função do Lambda, que é acionada no upload da imagem, possa ler os dados da imagem e os metadados da conta e usá-los para realizar uma pesquisa na tabela do DynamoDB para os dados associados do usuário. Isso significa que não só precisaremos invocar a função do Amazon Rekognition, mas também de uma DetectLabels função que a receba AccountNumber e a use para recuperar o endereço de e-mail associado do DynamoDB. Além disso, precisaremos de uma função que envie um e-mail para o

endereço de e-mail usando o Amazon SES. Por fim, usaremos um registrador para registrar em log informações sobre o processo. Esses dados serão exibidos por CloudWatch.

Criar a função do Lambda

Aqui está um resumo de como uma função do Lambda pode lidar com esses requisitos. No exemplo de código a seguir, verifique se a tabela correta do DynamoDB está especificada, caso você tenha usado algo CustomerDataTable diferente do nome da tabela. Além disso, na função "send_marketing_email", você deve substituir o valor do argumento "Source" por um endereço de e-mail ao qual você tenha acesso e que funcionará como o e-mail de envio.

```
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")
```

Agora que escrevemos a função Lambda, precisamos configurar as permissões para o Lambda e criar uma instância da nossa função Lambda no Management Console. AWS

Configurar permissões e a função do Lambda.

Criar ou atualizar o perfil do IAM

Antes de usar uma função do Lambda para lidar com qualquer imagem enviada pelos usuários, é preciso lidar com as permissões para ela. A função Lambda precisa de uma função do IAM com escopo específico que tenha políticas que permitam a interação com o Amazon S3, o DynamoDB, o Amazon SES, o Amazon Rekognition e. CloudWatch

Para configurar o perfil do IAM para a função do Lambda:

1. Abra o console AWS de gerenciamento.
2. Acesse IAM > Funções > Criar função.

   Selecione “Criar função”.
3. Selecione Lambda como o serviço que usará essa função. Clique em “Avançar”.
4. Anexe as seguintes políticas (observe que essas políticas são selecionadas apenas para demonstração. Em uma configuração de produção real, o ideal é limitar as permissões somente às necessárias.):

   - Amazon S3 ReadOnlyAccess
   - AmazonRekognitionReadOnlyAccess
   - AmazonDynamoDBFullAcesso
   - Amazon SESFull Access
   - AWSLambdaExecutar
   - AWSLambdaBasicExecutionRole (para CloudWatch registro)

   Clique em “Avançar”.
5. Dê um nome à função e uma descrição, e crie-a escolhendo “Criar função”.
6. Depois de configurarmos as permissões adequadas para o Lambda, podemos criar uma instância da função Lambda usando o Management Console. AWS

   Vá para o serviço do Lambda no Console de Gerenciamento da AWS.
7. Clique em “Criar função”. Escolha “Criar do zero”.

8. Insira um nome de função e selecione o runtime do Python. Em “Alterar função de execução padrão”, selecione “Usar uma função existente” e escolha o perfil do IAM criado antes.
9. Agora você precisa criar e atualizar o código na guia da função do lambda. Vá até a guia chamada “lambda_function” e substitua o código lá pelo exemplo de código Lambda a seguir.

   Salve e implante as alterações.
10. Agora você precisa configurar um evento do Amazon S3 como o Gatilho do Lambda.

    Na guia/página de configuração da função do Lambda, acesse Gatilho e clique em “Adicionar gatilho”.
11. Selecione Amazon S3 na lista de gatilhos disponíveis.
12. Para configurar o gatilho: selecione o bucket do qual a função deve ser acionada.

    Escolha o tipo de evento, PUT. Opcionalmente, você pode especificar um prefixo ou sufixo se quiser processar apenas arquivos com determinados nomes ou tipos.
13. Ative o gatilho clicando em “Adicionar” e salve a configuração.

Verificar endereços de e-mail no Amazon SES

Antes de usar o Amazon SES para enviar e-mails, verifique os endereços de e-mail do remetente e do destinatário. Para fazer isso:

1. Vá para o console do Amazon SES. Navegue até “Gerenciamento de identidade” e depois até “Endereços de e-mail”.
2. Clique em “Verificar um novo endereço de e-mail”. Insira o endereço de e-mail que você quer verificar e clique em “Verificar este endereço de e-mail”. Você receberá um e-mail com um link de verificação. Clique no link para concluir o processo de verificação.
3. Depois de verificar os endereços de e-mail de ambas as contas, configurar sua função do Lambda com as permissões corretas e criar alguns exemplos de dados do cliente, você pode testar a função do Lambda fazendo o upload de uma imagem de teste para o bucket do Amazon S3 escolhido.

Fazer o upload de imagens e metadados com o Amazon S3.

Agora podemos fazer o upload de uma imagem de teste para o bucket Amazon S3 escolhido usando o AWS console ou o script que preparamos anteriormente. Escreva um script que carregue

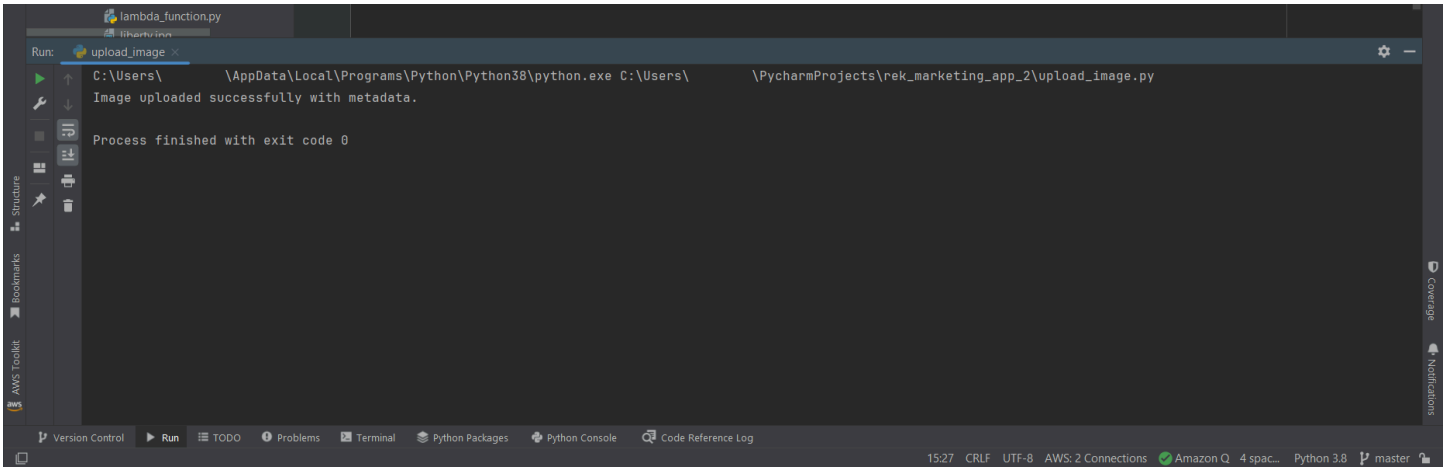
uma imagem no bucket do Amazon S3 que você especificou antes em sua função do Lambda.
No exemplo de código abaixo, especifique o caminho da sua imagem como o primeiro argumento para “upload_image_to_s3", o nome do bucket como o segundo argumento e o número da conta do usuário que está fazendo o upload da imagem como argumento final.

```
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')
```

Nessa função, o parâmetro `ExtraArgs` no método `upload_fileobj` é usado para incluir metadados definidos pelo usuário (`account_number`) junto com a imagem. Esses metadados podem ser usados posteriormente AWS para processar a imagem adequadamente.

Salve e execute o script: Isso fará o upload da imagem.



Alguns minutos após o upload da imagem, você deverá receber um e-mail no endereço já associado à conta especificada aqui.

CloudWatch Registros de pesquisa do cliente

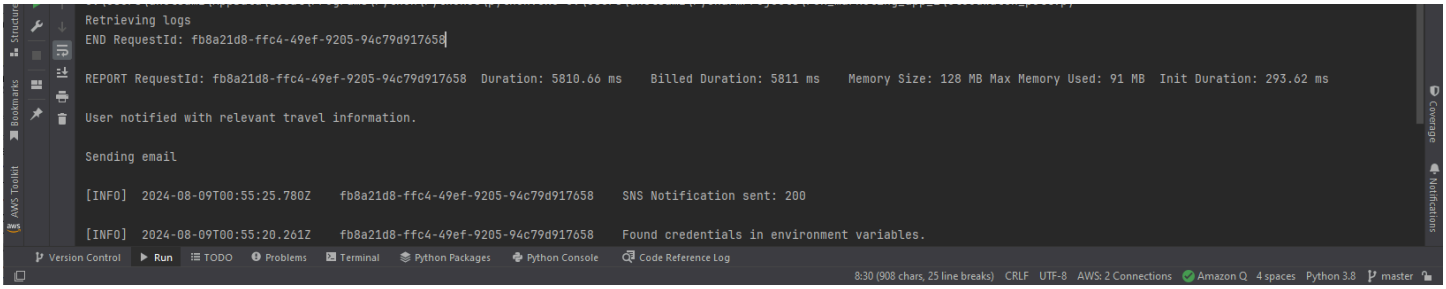
Verifique os CloudWatch registros da sua função Lambda para ver se ela foi acionada e executada conforme o esperado. Os registros podem ser encontrados em CloudWatch > Logs > Grupos de registros >/AWS/lambda/your_function_name. Você também pode escrever um script para acessar e imprimir os registros. O exemplo de código a seguir pesquisa o grupo Lambda em busca de logs, imprimindo os logs gerados na última hora. Salve e execute o código.

```
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'])
```

A execução do código deve imprimir os registros e você deve poder ver que o usuário foi notificado por meio de um e-mail contendo as informações de viagem relevantes.



Você criou um aplicativo capaz de detectar rótulos em imagens carregadas em um bucket do Amazon S3 e enviou uma mensagem promocional ao usuário que fez o upload da imagem. Se não precisar mais de algum recurso, lembre-se de excluí-lo.

Detectando rótulos em uma imagem usando Lambda e Python

AWS Lambda é um serviço de computação que você pode usar para executar código sem provisionar ou gerenciar servidores. Você pode chamar operações da API Rekognition de dentro de uma função do Lambda. As instruções a seguir mostram como criar uma função do Lambda em Python que chame `DetectLabels`.

A função do Lambda chama `DetectLabels` e retorna uma matriz de rótulos detectados na imagem, bem como o nível de confiança pelo qual eles foram detectados.

As instruções incluem um exemplo de código Python que mostra como chamar a função do Lambda e fornecer a ela uma imagem proveniente de um bucket do Amazon S3 ou do seu computador local.

Certifique-se de que as imagens escolhidas atendam aos limites do Rekognition. Consulte Diretrizes e cotas no Rekognition e na Referência da API para obter informações sobre DetectLabels os limites de tamanho e tipo de arquivo de imagem.

Crie uma função do Lambda (console)

Nesta etapa, você cria uma função do Lambda vazia e uma função de execução do IAM que permite que sua função do Lambda chame a `DetectLabels` operação. Nas etapas posteriores, você adiciona o código-fonte e, opcionalmente, adiciona uma camada à função do Lambda.

Se você estiver usando documentos armazenados em um bucket do Amazon S3, esta etapa também demonstra como conceder acesso ao bucket que armazena seus documentos.

Para criar uma AWS Lambda função (console)

1. Faça login no AWS Management Console e abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Escolha a opção Criar função. Para obter mais informações, consulte Criar uma função do Lambda com o console.
3. Escolha as seguintes opções:
   - Escolha Criar do zero.
   - Insira um valor para Nome da função.
   - Para Runtime, escolha a versão mais recente do Python.
   - Em Architecture (Arquitetura), escolha x86_64.
4. Escolha Criar função para criar a função do AWS Lambda .
5. Na página da função, escolha a guia Configuração.
6. No painel Permissões, em Função de execução, escolha o nome do perfil para abrir o perfil no console do IAM.
7. Na guia Permissões, escolha Adicionar permissões e, em seguida, Criar política em linha.
8. Escolha a guia JSON e substitua a política pela seguinte política:

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

```
}
```

9. Escolha Revisar política.
10. Insira um nome para a política, por exemplo DetectLabels-access.
11. Escolha Criar política.
12. Se você estiver armazenando documentos para análise em um bucket do Amazon S3, deverá adicionar uma política de acesso do Amazon S3. Para fazer isso, repita as etapas 7 a 11 no console da AWS Lambda e faça as seguintes alterações.

    a. Para a etapa 8, use a política a seguir. *bucket/folder path*Substitua pelo bucket e pelo caminho da pasta do Amazon S3 até os documentos que você deseja analisar.

    ```
    {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "S3Access",
                "Effect": "Allow",
                "Action": "s3:GetObject",
                "Resource": "arn:aws:s3:::bucket/folder path/*"
            }
        ]
    }
    ```

    b. Para a etapa 10, escolha um nome de política diferente, como S3Bucket-Access.

(Opcional) Crie uma camada (console)

Você não precisa executar essa etapa para usar uma função do Lambda e chamar `DetectLabels`.

A `DetectLabels` operação está incluída no ambiente padrão do Lambda Python como parte do AWS SDK for Python (Boto3).

Se outras partes da sua função do Lambda exigirem atualizações de AWS serviço recentes que não estejam no ambiente padrão do Lambda Python, você poderá realizar esta etapa para adicionar a versão mais recente do SDK do Boto3 como uma camada à sua função.

Para adicionar o SDK como uma camada, primeiro você cria um arquivo zip que contém o SDK do Boto3. Em seguida, você cria uma camada e adiciona o arquivo zip à camada. Para obter mais informações, consulte Usar camadas com sua função do Lambda.

Para criar e adicionar uma camada (console)

1. Abra um prompt de comando e insira os comandos a seguir para criar um pacote de implantação com a versão mais recente do AWS SDK.

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

2. Anote o nome do arquivo zip (boto3-layer.zip), que você usa na etapa 8 deste procedimento.
3. Abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
4. No painel de navegação, escolha Camadas.
5. Escolha Criar camada.
6. Insira valores para Nome e Descrição.
7. Em Tipo de entrada de código, escolha Carregar um arquivo.zip e selecione Upload.
8. Na caixa de diálogo, escolha o arquivo zip (boto3-layer.zip) que você criou na etapa 1 deste procedimento.
9. Para Tempos de execução compatíveis, escolha a versão mais recente do Python.
10. Escolha Criar para criar a camada.
11. Escolha o ícone do menu do painel de navegação.
12. Selecione Funções no painel de navegação.
13. Na lista de recursos, escolha a função que você criou anteriormente em ???.
14. Escolha a guia Código.
15. Na seção Camadas, escolha Adicionar uma camada.
16. Escolha camadas personalizadas.
17. Em Camadas personalizadas, escolha o nome da camada que você inseriu na etapa 6.
18. Em Versão, escolha a versão da camada, que deve ser 1.
19. Escolha Adicionar.

Adicionar código Python (console)

Nesta etapa, você adiciona seu código Python à sua função do Lambda por meio do editor de código do console Lambda. O código detecta rótulos em uma imagem usando a operação `DetectLabels`.

Ele retorna uma matriz de rótulos detectados na imagem, bem como o nível de confiança nos rótulos detectados.

O documento que você fornece para a operação `DetectLabels` pode estar localizado em um bucket do Amazon S3 ou em um computador local.

Para adicionar código Python (console)

1. Navegue até a guia Código .
2. No editor de código, substitua o código em lambda_function.py pelo código a seguir:

```
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. Escolha Implantar para implantar sua função do Lambda.

Para adicionar código Python (console)

Agora que você criou sua função do Lambda, você pode invocá-la para detectar rótulos em uma imagem.

Nesta etapa, você executa o código Python em seu computador, que passa uma imagem local ou uma imagem em um bucket do Amazon S3 para sua função do Lambda.

Certifique-se de executar o código na mesma AWS região em que você criou a função Lambda. Você pode visualizar a AWS região da sua função Lambda na barra de navegação da página de detalhes da função no console do Lambda.

Se a função do Lambda retornar um erro de tempo limite, estenda o período de tempo limite para a função do Lambda. Para obter mais informações, consulte Configurar o tempo limite da função (console).

Para obter mais informações sobre como invocar uma função do Lambda a partir do seu código, consulte Chamada de funções do AWS Lambda.

Para testar sua função do Lambda

1. Se você ainda não fez isso, faça o seguinte:

   a. Certifique-se de que o usuário tenha permissão `lambda:InvokeFunction`. Você pode usar a seguinte política:

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Sid": "InvokeLambda",
               "Effect": "Allow",
               "Action": "lambda:InvokeFunction",
               "Resource": "ARN for lambda function"
           }
       ]
   }
   ```

   Você pode obter o ARN da sua função do Lambda na visão geral da função no console do Lambda.

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos em AWS IAM Identity Center:

  Crie um conjunto de permissões. Siga as instruções em Criação de um conjunto de permissões no Guia do usuário do AWS IAM Identity Center .

- Usuários gerenciados no IAM com provedor de identidades:

  Crie um perfil para a federação de identidades. Siga as instruções em Criando um perfil para um provedor de identidades de terceiros (federação) no Guia do Usuário do IAM.

- Usuários do IAM:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de um perfil para um usuário do IAM no Guia do usuário do IAM.
  - (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adição de permissões a um usuário (console) no Guia do usuário do IAM.

b. Instale e configure o AWS SDK para Python. Para obter mais informações, consulte Etapa 2: configurar o AWS CLI e AWS SDKs.

2. Salve o código a seguir em um arquivo chamado `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. Execute o código. Se o documento estiver em um bucket do Amazon S3, certifique-se de que seja o mesmo bucket que você especificou anteriormente na etapa 12 do ???.

   Se for bem-sucedido, seu código retornará uma resposta JSON parcial para cada tipo de bloco detectado no documento.

Exemplos de código para o Amazon Rekognition usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Rekognition AWS com um kit de desenvolvimento de software (SDK). Os exemplos de código deste capítulo têm como objetivo complementar os exemplos encontrados no restante deste guia.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Conceitos básicos

Olá, Amazon Rekognition

O exemplo de código a seguir mostra como começar a usar o Amazon Rekognition.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o CMake arquivo 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})
```

Código para o arquivo de origem 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;
}
```

- Para obter detalhes da API, consulte ListCollectionsa Referência AWS SDK para C++ da API.

Exemplos de código

- Exemplos básicos para o uso do Amazon Rekognition AWS SDKs
  - Olá, Amazon Rekognition
  - Ações para o Amazon Rekognition usando AWS SDKs
    - Use CompareFaces com um AWS SDK ou CLI
    - Use CreateCollection com um AWS SDK ou CLI
    - Use DeleteCollection com um AWS SDK ou CLI
    - Use DeleteFaces com um AWS SDK ou CLI
    - Use DescribeCollection com um AWS SDK ou CLI
    - Use DetectFaces com um AWS SDK ou CLI
    - Use DetectLabels com um AWS SDK ou CLI
    - Use DetectModerationLabels com um AWS SDK ou CLI
    - Use DetectText com um AWS SDK ou CLI
    - Use DisassociateFaces com um AWS SDK ou CLI
    - Use GetCelebrityInfo com um AWS SDK ou CLI
    - Use IndexFaces com um AWS SDK ou CLI
    - Use ListCollections com um AWS SDK ou CLI
    - Use ListFaces com um AWS SDK ou CLI
    - Use RecognizeCelebrities com um AWS SDK ou CLI
    - Use SearchFaces com um AWS SDK ou CLI

    - Use SearchFacesByImage com um AWS SDK ou CLI
- Cenários para o uso do Amazon Rekognition AWS SDKs
  - Crie uma coleção do Amazon Rekognition e encontre faces nela usando um SDK AWS
  - Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
  - Detecte PPE em imagens com o Amazon Rekognition usando um SDK AWS
  - Detecte e exiba elementos em imagens com o Amazon Rekognition usando um SDK AWS
  - Detecte rostos em uma imagem usando um AWS SDK
  - Detecte informações em vídeos usando o Amazon Rekognition e o SDK AWS
  - Detecte objetos em imagens com o Amazon Rekognition usando um SDK AWS
  - Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um SDK AWS
  - Salve EXIF e outras informações de imagem usando um SDK AWS

Exemplos básicos para o uso do Amazon Rekognition AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Rekognition com. AWS SDKs

Exemplos

- Olá, Amazon Rekognition
- Ações para o Amazon Rekognition usando AWS SDKs
  - Use CompareFaces com um AWS SDK ou CLI
  - Use CreateCollection com um AWS SDK ou CLI
  - Use DeleteCollection com um AWS SDK ou CLI
  - Use DeleteFaces com um AWS SDK ou CLI
  - Use DescribeCollection com um AWS SDK ou CLI
  - Use DetectFaces com um AWS SDK ou CLI
  - Use DetectLabels com um AWS SDK ou CLI
  - Use DetectModerationLabels com um AWS SDK ou CLI
  - Use DetectText com um AWS SDK ou CLI
  - Use DisassociateFaces com um AWS SDK ou CLI
  - Use GetCelebrityInfo com um AWS SDK ou CLI

- Use IndexFaces com um AWS SDK ou CLI
- Use ListCollections com um AWS SDK ou CLI
- Use ListFaces com um AWS SDK ou CLI
- Use RecognizeCelebrities com um AWS SDK ou CLI
- Use SearchFaces com um AWS SDK ou CLI
- Use SearchFacesByImage com um AWS SDK ou CLI

Olá, Amazon Rekognition

O exemplo de código a seguir mostra como começar a usar o Amazon Rekognition.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o CMake arquivo 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})
```

Código para o arquivo de origem 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;
}
```

- Para obter detalhes da API, consulte ListCollectionsa Referência AWS SDK para C++ da API.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Ações para o Amazon Rekognition usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Rekognition com. AWS SDKs Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon Rekognition e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o uso do Amazon Rekognition AWS SDKs .

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência da API do Amazon Rekognition.

Exemplos

- Use CompareFaces com um AWS SDK ou CLI
- Use CreateCollection com um AWS SDK ou CLI
- Use DeleteCollection com um AWS SDK ou CLI
- Use DeleteFaces com um AWS SDK ou CLI
- Use DescribeCollection com um AWS SDK ou CLI
- Use DetectFaces com um AWS SDK ou CLI
- Use DetectLabels com um AWS SDK ou CLI
- Use DetectModerationLabels com um AWS SDK ou CLI
- Use DetectText com um AWS SDK ou CLI
- Use DisassociateFaces com um AWS SDK ou CLI
- Use GetCelebrityInfo com um AWS SDK ou CLI
- Use IndexFaces com um AWS SDK ou CLI
- Use ListCollections com um AWS SDK ou CLI
- Use ListFaces com um AWS SDK ou CLI

- Use RecognizeCelebrities com um AWS SDK ou CLI
- Use SearchFaces com um AWS SDK ou CLI
- Use SearchFacesByImage com um AWS SDK ou CLI

Use **`CompareFaces`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CompareFaces`.

Para obter mais informações, consulte Comparar faces em imagens.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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.");
        }
    }
```

- Para obter detalhes da API, consulte CompareFacesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como comparar faces em duas imagens

O comando `compare-faces` a seguir compara faces em duas imagens armazenadas em um bucket do Amazon S3.

```
aws rekognition compare-faces \
    --source-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"source.jpg"}}' \
    --target-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"target.jpg"}}'
```

Saída:

```
{
    "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
    }
}
```

Para obter mais informações, consulte Compare rostos em imagens no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte CompareFacesna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte CompareFacesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}")
    }
}
```

- Para obter detalhes da API, consulte a CompareFacesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a CompareFacesReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`CreateCollection`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateCollection`.

Para obter mais informações, consulte Criar uma coleção.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}");
        }
    }
```

- Para obter detalhes da API, consulte CreateCollectiona Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como criar uma coleção

O comando `create-collection` a seguir cria uma coleção com o nome especificado.

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

Saída:

```
{
    "CollectionArn": "aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
    "FaceModelVersion": "4.0",
    "StatusCode": 200
}
```

Para obter mais informações, consulte Criar uma coleção no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte CreateCollectionna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateCollectiona Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}")
    }
}
```

- Para obter detalhes da API, consulte a CreateCollectionreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a CreateCollectionReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DeleteCollection`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteCollection`.

Para obter mais informações, consulte Excluir uma coleção.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}");
```

```
        }
    }
```

- Para obter detalhes da API, consulte DeleteCollectiona Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir uma coleção

O comando `delete-collection` a seguir exclui a coleção especificada.

```
aws rekognition delete-collection \
    --collection-id MyCollection
```

Saída:

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

Para obter mais informações, consulte Excluir uma coleção no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte DeleteCollectionna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteCollectiona Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}")
    }
}
```

- Para obter detalhes da API, consulte a DeleteCollectionreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a DeleteCollectionReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DeleteFaces`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteFaces`.

Para obter mais informações, consulte Excluir faces de uma coleção.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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}");
            });
        }
    }
```

- Para obter detalhes da API, consulte DeleteFacesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir faces de uma coleção

O comando `delete-faces` a seguir exclui uma face especificada de uma coleção.

```
aws rekognition delete-faces \
    --collection-id MyCollection
    --face-ids '["0040279c-0178-436e-b70a-e61b074e96b0"]'
```

Saída:

```
{
    "DeletedFaces": [
        "0040279c-0178-436e-b70a-e61b074e96b0"
    ]
}
```

Para obter mais informações, consulte Excluir faces de uma coleção no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte DeleteFacesna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteFacesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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")
    }
}
```

- Para obter detalhes da API, consulte a DeleteFacesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a DeleteFacesReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DescribeCollection`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeCollection`.

Para obter mais informações, consulte Descrever uma coleção.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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}");
        }
    }
```

- Para obter detalhes da API, consulte DescribeCollectiona Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como descrever uma coleção

O exemplo de `describe-collection` a seguir exibe os detalhes da coleção especificada.

```
aws rekognition describe-collection \
    --collection-id MyCollection
```

Saída:

```
{
    "FaceCount": 200,
    "CreationTimestamp": 1569444828.274,
    "CollectionARN": "arn:aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
    "FaceModelVersion": "4.0"
}
```

Para obter mais informações, consulte Descrever uma coleção no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte DescribeCollectionna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeCollectiona Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}")
    }
}
```

- Para obter detalhes da API, consulte a DescribeCollectionreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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()
```

- Para obter detalhes da API, consulte a DescribeCollectionReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DetectFaces`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectFaces`.

Para obter mais informações, consulte Detectar faces em uma imagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
            }
        }
    }
```

Exiba as informações da caixa delimitadora de todas as faces em uma imagem.

```
    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}");
        }
    }
```

- Para obter detalhes da API, consulte DetectFacesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como detectar faces em uma imagem

O comando `detect-faces` a seguir detecta faces na imagem especificada armazenada em um bucket do Amazon S3.

```
aws rekognition detect-faces \
    --image '{"S3Object":{"Bucket":"MyImageS3Bucket","Name":"MyFriend.jpg"}}' \
    --attributes "ALL"
```

Saída:

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

Para obter mais informações, consulte Detectar faces em uma imagem no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte DetectFacesna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectFacesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}")
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte a DetectFacesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a DetectFacesReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DetectLabels`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectLabels`.

Para obter mais informações, consulte Detectar rótulos em uma imagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
            }
        }
    }
```

Detecte rótulos em um arquivo de imagem armazenado em seu computador.

```
    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);
            }
        }
    }
```

- Para obter detalhes da API, consulte DetectLabelsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte DetectLabelsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como detectar rótulos em uma imagem

O exemplo de `detect-labels` a seguir detecta cenas e objetos em uma imagem armazenada em um bucket do Amazon S3.

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

Saída:

```
{
    "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"
}
```

Para obter mais informações, consulte Detectar rótulos em uma imagem no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte DetectLabelsna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectLabelsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DetectLabelsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a DetectLabelsReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DetectModerationLabels`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectModerationLabels`.

Para obter mais informações, consulte Detectar imagens inapropriadas.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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);
            }
        }
    }
```

- Para obter detalhes da API, consulte DetectModerationLabelsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como detectar conteúdo não seguro em uma imagem

O comando `detect-moderation-labels` a seguir detecta conteúdo não seguro na imagem especificada armazenada em um bucket do Amazon S3.

```
aws rekognition detect-moderation-labels \
    --image "S3Object={Bucket=MyImageS3Bucket,Name=gun.jpg}"
```

Saída:

```
{
    "ModerationModelVersion": "3.0",
    "ModerationLabels": [
```

```
        {
            "Confidence": 97.29618072509766,
            "ParentName": "Violence",
            "Name": "Weapon Violence"
        },
        {
            "Confidence": 97.29618072509766,
            "ParentName": "",
            "Name": "Violence"
        }
    ]
}
```

Para obter mais informações, consulte Detectando imagens inapropriadas no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte DetectModerationLabelsna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectModerationLabelsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DetectModerationLabelsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a DetectModerationLabelsReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DetectText`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectText`.

Para obter mais informações, consulte Detectar texto em uma imagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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);
            }
        }
    }
```

- Para obter detalhes da API, consulte DetectTexta Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como detectar texto em uma imagem

O comando `detect-text` a seguir detecta texto na imagem especificada.

```
aws rekognition detect-text \
    --image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"ExamplePicture.jpg"}}'
```

Saída:

```
{
    "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
        }
    ]
}
```

- Para obter detalhes da API, consulte DetectTextna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectTexta Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DetectTextreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a DetectTextReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DisassociateFaces`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `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
```

- Para obter detalhes da API, consulte DisassociateFacesna Referência de AWS CLI Comandos.

Python

SDK para 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()
```

- Para obter detalhes da API, consulte a DisassociateFacesReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`GetCelebrityInfo`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetCelebrityInfo`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte GetCelebrityInfoa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como obter informações sobre uma celebridade

O comando `get-celebrity-info` a seguir exibe informações sobre a celebridade especificada. O parâmetro `id` é proveniente de uma chamada anterior a `recognize-celebrities`.

```
aws rekognition get-celebrity-info --id nnnnnnn
```

Saída:

```
{
    "Name": "Celeb A",
    "Urls": [
        "www.imdb.com/name/aaaaaaaaa"
    ]
}
```

Para obter mais informações, consulte Obter informações sobre uma celebridade no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte GetCelebrityInfona Referência de AWS CLI Comandos.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`IndexFaces`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `IndexFaces`.

Para obter mais informações, consulte Adicionar faces a uma coleção.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}");
            }
        }
    }
```

- Para obter detalhes da API, consulte IndexFacesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como adicionar faces a uma coleção

O comando `index-faces` a seguir adiciona as faces encontradas em uma imagem à coleção especificada.

```
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"
```

Saída:


```
{
    "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"
}
```

Para obter mais informações, consulte Adicionar faces a uma coleção no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte IndexFacesna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte IndexFacesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a IndexFacesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a IndexFacesReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`ListCollections`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListCollections`.

Para obter mais informações, consulte Listar coleções.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
```

- Para obter detalhes da API, consulte ListCollectionsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar as coleções disponíveis

O `list-collections` comando a seguir lista as coleções disponíveis na AWS conta.

```
aws rekognition list-collections
```

Saída:

```
{
    "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"
    ]
}
```

Para obter mais informações, consulte Listar coleções no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte ListCollectionsna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte ListCollectionsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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)
        }
    }
}
```

- Para obter detalhes da API, consulte a ListCollectionsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a ListCollectionsReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`ListFaces`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListFaces`.

Para obter mais informações, consulte Listar faces em uma coleção.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    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));
        }
    }
```

- Para obter detalhes da API, consulte ListFacesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar faces de uma coleção

O comando `list-faces` a seguir lista as faces na coleção especificada.

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

Saída:

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

Para obter mais informações, consulte Listar faces em uma coleção no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte ListFacesna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte ListFacesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListFacesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a ListFacesReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`RecognizeCelebrities`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RecognizeCelebrities`.

Para obter mais informações, consulte Reconhecer celebridades em uma imagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.");
        }
    }
```

- Para obter detalhes da API, consulte RecognizeCelebritiesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como reconhecer celebridades em uma imagem

O seguinte comando `recognize-celebrities` reconhece celebridades na imagem especificada armazenada em um bucket do Amazon S3:

```
aws rekognition recognize-celebrities \
    --image "S3Object={Bucket=MyImageS3Bucket,Name=moviestars.jpg}"
```

Saída:

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

Para obter mais informações, consulte Reconhecer celebridades em uma imagem no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte RecognizeCelebritiesna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte RecognizeCelebritiesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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.")
    }
}
```

- Para obter detalhes da API, consulte a RecognizeCelebritiesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a RecognizeCelebritiesReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`SearchFaces`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SearchFaces`.

Para obter mais informações, consulte Pesquisar uma face (Face ID).

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}");
            });
        }
    }
```

- Para obter detalhes da API, consulte SearchFacesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como pesquisar faces em uma coleção que corresponda a um ID facial.

O comando `search-faces` a seguir pesquisa faces em uma coleção que correspondem ao ID facial especificado.

```
aws rekognition search-faces \
    --face-id 8d3cfc70-4ba8-4b36-9644-90fba29c2dac \
    --collection-id MyCollection
```

Saída:

```
{
    "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
        }
    ]
}
```

Para obter mais informações, consulte Procurando uma face com um ID facial no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte SearchFacesna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte SearchFacesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a SearchFacesReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`SearchFacesByImage`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SearchFacesByImage`.

Para obter mais informações, consulte Pesquisar uma face (imagem).

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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}");
            });
        }
    }
```

- Para obter detalhes da API, consulte SearchFacesByImagea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como pesquisar faces em uma coleção que corresponda à maior face em uma imagem.

O seguinte comando `search-faces-by-image` pesquisa faces em uma coleção que corresponda à maior face na imagem especificada:

```
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"
}
```


Para obter mais informações, consulte Procurando um rosto com uma imagem no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte SearchFacesByImagena Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte SearchFacesByImagea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
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
```

- Para obter detalhes da API, consulte a SearchFacesByImageReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Cenários para o uso do Amazon Rekognition AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Rekognition com. AWS SDKs Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon Rekognition ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie uma coleção do Amazon Rekognition e encontre faces nela usando um SDK AWS
- Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
- Detecte PPE em imagens com o Amazon Rekognition usando um SDK AWS
- Detecte e exiba elementos em imagens com o Amazon Rekognition usando um SDK AWS
- Detecte rostos em uma imagem usando um AWS SDK

- Detecte informações em vídeos usando o Amazon Rekognition e o SDK AWS
- Detecte objetos em imagens com o Amazon Rekognition usando um SDK AWS
- Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um SDK AWS
- Salve EXIF e outras informações de imagem usando um SDK AWS

Crie uma coleção do Amazon Rekognition e encontre faces nela usando um SDK AWS

O código de exemplo a seguir mostra como:

- Criar uma coleção do Amazon Rekognition.
- Adicionar imagens à coleção e detectar faces nela.
- Pesquisar na coleção faces que correspondam a uma imagem de referência.
- Excluir uma coleção.

Para obter mais informações, consulte Pesquisar faces em uma coleção.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar classes que envolvam as funções do 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
```

Usar as classes wrapper para criar uma coleção de faces a partir de um conjunto de imagens e, em seguida, pesquisar faces na coleção.

```
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)
```

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

.NET

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços utilizados neste exemplo

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços utilizados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Detecte PPE em imagens com o Amazon Rekognition usando um SDK AWS

Os exemplos de código a seguir mostram como criar uma aplicação que usa o Amazon Rekognition para detectar equipamentos de proteção individual (EPI) em imagens.

Java

SDK para Java 2.x

Mostra como criar uma AWS Lambda função que detecta imagens com equipamento de proteção individual.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Detecte e exiba elementos em imagens com o Amazon Rekognition usando um SDK AWS

O código de exemplo a seguir mostra como:

- Detectar elementos em imagens usando o Amazon Rekognition.
- Exibir imagens e desenhar caixas delimitadoras ao redor dos elementos detectados.

Para obter mais informações, consulte Exibindo caixas delimitadoras.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar classes para agrupar as funções do 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
```

Criar funções auxiliares para desenhar caixas delimitadoras e polígonos.

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

logger = logging.getLogger(__name__)


def show_bounding_boxes(image_bytes, box_sets, colors):
```

```
    """
    Draws bounding boxes on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param box_sets: A list of lists of bounding boxes to draw on the image.
    :param colors: A list of colors to use to draw the bounding boxes.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for boxes, color in zip(box_sets, colors):
        for box in boxes:
            left = image.width * box["Left"]
            top = image.height * box["Top"]
            right = (image.width * box["Width"]) + left
            bottom = (image.height * box["Height"]) + top
            draw.rectangle([left, top, right, bottom], outline=color, width=3)
    image.show()



def show_polygons(image_bytes, polygons, color):
    """
    Draws polygons on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param polygons: The list of polygons to draw on the image.
    :param color: The color to use to draw the polygons.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for polygon in polygons:
        draw.polygon(
            [
                (image.width * point["X"], image.height * point["Y"])
                for point in polygon
            ],
            outline=color,
        )
    image.show()
```

Criar classes para analisar objetos retornados pelo 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
```

Use as classes wrapper para detectar elementos em imagens e exibir suas caixas delimitadoras. As imagens usadas neste exemplo podem ser encontradas GitHub junto com instruções e mais códigos.

```
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)
```

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Detecte rostos em uma imagem usando um AWS SDK

O código de exemplo a seguir mostra como:

- Salvar uma imagem em um bucket do Amazon S3.
- Usar o Amazon Rekognition para detectar detalhes faciais, como faixa etária, gênero e emoções (sorriso, etc.).
- Exibir esses detalhes.

Rust

SDK para Rust

Salve a imagem em um bucket do Amazon S3 com um prefixo uploads, use o Amazon Rekognition para detectar detalhes faciais, como faixa etária, gênero e emoções (sorriso, etc.), e exiba esses detalhes.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- Amazon Rekognition
- Amazon S3

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Detecte informações em vídeos usando o Amazon Rekognition e o SDK AWS

Os exemplos de código a seguir mostram como:

- Iniciar trabalhos do Amazon Rekognition para detectar elementos como pessoas, objetos e texto em vídeos.
- Verificar o status do trabalho até que os trabalhos terminem.
- Visualizar a lista de elementos detectados por cada trabalho.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha resultados de celebridades a partir de um vídeo localizado em um bucket do 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);
        }
    }
}
```

Detecte rótulos em um vídeo por meio de uma operação de detecção de rótulos.

```
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);
        }
    }
}
```

Detecte faces em um vídeo armazenado em um bucket do 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);
        }
    }
}
```

Detecte conteúdo impróprio ou ofensivo em um vídeo armazenado em um bucket do 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);
        }
    }
}
```

Detecte segmentos de sinal técnico e segmentos de detecção de tomada em um vídeo armazenado em um bucket do 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);
        }
    }
}
```

Detecte texto em um vídeo armazenado em um bucket do 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);
        }
    }
}
```

Detecte pessoas em um vídeo armazenado em um bucket do 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);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection

- StartTextDetection

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Detecte faces em um vídeo armazenado em um bucket do 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}")
        }
    }
}
```

Detecte conteúdo impróprio ou ofensivo em um vídeo armazenado em um bucket do 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)
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition

- StartContentModeration
- StartLabelDetection
- StartPersonTracking
- StartSegmentDetection
- StartTextDetection

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Detecte objetos em imagens com o Amazon Rekognition usando um SDK AWS

Os exemplos de código a seguir mostram como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

.NET

SDK para .NET

Mostra como usar a API .NET do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK para JavaScript (v3)

Mostra como usar o Amazon Rekognition AWS SDK para JavaScript com o para criar um aplicativo que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Aprenda como:

- Criar um usuário não autenticado usando o Amazon Cognito.
- Analisar imagens em busca de objetos usando o Amazon Rekognition.
- Verificar um endereço de e-mail para o Amazon SES.
- Enviar uma notificação por e-mail usando o Amazon SES.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um aplicativo web que permite fazer o seguinte:

- Carregar fotos em um bucket do Amazon Simple Storage Service (Amazon S3).
- Usar o Amazon Rekognition para analisar e rotular as fotos.
- Usar o Amazon Simple Email Service (Amazon SES) para enviar relatórios de análise da imagem por e-mail.

Este exemplo contém dois componentes principais: uma página da Web criada com o React e um serviço REST escrito em Python que é construído com o Flask-. JavaScript RESTful

Você pode usar a página da Web do React para:

- Exibir uma lista de imagens que estão armazenadas no bucket do S3.
- Carregar imagens do computador para o bucket do S3.
- Exibir imagens e rótulos que identificam os itens detectados na imagem.
- Obter um relatório de todas as imagens no bucket do S3 e enviar um relatório por e-mail.

A página da Web chama o serviço REST. O serviço envia solicitações à AWS para realizar as seguintes ações:

- Obter e filtrar a lista de imagens no bucket do S3.
- Carregar fotos no bucket do S3.
- Usar o Amazon Rekognition para analisar fotos individuais e obter uma lista dos rótulos que identifiquem os itens detectados nas fotos.
- Analisar todas as fotos no bucket do S3 e usar o Amazon SES para enviar um relatório por e-mail.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um SDK AWS

Os exemplos de código a seguir mostram como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK para Python (Boto3)

Use o Amazon Rekognition para detectar faces, objetos e pessoas em vídeos iniciando trabalhos de detecção assíncrona. Este exemplo também configura o Amazon Rekognition para notificar um tópico do Amazon Simple Notification Service (Amazon SNS) quando os trabalhos são concluídos e inscreve uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico. Quando a fila recebe uma mensagem sobre um trabalho, o trabalho é recuperado e os resultados são apresentados.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços utilizados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

- Amazon SNS
- Amazon SQS

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Salve EXIF e outras informações de imagem usando um SDK AWS

O código de exemplo a seguir mostra como:

- Obter informações de EXIF de um arquivo JPG, JPEG ou PNG.
- Fazer upload do arquivo de imagem para um bucket do Amazon S3.
- Usar o Amazon Rekognition para identificar os três principais atributos (rótulos) no arquivo.
- Adicione as informações de EXIF e rótulo a uma tabela do Amazon DynamoDB na região.

Rust

SDK para Rust

Obtenha informações de EXIF de um arquivo JPG, JPEG ou PNG, faça upload do arquivo de imagem para um bucket do Amazon S3, use o Amazon Rekognition para identificar os três principais atributos (rótulos no Amazon Rekognition) no arquivo e adicione as informações de EXIF e de rótulo a uma tabela do Amazon DynamoDB na região.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando o Rekognition com um SDK AWS. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Referência da API

A referência da API Amazon Rekognition agora está localizada em Referência da API do Amazon Rekognition.

Segurança do Amazon Rekognition

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de um data center e de uma arquitetura de rede criados para atender aos requisitos das organizações mais sensíveis à segurança.

Use os tópicos a seguir para saber como proteger os recursos do Amazon Rekognition.

Tópicos

- Gerenciamento de identidade e acesso para o Amazon Rekognition
- Proteção de dados no Amazon Rekognition
- Usando o Amazon Rekognition com endpoints da VPC da Amazon
- Validação de conformidade para o Amazon Rekognition
- Resiliência no Amazon Rekognition
- Análise de configuração e vulnerabilidade no Amazon Rekognition
- Prevenção do problema do substituto confuso entre serviços
- Segurança da infraestrutura no Amazon Rekognition

Gerenciamento de identidade e acesso para o Amazon Rekognition

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (ter permissões) para usar os recursos do Amazon Rekognition. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

- Público
- Autenticar com identidades
- Gerenciar o acesso usando políticas
- Como o Amazon Rekognition funciona com o IAM

- AWS políticas gerenciadas para o Amazon Rekognition
- Práticas recomendadas de política
- Usar o console do Amazon Rekognition
- Exemplos de uso de políticas baseadas em identidade do Amazon Rekognition
- Exemplos de políticas baseadas em recursos do Amazon Rekognition
- Solução de problemas de identidade e acesso do Amazon Rekognition

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz no Amazon Rekognition.

Usuário do serviço — Se você usar o serviço Amazon Rekognition para fazer seu trabalho, seu administrador fornecerá as credenciais e permissões de que você precisa. À medida que você usa mais recursos do Amazon Rekognition para realizar seu trabalho, talvez precise de permissões adicionais. Compreenda como o acesso é gerenciado pode ajudar a solicitar as permissões corretas ao administrador. Se você não conseguir acessar um recurso no Amazon Rekognition, consulte Solução de problemas de identidade e acesso do Amazon Rekognition.

Administrador de serviços — Se você é responsável pelos recursos do Amazon Rekognition em sua empresa, provavelmente tem acesso total ao Amazon Rekognition. É seu trabalho determinar quais recursos e recursos do Amazon Rekognition seus usuários do serviço devem acessar. Envie as solicitações ao administrador do IAM para alterar as permissões dos usuários de serviço. Revise as informações nesta página para compreender os conceitos básicos do IAM. Para saber mais sobre como sua empresa pode usar o IAM com o Amazon Rekognition, consulte Como o Amazon Rekognition funciona com o IAM.

Administrador do IAM — Se você for administrador do IAM, talvez queira saber detalhes sobre como criar políticas para gerenciar o acesso ao Amazon Rekognition. Para ver exemplos de políticas baseadas em identidade do Amazon Rekognition que você pode usar no IAM, consulte Exemplos de uso de políticas baseadas em identidade do Amazon Rekognition.

Autenticar com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como o Usuário raiz da conta da AWS, como usuário do IAM ou assumindo uma função do IAM.

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Usuários (IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como identidade federada, o administrador já configurou anteriormente a federação de identidades usando perfis do IAM. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login em AWS, consulte Como fazer login Conta da AWS no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para designar solicitações por conta própria, consulte Versão 4 do AWS Signature para solicitações de API no Guia do usuário do IAM.

Independente do método de autenticação usado, também pode ser necessário fornecer informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação multifator no Guia do usuário do AWS IAM Identity Center e Usar a autenticação multifator da AWS no IAM no Guia do usuário do IAM.

Grupos e usuários do IAM

Um usuário do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, é recomendável contar com credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, é recomendável alternar as chaves de acesso. Para obter mais informações, consulte Alternar as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo no Guia do Usuário do IAM.

Um grupo do IAM é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdminse conceder a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Casos de uso para usuários do IAM no Guia do usuário do IAM.

Perfis do IAM

Uma função do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. Para assumir temporariamente uma função do IAM no AWS Management Console, você pode alternar de um usuário para uma função do IAM (console). Você pode assumir uma função chamando uma operação de AWS API AWS CLI ou usando uma URL personalizada. Para obter mais informações sobre métodos para usar perfis, consulte Métodos para assumir um perfil no Guia do usuário do IAM.

Perfis do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, é possível criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas por ele. Para ter mais informações sobre perfis para federação, consulte Criar um perfil para um provedor de identidade de terceiros (federação) no Guia do usuário do IAM. Se usar o Centro de Identidade do IAM, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o Centro de Identidade do IAM correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de Permissões no Guia do Usuário do AWS IAM Identity Center .
- Permissões temporárias para usuários do IAM: um usuário ou um perfil do IAM pode presumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas: é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.
- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos na Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso

usando as permissões da entidade principal da chamada, usando um perfil de serviço ou um perfil vinculado ao serviço.

- Sessões de acesso direto (FAS) — Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Sessões de acesso direto.
- Perfil de serviço: um perfil de serviço é um perfil do IAM que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.
- Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir o perfil de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados ao serviço.

- Aplicativos em execução na Amazon EC2 — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma EC2 instância e fazendo solicitações AWS CLI de AWS API. Isso é preferível a armazenar chaves de acesso na EC2 instância. Para atribuir uma AWS função a uma EC2 instância e disponibilizá-la para todos os aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém a função e permite que os programas em execução na EC2 instância recebam credenciais temporárias. Para obter mais informações, consulte Usar uma função do IAM para conceder permissões a aplicativos executados em EC2 instâncias da Amazon no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida

ou negada. A maioria das políticas é armazenada AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e perfis não têm permissões. Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem assumir os perfis.

As políticas do IAM definem permissões para uma ação independentemente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função da AWS Management Console AWS CLI, da ou da AWS API.

Usar políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Usando políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico.

Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o AWS WAF Amazon VPC são exemplos de serviços que oferecem suporte. ACLs Para saber mais ACLs, consulte a visão geral da lista de controle de acesso (ACL) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um recurso avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs) — SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em AWS Organizations. AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as suas contas. O SCP limita as permissões para entidades nas contas dos membros, incluindo cada

uma Usuário raiz da conta da AWS. Para obter mais informações sobre Organizations e SCPs, consulte Políticas de controle de serviços no Guia AWS Organizations do Usuário.

- Políticas de controle de recursos (RCPs) — RCPs são políticas JSON que você pode usar para definir o máximo de permissões disponíveis para recursos em suas contas sem atualizar as políticas do IAM anexadas a cada recurso que você possui. O RCP limita as permissões para recursos nas contas dos membros e pode afetar as permissões efetivas para identidades, incluindo a Usuário raiz da conta da AWS, independentemente de pertencerem à sua organização. Para obter mais informações sobre Organizations e RCPs, incluindo uma lista Serviços da AWS desse suporte RCPs, consulte Políticas de controle de recursos (RCPs) no Guia AWS Organizations do usuário.
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em recursos. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Como o Amazon Rekognition funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao Amazon Rekognition, você deve entender quais recursos do IAM estão disponíveis para uso com o Amazon Rekognition. Para ter uma visão de alto nível de como o Amazon Rekognition AWS e outros serviços funcionam com o IAM AWS , consulte Serviços que funcionam com o IAM no Guia do usuário do IAM.

Tópicos

- Políticas baseadas em identidade do Amazon Rekognition
- Políticas baseadas em recursos do Amazon Rekognition
- Perfis do IAM do Amazon Rekognition

Políticas baseadas em identidade do Amazon Rekognition

Com as políticas baseadas em identidade do IAM, é possível especificar ações e recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. O Amazon Rekognition oferece suporte a ações, recursos e chaves de condição específicas. Para conhecer todos os elementos usados em uma política JSON, consulte Referência de elementos de política JSON do IAM no Guia do usuário do IAM.

Ações

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome da operação de AWS API associada. Existem algumas exceções, como ações somente de permissão, que não têm uma operação de API correspondente. Algumas operações também exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

As ações de política no Amazon Rekognition usam o seguinte prefixo antes da ação: `rekognition:`. Por exemplo, para conceder permissão a alguém para detectar objetos, cenas ou conceitos em uma imagem com a operação da API `DetectLabels` do Amazon Rekognition, você inclui a ação `rekognition:DetectLabels` na política. As instruções de política devem incluir um elemento `Action` ou `NotAction`. O Amazon Rekognition define seu próprio conjunto de ações que descrevem as tarefas que você pode realizar com esse serviço.

Para especificar várias ações em uma única instrução, separe-as com vírgulas, como segue:

```
"Action": [
      "rekognition:action1",
      "rekognition:action2"
```

Você também pode especificar várias ações usando caracteres curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Describe`, inclua a seguinte ação:

```
"Action": "rekognition:Describe*"
```

Para ver uma lista de ações do Amazon Rekognition, consulte Ações definidas pelo Amazon Rekognition no Guia do usuário do IAM.

Recursos

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou `NotResource`. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Isso pode ser feito para ações que oferecem compatibilidade com um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem compatibilidade com permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Para obter mais informações sobre o formato de ARNs, consulte Amazon Resource Names (ARNs) e AWS Service Namespaces.

Por exemplo, para especificar a coleção `MyCollection` em sua declaração, use o ARN a seguir:

```
"Resource": "arn:aws:rekognition:us-east-1:123456789012:collection/MyCollection"
```

Para especificar todas as instâncias que pertencem a uma conta específica, use o caractere curinga (*):

```
"Resource": "arn:aws:rekognition:us-east-1:123456789012:collection/*"
```

Algumas ações do Amazon Rekognition, como aquelas para criar recursos, não podem ser executadas em um recurso específico. Nesses casos, você deve utilizar o caractere curinga (*).

```
"Resource": "*"
```

Para ver uma lista dos tipos de recursos do Amazon Rekognition ARNs e seus, consulte Recursos definidos pelo Amazon Rekognition no Guia do usuário do IAM. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Ações definidas pelo Amazon Rekognition.

Chaves de condição

O Amazon Rekognition não fornece nenhuma chave de condição específica do serviço, mas oferece suporte ao uso de algumas chaves de condição globais. Para ver todas as chaves de condição AWS globais, consulte Chaves de contexto de condição AWS global no Guia do usuário do IAM.

Políticas baseadas em recursos do Amazon Rekognition

O Amazon Rekognition oferece suporte a políticas baseadas em recursos para operações de cópia de modelos de etiquetas personalizadas. Para obter mais informações, consulte exemplos de políticas baseadas em recursos do Amazon Rekognition.

Outros serviços, como o Amazon S3, também aceitam políticas de permissões baseadas em recurso. Por exemplo: você pode anexar uma política a um bucket do S3 para gerenciar permissões de acesso a esse bucket.

Para acessar imagens armazenadas em um bucket do Amazon S3, você deve ter permissão para acessar o objeto no bucket do S3. Com essa permissão, o Amazon Rekognition pode baixar imagens do bucket do S3. O exemplo de política a seguir permite que o usuário execute a `s3:GetObject` ação no bucket do S3 chamado amzn-s3-demo-bucket3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket3/*"
            ]
        }
    ]
}
```

Para usar um bucket do S3 com o versionamento ativado, adicione a ação `s3:GetObjectVersion`, conforme mostrado no exemplo a seguir.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket3/*"
            ]
        }
    ]
```

Perfis do IAM do Amazon Rekognition

Uma função do IAM é uma entidade dentro da sua AWS conta que tem permissões específicas.

Usando credenciais temporárias com o Amazon Rekognition

É possível usar credenciais temporárias para fazer login com federação, assumir um perfil do IAM ou assumir um perfil entre contas. Você obtém credenciais de segurança temporárias chamando operações de AWS STS API, como AssumeRoleou GetFederationToken.

O Amazon Rekognition oferece suporte ao uso de credenciais temporárias.

Perfis vinculados ao serviço

As funções vinculadas ao serviço permitem que AWS os serviços acessem recursos em outros serviços para concluir uma ação em seu nome. Os perfis vinculados a serviço aparecem em sua conta do IAM e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para perfis vinculados a serviço.

O Amazon Rekognition não oferece suporte a funções vinculadas a serviços.

Perfis de serviço

Esse atributo permite que um serviço assuma um perfil de serviço em seu nome. O perfil permite que o serviço acesse recursos em outros serviços para concluir uma ação em seu nome. Os perfis de serviço aparecem em sua conta do IAM e são de propriedade da conta. Isso significa que um administrador do IAM pode alterar as permissões para esse perfil. Porém, fazer isso pode alterar a funcionalidade do serviço.

O Amazon Rekognition oferece suporte a funções de serviço.

O uso de um perfil de serviço pode criar um problema de segurança em que o Amazon Rekognition é usado para chamar outro serviço e agir sobre recursos aos quais ele não deveria ter acesse. Para manter sua conta segura, você deve limitar o escopo do acesso do Amazon Rekognition apenas aos recursos que você está usando. Isso pode ser feito anexando uma política de confiança à suo perfil de serviço do IAM. Para obter informações sobre como fazer isso, consulte Prevenção do problema do substituto confuso entre serviços.

Escolha de um perfil do IAM no Amazon Rekognition

Ao configurar o Amazon Rekognition para analisar vídeos armazenados, você deve escolher uma função para permitir que o Amazon Rekognition acesse o Amazon SNS em seu nome. Se você já criou um perfil de serviço ou uma função vinculada a um serviço, o Amazon Rekognition fornece uma lista de funções para você escolher. Para obter mais informações, consulte the section called "Configuração do Amazon Rekognition Video".

Exemplo: configuração do Amazon Rekognition para acessar imagens em um bucket do Amazon S3

Veja a seguir um exemplo de como você pode configurar o Amazon Rekognition para analisar imagens em um bucket do Amazon S3. Se você quiser usar o Amazon Rekognition para analisar imagens em um bucket do Amazon S3, você deve fazer o seguinte:

1. Certifique-se de que seu usuário/função do IAM (o cliente) tenha permissão para chamar as operações relevantes da API Amazon Rekognition (como, etc.) DetectLabels DetectFaces

   Anexe uma política baseada em identidade que conceda as permissões apropriadas para invocar as operações de API desejadas. Por exemplo, para dar à sua função permissões para chamar `DetectLabels` e`DetectFaces`, você anexaria à sua função uma política parecida com esta:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:DetectLabels",
                "rekognition:DetectFaces"
                // other Rekognition permissions as needed
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

2. O serviço Amazon Rekognition precisa de permissão para acessar seu bucket Amazon S3. Crie uma função de serviço do IAM, que você precisará passar para o Amazon Rekognition ao fazer chamadas de API. Certifique-se de que a função de serviço: confia no diretor do serviço Amazon Rekognition tenha permissões para seu bucket. `s3:GetObject`

   A política de confiança pode ter a seguinte aparência:

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

   A política baseada em identidade anexada à função de serviço pode ter a seguinte aparência:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

AWS políticas gerenciadas para o Amazon Rekognition

Para adicionar permissões a usuários, grupos e funções, é mais fácil usar políticas AWS gerenciadas do que escrever políticas você mesmo. É necessário tempo e experiência para criar políticas gerenciadas pelo cliente do IAM que fornecem à sua equipe apenas as permissões de que precisam. Para começar rapidamente, você pode usar nossas políticas AWS gerenciadas. Essas políticas abrangem casos de uso comuns e estão disponíveis em sua AWS conta. Para obter mais informações sobre políticas AWS gerenciadas, consulte políticas AWS gerenciadas no Guia do usuário do IAM.

AWS os serviços mantêm e atualizam as políticas AWS gerenciadas. Você não pode alterar as permissões nas políticas AWS gerenciadas. Os serviços ocasionalmente acrescentam permissões adicionais a uma política gerenciada pela AWS para oferecer suporte a novos recursos. Esse tipo de atualização afeta todas as identidades (usuários, grupos e funções) em que a política está anexada. É mais provável que os serviços atualizem uma política gerenciada pela AWS quando um novo recurso for iniciado ou novas operações se tornarem disponíveis. Os serviços não removem as permissões de uma política AWS gerenciada, portanto, as atualizações de políticas não violarão suas permissões existentes.

Além disso, AWS oferece suporte a políticas gerenciadas para funções de trabalho que abrangem vários serviços. Por exemplo, a política ReadOnlyAccess AWS gerenciada fornece acesso somente de leitura a todos os AWS serviços e recursos. Quando um serviço lança um novo recurso, AWS adiciona permissões somente de leitura para novas operações e recursos. Para obter uma lista e descrições das políticas de perfis de trabalho, consulte Políticas gerenciadas pela AWS para perfis de trabalho no Guia do usuário do IAM.

Política gerenciada pela AWS: AmazonRekognitionFullAccess

`AmazonRekognitionFullAccess` concede acesso total aos recursos do Amazon Rekognition, incluindo a criação e a exclusão de coleções.

É possível anexar a política `AmazonRekognitionFullAccess` às identidades do IAM.

Detalhes das permissões

Esta política inclui as seguintes permissões.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Política gerenciada pela AWS: AmazonRekognitionReadOnlyAccess

`AmazonRekognitionReadOnlyAccess` concede acesso somente para leitura aos recursos do Amazon Rekognition.

É possível anexar a política `AmazonRekognitionReadOnlyAccess` às identidades do IAM.

Detalhes das permissões

Esta política inclui as seguintes permissões.

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

Política gerenciada pela AWS: AmazonRekognitionServiceRole

`AmazonRekognitionServiceRole` permite que o Amazon Rekognition chame os serviços do Amazon Kinesis Data Streams e do Amazon SNS em seu nome.

É possível anexar a política `AmazonRekognitionServiceRole` às identidades do IAM.

Ao usar esse perfil de serviço, você deve manter sua conta segura limitando o escopo do acesse do Amazon Rekognition apenas aos recursos que você está usando. Isso pode ser feito anexando uma política de confiança à suo perfil de serviço do IAM. Para obter informações sobre como fazer isso, consulte Prevenção do problema do substituto confuso entre serviços.

Detalhes das permissões

Esta política inclui as seguintes permissões.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:*:*:AmazonRekognition*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": "arn:aws:kinesis:*:*:stream/AmazonRekognition*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesisvideo:GetDataEndpoint",
                "kinesisvideo:GetMedia"
            ],
            "Resource": "*"
        }
    ]
}
```

Política gerenciada pela AWS: AmazonRekognitionCustomLabelsFullAccess

Esta política é para Amazon Rekognition Custom Labels; usuários. Use a AmazonRekognitionCustomLabelsFullAccess política para permitir aos usuários acesso total à API Amazon Rekognition Custom Labels e acesso total aos buckets do console criados pelo console Amazon Rekognition Custom Labels.

Detalhes das permissões

Esta política inclui as seguintes permissões.

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

```
    ]
}
```

Atualizações do Amazon Rekognition para políticas gerenciadas AWS

Veja detalhes sobre as atualizações das políticas AWS gerenciadas do Amazon Rekognition desde que esse serviço começou a monitorar essas alterações. Para receber alertas automáticos sobre alterações nesta página, assine o feed RSS na página de histórico de documentos do Amazon Rekognition.

Alteração	Descrição	Data
Ações envolvendo trabalhos de análise de mídia foram adicionadas à seguinte política gerenciada: • Política gerenciada pela AWS: AmazonRekognitionReadOnlyAccess	O Amazon Rekognition adicionou as seguintes ações à política gerenciada `AmazonRekognitionReadOnlyAccess` : • `GetMediaAnalysisJob` • `ListMediaAnalysisJob`	31 de outubro de 2023
As ações que envolvem o gerenciamento de usuários foram adicionadas à seguinte política gerenciada: • Política gerenciada pela AWS: AmazonRekognitionReadOnlyAccess	O Amazon Rekognition adicionou as seguintes ações à política gerenciada `AmazonRekognitionReadOnlyAccess` : • `ListUsers` • `SearchUsers` • `SearchUsersByImage`	12 de junho de 2023
As ações ProjectPolicy e a cópia personalizada do modelo de etiquetas foram adicionadas às seguintes políticas gerenciadas:	O Amazon Rekognition adicionou as seguintes ações às políticas gerenciadas `AmazonRekognitionCustomLabelsFullAcc`	21 de julho de 2022

Alteração	Descrição	Data
• Política gerenciada pela AWS: AmazonRekognitionFullAccess • Política gerenciada pela AWS: AmazonRekognitionCustomLabelsFullAccess	ess e AmazonRekognitionFullAccess : • CopyProjectVersion • PutProjectPolicy • ListProjectPolicies • DeleteProjectPolicy	
As ações ProjectPolicy e a cópia personalizada do modelo de etiquetas foram adicionadas às seguintes políticas gerenciadas: • Política gerenciada pela AWS: AmazonRekognitionReadOnlyAccess	O Amazon Rekognition adicionou as seguintes ações à política gerenciada AmazonRekognitionReadOnlyAccess : • ListProjectPolicies	21 de julho de 2022

Alteração	Descrição	Data
Atualização do gerenciamento do conjunto de dados para as seguintes políticas gerenciadas: • Política gerenciada pela AWS: AmazonRekognitionReadOnlyAccess • Política gerenciada pela AWS: AmazonRekognitionFullAccess • Política gerenciada pela AWS: AmazonRekognitionCustomLabelsFullAccess	O Amazon Rekognition adicionou as seguintes ações ao, e gerenciou as políticas AmazonRekognitionReadOnlyAccess AmazonRekognitionFullOnlyAccess AmazonRekognitionCustomLabelsFullAccess • `CreateDataset` • `ListDatasetEntries` • `ListDatasetLabels` • `DescribeDataset` • `UpdateDatasetEntries` • `DistributeDatasetEntries` • `DeleteDataset`	1 de novembro de 2021
Atualização de marcação para Política gerenciada pela AWS: AmazonRekognitionReadOnlyAccess e Política gerenciada pela AWS: AmazonRekognitionFullAccess	O Amazon Rekognition adicionou novas ações de marcação às políticas e. AmazonRekognitionFullAccess AmazonRekognitionReadOnlyAccess	2 de abril de 2021
O Amazon Rekognition começou a monitorar as alterações	O Amazon Rekognition começou a monitorar as mudanças em suas políticas gerenciadas. AWS	2 de abril de 2021

Práticas recomendadas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do Amazon Rekognition em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para obter mais informações, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: você pode adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, você pode escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como AWS CloudFormation. Para obter mais informações, consulte Elementos da política JSON do IAM: condição no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar a criar políticas seguras e funcionais. Para obter mais informações, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para obter mais informações, consulte Configuração de acesso à API protegido por MFA no Guia do Usuário do IAM.

Para obter mais informações sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Usar o console do Amazon Rekognition

Com exceção do recurso de rótulos personalizados, o Amazon Rekognition não exige nenhuma permissão adicional ao usar o console do Amazon Rekognition. Para obter informações sobre Amazon Rekognition Custom Labels, consulte Etapa 5: Configure as permissões do console de Amazon Rekognition Custom Labels.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para a API AWS CLI ou para a AWS API. Em vez disso, permita o acesso somente às ações que corresponderem a operação da API que você estiver tentando executar.

Exemplos de uso de políticas baseadas em identidade do Amazon Rekognition

Por padrão, usuários e funções não têm permissão para criar ou modificar recursos do Amazon Rekognition. Eles também não podem realizar tarefas usando a AWS API AWS Management Console AWS CLI, ou. Um administrador do IAM deve criar políticas do IAM que concedam aos usuários e perfis permissão para executarem operações de API específicas nos recursos especificados de que precisam. O administrador deve anexar essas políticas aos usuários ou grupos que exigem essas permissões.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documentos de política JSON, consulte Criar políticas na guia JSON no Guia do usuário do IAM.

Tópicos

- Exemplo de políticas de Amazon Rekognition Custom Labels
- Exemplo 1: Permitir que um usuário acesse os recursos somente para leitura
- Exemplo 2: Permitir que um usuário tenha acesso total aos recursos
- Permitir que os usuários visualizem suas próprias permissões

Exemplo de políticas de Amazon Rekognition Custom Labels

Você pode criar políticas baseadas em identidade para Amazon Rekognition Custom Labels. Para obter mais informações, consulte Segurança.

Exemplo 1: Permitir que um usuário acesse os recursos somente para leitura

O exemplo a seguir concede acesso somente para leitura aos recursos do 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": "*"
        }
    ]
```

```
}
```

Exemplo 2: Permitir que um usuário tenha acesso total aos recursos

O exemplo a seguir concede acesso total aos recursos do Amazon Rekognition.

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

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou 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": "*"
        }
    ]
}
```

Exemplos de políticas baseadas em recursos do Amazon Rekognition

Os Amazon Rekognition Custom Labels usam políticas baseadas em recursos, conhecidas como políticas de projeto, para gerenciar permissões de cópia para uma versão modelo.

Uma política de projeto concede ou nega permissão para copiar uma versão do modelo de um projeto de origem para um projeto de destino. Você precisa de uma política de projeto se o projeto de destino estiver em uma AWS conta diferente ou se quiser restringir o acesso a uma AWS conta. Por exemplo, talvez queira negar permissões de cópia para uma função específica do IAM. Para obter mais informações, consulte Copiar um modelo.

Dar permissão para copiar uma versão do modelo

O exemplo a seguir permite que a entidade principal `arn:aws:iam::123456789012:role/Admin` copie a versão do modelo `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"
    }
  ]
}
```

Solução de problemas de identidade e acesso do Amazon Rekognition

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com o Amazon Rekognition e o IAM.

Tópicos

- Não estou autorizado a realizar uma ação no Amazon Rekognition
- Não estou autorizado a realizar iam: PassRole
- Sou administrador e quero permitir que outras pessoas acessem o Amazon Rekognition
- Quero permitir que pessoas fora da minha AWS conta acessem meus recursos do Amazon Rekognition

Não estou autorizado a realizar uma ação no Amazon Rekognition

Se isso AWS Management Console indicar que você não está autorizado a realizar uma ação, entre em contato com o administrador para obter ajuda. Caso seu administrador seja a pessoa que forneceu suas credenciais de início de sessão.

O erro de exemplo a seguir ocorre quando o usuário `mateojackson` do IAM tenta usar o console para visualizar detalhes sobre um *`widget`*, mas não tem as permissões `rekognition:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 rekognition:GetWidget on resource: my-example-widget
```

Neste caso, Mateo pede ao administrador para atualizar suas políticas para permitir a ele o acesso ao recurso *`my-example-widget`* usando a ação `rekognition:`*`GetWidget`*.

Não estou autorizado a realizar iam: PassRole

Se você receber um erro informando que não está autorizado a realizar a ação `iam:PassRole`, suas políticas devem ser atualizadas para permitir que você passe uma função para o Amazon Rekognition.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando um usuário do IAM chamado `marymajor` tenta usar o console para realizar uma ação no Amazon Rekognition. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Sou administrador e quero permitir que outras pessoas acessem o Amazon Rekognition

Para permitir que outras pessoas acessem o Amazon Rekognition, você deve conceder permissão às pessoas ou aplicações que precisam de acesso. Se você estiver usando o AWS IAM Identity Center para gerenciar pessoas e aplicações, atribua conjuntos de permissões a usuários ou grupos para definir o nível de acesso. Os conjuntos de permissões criam e atribuem automaticamente políticas do IAM aos perfis do IAM associados à pessoa ou aplicação. Para ter mais informações, consulte Conjuntos de permissões no Guia do usuário do AWS IAM Identity Center .

Se você não estiver usando o Centro de Identidade do IAM, deverá criar entidades do IAM (usuários ou perfis) para as pessoas ou aplicações que precisam de acesso. Em seguida, você deve anexar uma política à entidade que concede a ela as permissões corretas no Amazon Rekognition. Depois que as permissões forem concedidas, forneça as credenciais ao usuário ou desenvolvedor da aplicação. Eles usarão essas credenciais para acessar AWS. Para saber mais sobre como

criar grupos, políticas, permissões e usuários do IAM, consulte Identidades do IAM e Políticas e permissões no IAM no Guia do usuário do IAM.

Quero permitir que pessoas fora da minha AWS conta acessem meus recursos do Amazon Rekognition

É possível criar um perfil que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o Amazon Rekognition oferece suporte a esses recursos, consulte Como o Amazon Rekognition funciona com o IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Proteção de dados no Amazon Rekognition

O modelo de responsabilidade AWS compartilhada O modelo se aplica à proteção de dados no Amazon Rekognition. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre a privacidade de dados, consulte as Data Privacy FAQ. Para obter mais informações sobre a proteção de dados na Europa, consulte a postagem do blog AWS Shared Responsibility Model and RGPD no Blog de segurança da AWS .

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os recursos. AWS Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como trabalhar com CloudTrail trilhas no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para obter mais informações sobre os endpoints FIPS disponíveis, consulte Federal Information Processing Standard (FIPS) 140-3.

É altamente recomendável que nunca sejam colocadas informações confidenciais ou sigilosas, como endereços de e-mail de clientes, em tags ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com o Rekognition ou Serviços da AWS outro usando o console, a API ou. AWS CLI AWS SDKs Quaisquer dados inseridos em tags ou em campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é fortemente recomendável que não sejam incluídas informações de credenciais no URL para validar a solicitação nesse servidor.

Criptografia de dados

As informações a seguir explicam onde o Amazon Rekognition usa criptografia de dados para proteger seus dados.

Criptografia em repouso

Amazon Rekognition Image

Imagens

As imagens passadas para as operações da API Amazon Rekognition podem ser armazenadas e usadas para melhorar o serviço, a menos que você tenha optado por não participar visitando a página de política de exclusão de serviços de IA e seguindo o processo explicado nela. As imagens armazenadas são criptografadas em repouso (Amazon S3) usando o AWS Key Management Service (SSE-KMS).

Coleções

Para operações de comparação de faces que armazenam informações em uma coleção, o algoritmo de detecção subjacente primeiro detecta as faces na imagem de entrada, extrai um vetor para cada face e depois armazena os vetores faciais na coleção. O Amazon Rekognition usa esses vetores faciais ao realizar a comparação facial. Os vetores faciais são armazenados como uma matriz de flutuadores e criptografados em repouso.

Amazon Rekognition Video

Vídeos

Para analisar um vídeo, o Amazon Rekognition copia seus vídeos no serviço para processamento. O vídeo pode ser armazenado e usado para melhorar o serviço, a menos que você tenha optado por não participar visitando a página de política de exclusão de serviços de IA e seguindo o processo explicado lá. Os vídeos são criptografados em repouso (Amazon S3) usando o AWS Key Management Service (SSE-KMS).

Amazon Rekognition Custom Labels

Os Amazon Rekognition Custom Labels criptografam seus dados em repouso.

Imagens

Para treinar seu modelo, as Amazon Rekognition Custom Labels fazem uma cópia de suas imagens de treinamento e teste de origem. As imagens copiadas são criptografadas em repouso no Amazon Simple Storage Service (S3) usando criptografia do lado do servidor com uma AWS KMS key chave KMS fornecida por você ou de sua propriedade. AWS Os Amazon Rekognition Custom Labels só

oferecem suporte a chaves KMS simétricas. Suas imagens de origem não são afetadas. Para obter mais informações, consulte Treinamento de um modelo de Amazon Rekognition Custom Labels.

Modelos da

Por padrão, os Amazon Rekognition Custom Labels criptografam modelos treinados e arquivos de manifesto armazenados em buckets do Amazon S3 usando criptografia do lado do servidor com um Chave pertencente à AWS. Para obter mais informações, consulte Proteger dados usando a criptografia no lado do servidor. Os resultados do treinamento são gravados no bucket especificado no parâmetro `OutputConfig` de entrada paraCreateProjectVersion. Os resultados do treinamento são criptografados usando as configurações de criptografia definidas para o bucket (`OutputConfig`).

Bucket de console

O console Amazon Rekognition Custom Labels cria um bucket do Amazon S3 (bucket de console) que você pode usar para gerenciar seus projetos. O bucket do console é criptografado usando a criptografia padrão do Amazon S3. Para obter mais informações, consulte Criptografia padrão do Amazon Simple Storage Service para buckets do S3. Se você estiver usando sua própria chave KMS, configure o bucket do console depois que ele for criado. Para obter mais informações, consulte Proteger dados usando a criptografia no lado do servidor. Os Amazon Rekognition Custom Labels bloqueiam o acesso público ao bucket do console.

Rekognition Face Liveness

Todos os dados relacionados à sessão armazenados na conta do serviço Rekognition Face Liveness são totalmente criptografados quando em repouso. Por padrão, as imagens de referência e auditoria são criptografadas usando AWS uma chave própria na conta de serviço. No entanto, você pode optar por fornecer suas próprias AWS KMS chaves para criptografar essas imagens.

Criptografia em trânsito

Os endpoints da API Amazon Rekognition só oferecem suporte a conexões seguras por HTTPS. Toda a comunicação é criptografada com Transport Layer Security (TLS).

Gerenciamento de chaves

Você pode usar o AWS Key Management Service (KMS) para gerenciar chaves para as imagens e vídeos de entrada que você armazena nos buckets do Amazon S3. Para obter mais informações, consulte os conceitos do AWS Key Management Service.

Criptografia de chave gerenciada pelo cliente para vivacidade facial

A CreateFaceLivenessSessionAPI usa um `KmsKeyId` parâmetro opcional. Você pode fornecer o `id` da chave KMS que você criou em sua conta. Essa chave será usada para criptografar imagens de referência e auditoria obtidas durante a StartFaceLivenessSessionAPI e, durante a GetFaceLivenessSessionResultsAPI, as imagens serão descriptografadas usando essa chave antes de retornar os resultados. Se a CreateFaceLivenessSession solicitação incluir um OutputConfig, as imagens de referência e auditoria serão carregadas nos caminhos especificados do Amazon S3. Recomendamos habilitar a criptografia do lado do servidor (SSE-S3) em seus buckets do Amazon S3 para que os dados continuem criptografados em repouso.

Quando você fornece seu próprio ID de AWS KMS chave, o serviço Rekognition Face Liveness obtém permissão para usar a chave gerenciada pelo cliente em nome do diretor que invoca o. APIs Os diretores (usuários ou funções) usados para invocar o back-end APIs do cliente (APIs`CreateFaceLivenessSession`e`GetFaceLivenessSessionResults`) devem ter acesso para realizar o seguinte:

- kms:DescribeKey
- kms:GenerateDataKey
- kms:Decrypt

Privacidade do tráfego entre redes

Um endpoint da Amazon Virtual Private Cloud (Amazon VPC) para o Amazon Rekognition é uma entidade lógica dentro de uma VPC que permite conectividade somente com o Amazon Rekognition. A Amazon VPC encaminha solicitações para o Amazon Rekognition e encaminha as respostas de volta para a VPC. Para obter mais informações, consulte Endpoints da VPC no Guia do usuário da Amazon VPC. Para obter informações sobre o uso de endpoints do Amazon VPC com o Amazon Rekognition, consulte Usando o Amazon Rekognition com endpoints da VPC da Amazon.

Usando o Amazon Rekognition com endpoints da VPC da Amazon

Se você usar a Amazon Virtual Private Cloud (Amazon VPC) para hospedar os recursos da AWS, poderá estabelecer uma conexão privada entre a VPC e o Amazon Rekognition. É possível usar essa conexão para habilitar o Amazon Rekognition a se comunicar com os seus recursos na VPC sem passar pela Internet pública.

O Amazon VPC é um serviço da AWS que você pode usar para iniciar recursos da AWS em uma rede virtual que você define. Com a VPC, você tem controle sobre as configurações de rede, como o intervalo de endereços IP, sub-redes, tabelas de rotas e gateways de rede. Com VPC endpoints, a rede da AWS lida com o roteamento entre a VPC e os serviços da AWS.

Para conectar sua VPC ao Amazon Rekognition, você define um endpoint de interface VPC para o Amazon Rekognition. Um endpoint de interface é uma interface de rede elástica com um endereço IP privado que serve como ponto de entrada para o tráfego destinado a um serviço compatível AWS . O endpoint fornece conectividade confiável e escalável ao Amazon Rekognition e não requer um gateway da Internet, uma instância de tradução de endereço de rede (NAT) ou uma conexão VPN. Para obter mais informações, consulte O que é a Amazon VPC? no Guia do usuário da Amazon VPC.

Os endpoints de interface VPC são habilitados pela AWS. PrivateLink Essa tecnologia da AWS permite a comunicação privada entre os serviços da AWS usando uma interface de rede elástica com endereços IP privados.

> **Note**
>
> Todos os endpoints do Amazon Rekognition Federal Information Processing Standard (FIPS) são suportados pela AWS. PrivateLink

Criar endpoints do Amazon VPC para o Amazon Rekognition

Você pode criar dois tipos de endpoints do Amazon VPC para usar com o Amazon Rekognition.

- Você pode criar dois tipos de endpoints do Amazon VPC para usar com o Amazon Rekognition. Para a maioria dos usuários, esse é o tipo mais adequado de VPC endpoint.
- Um endpoint VPC para operações do Amazon Rekognition com endpoints que estão em conformidade com o padrão do governo dos EUA FIPS (Federal Information Processing Standard, Padrão federal de processamento de informações) Publicação 140-2.

Para começar a usar o Amazon Rekognition na VPC, use o console da Amazon VPC para criar um endpoint da VPC de interface para o Amazon Rekognition. Para obter instruções, consulte o procedimento "Como criar um endpoint de interface para um serviço da AWS usando o console" em Criar um endpoint de interface. Observe as seguintes etapas do procedimento:

- Etapa 3: para a categoria Serviço, escolha os serviços da AWS.
- Etapa 4: para Nome do serviço, escolha uma das seguintes opções:
  - com.amazonaws.region.rekognition: cria um endpoint da VPC para operações do Amazon Rekognition.
  - com.amazonaws.region.rekognition-fips: cria um endpoint da VPC para operações do Amazon Rekognition com endpoints que estão em conformidade com o padrão do governo dos EUA FIPS (Federal Information Processing Standard) Publicação 140-2.

Para obter mais informações, consulte Conceitos básicos no Guia do usuário da Amazon VPC.

Criar uma política de endpoint VPC para o Amazon Rekognition

Você pode criar uma política para os endpoints do Amazon VPC para o Amazon Rekognition para especificar o seguinte:

- A entidade principal que pode realizar ações.
- As ações que podem ser realizadas.
- Os recursos aos quais as ações podem ser aplicadas.

Para obter mais informações, consulte Controlar o acesso a serviços com endpoint da VPCs no Manual do usuário da Amazon VPC.

O exemplo de política a seguir permite que os usuários que se conectam ao Amazon Rekognition por meio do endpoint da VPC chamem a operação da API `DetectFaces`. A política impede que os usuários realizem outras operações da API Amazon Rekognition por meio do endpoint da VPC.

Os usuários ainda podem chamar outras operações de API Amazon Rekognition de fora da VPC. Para obter informações sobre como negar acesso a operações de API Amazon Rekognition que estejam fora da VPC, consulte Políticas baseadas em identidade do Amazon Rekognition.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "rekognition:DetectFaces"
```

```
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Principal": "*"
        }
    ]
}
```

Para modificar a política endpoint da VPC para o Amazon Rekognition

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.
2. Se você ainda não tiver criado o endpoint para o Amazon Rekognition, selecione Criar endpoint. Em seguida, selecione com.amazonaws. *Region*.rekognition e escolha Criar endpoint.
3. No painel de navegação, escolha Endpoints.
4. Selecione o com.amazonaws. ***Region***.rekognition endpoint e escolha a guia Política na metade inferior da tela.
5. Selecione Edit policy (Editar política) e faça as alterações na política.

Validação de conformidade para o Amazon Rekognition

Auditores terceirizados avaliam a segurança e a conformidade do Amazon Rekognition como parte de vários programas de conformidade. AWS Isso inclui SOC, PCI, FedRAMP, HIPAA e outros.

Para obter uma lista de AWS serviços no escopo de programas de conformidade específicos, consulte AWS Services in Scope by Compliance Program. Para obter informações gerais, consulte Programas de conformidade da AWS.

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Download de relatórios no AWS Artifact.

Sua responsabilidade de conformidade ao usar o Amazon Rekognition é determinada pela confidencialidade dos seus dados, pelos objetivos de conformidade da sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- Guias de início rápido sobre segurança e conformidade — Esses guias de implantação discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos focados em segurança e conformidade em. AWS

- Documento técnico sobre arquitetura para segurança e conformidade com a HIPAA — Este whitepaper descreve como as empresas podem usar para criar aplicativos compatíveis com a HIPAA. AWS
- AWS Recursos de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- AWS Config— Esse AWS serviço avalia se suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- AWS Security Hub— Esse AWS serviço fornece uma visão abrangente do seu estado de segurança interno, AWS que ajuda você a verificar sua conformidade com os padrões e as melhores práticas do setor de segurança.

Resiliência no Amazon Rekognition

A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

Além da infraestrutura AWS global, o Amazon Rekognition oferece vários recursos para ajudar a suportar suas necessidades de resiliência e backup de dados.

Análise de configuração e vulnerabilidade no Amazon Rekognition

A configuração e os controles de TI são uma responsabilidade compartilhada entre a AWS e você, nosso cliente. Para obter mais informações, consulte o modelo de responsabilidade AWS compartilhada.

Prevenção do problema do substituto confuso entre serviços

Em AWS, a representação entre serviços pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado de modo a

atuar nos recursos de outro cliente, mesmo sendo que ele não deveria ter as permissões adequadas, resultando em um problema de “confused deputy”.

Para evitar isso, AWS fornece ferramentas que ajudam você a proteger seus dados para todos os serviços com diretores de serviços que receberam acesso aos recursos em sua conta.

Recomendamos o uso das chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` nas políticas de recursos para limitar as permissões que o Amazon Rekognition concede a outro serviço para o recurso.

Se o valor de `aws:SourceArn` não contiver o ID da conta, como um ARN de bucket do Amazon S3, você deverá usar as duas chaves para limitar as permissões. Se você usar as duas chaves e o valor `aws:SourceArn` contiver a ID da conta, o valor `aws:SourceAccount` e a conta no valor `aws:SourceArn` deverão usar a mesma ID da conta quando usados na mesma declaração de política.

Use `aws:SourceArn` se quiser apenas um recurso associado a acessibilidade de serviço. Use `aws:SourceAccount` se quiser permitir que qualquer recurso nessa conta seja associado ao uso entre serviços.

O valor de `aws:SourceArn` deve ser o ARN do recurso usado pelo Rekognition, que é especificado com o seguinte formato: `arn:aws:rekognition:region:account:resource`.

O valor de `arn:User ARN` deve ser o ARN do usuário que chamará a operação de análise de vídeo (o usuário que assume uma função).

A abordagem recomendada para o problema do deputado confuso é usar a chave de contexto de condição global do `aws:SourceArn` com o ARN completo do recurso.

Se você não souber o ARN completo do recurso ou se estiver especificando vários recursos, use a chave com caracteres curinga `aws:SourceArn` (`*`) para as partes desconhecidas do ARN. Por exemplo, `arn:aws:rekognition:*:111122223333:*`.

Para se proteger contra o confuso problema do deputado, execute as seguintes etapas:

1. No painel de navegação do console do IAM, escolha a opção Perfis . O console exibirá as funções da sua conta atual.
2. Escolha o nome da função que você deseja modificar. A função que você modifica deve ter a política de AmazonRekognitionServiceRolepermissões. Selecione a guia Relações de confiança.
3. Escolha Editar política de confiança.

4. Na página Editar política de confiança, substitua a política JSON padrão por uma política que utilize uma ou ambas as chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount`. Veja os exemplos de políticas a seguir.
5. Escolha Atualizar política.

Os exemplos a seguir são políticas de confiança que mostram como você pode usar as chaves de contexto de condição global `aws:SourceAccount` e `aws:SourceArn` e as chaves de contexto no Amazon Rekognition para evitar o confuso problema do delegado.

Se você estiver trabalhando armazenado e transmitindo vídeos, poderá usar uma política como a seguinte em perfil do 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 você estiver trabalhando exclusivamente com vídeo armazenado, poderá usar uma política como a seguinte em seu perfil do IAM (observe que não é necessário incluir o argumento `StringLike` que especifica o `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"
            }
         }
      }
   ]
}
```

Segurança da infraestrutura no Amazon Rekognition

Como um serviço gerenciado, o Amazon Rekognition é protegido pela segurança de rede global. AWS Para obter informações sobre serviços AWS de segurança e como AWS proteger a infraestrutura, consulte AWS Cloud Security. Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte Proteção de infraestrutura no Security Pillar AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar o Amazon Rekognition pela rede. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Monitorar o Amazon Rekognition

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho do Amazon Rekognition e de suas outras soluções da AWS. A AWS fornece as seguintes ferramentas de monitoramento para observar o Rekognition, relatar quando algo está errado e tomar ações automáticas quando apropriado:

- A Amazon CloudWatch monitora seus AWS recursos e os aplicativos em que você executa AWS em tempo real. Você pode coletar e rastrear métricas, criar painéis personalizados e definir alarmes que o notificam ou que realizam ações quando uma métrica especificada atinge um limite definido. Por exemplo, você pode CloudWatch rastrear o uso da CPU ou outras métricas de suas EC2 instâncias da Amazon e iniciar automaticamente novas instâncias quando necessário. Para obter mais informações, consulte o Guia CloudWatch do usuário da Amazon.
- O Amazon CloudWatch Logs permite que você monitore, armazene e acesse seus arquivos de log de EC2 instâncias da Amazon e de outras fontes. CloudTrail CloudWatch Os registros podem monitorar as informações nos arquivos de log e notificá-lo quando determinados limites forem atingidos. É possível também arquivar seus dados de log em armazenamento resiliente. Para obter mais informações, consulte o Guia do usuário do Amazon CloudWatch Logs.
- A Amazon EventBridge pode ser usada para automatizar seus AWS serviços e responder automaticamente a eventos do sistema, como problemas de disponibilidade de aplicativos ou alterações de recursos. Os eventos dos AWS serviços são entregues quase EventBridge em tempo real. Você pode escrever regras simples para determinar quais eventos são do seu interesse, e as ações automatizadas a serem tomadas quando um evento corresponder à regra. Para obter mais informações, consulte o Guia EventBridge do usuário da Amazon.
- AWS CloudTrailcaptura chamadas de API e eventos relacionados feitos por ou em nome de sua AWS conta e entrega os arquivos de log para um bucket do Amazon S3 que você especificar. Você pode identificar quais usuários e contas ligaram AWS, o endereço IP de origem a partir do qual as chamadas foram feitas e quando elas ocorreram. Para obter mais informações, consulte o Guia do usuário do AWS CloudTrail.

Monitorando o Rekognition com a Amazon CloudWatch

Com CloudWatch, você pode obter métricas para operações individuais do Rekognition ou métricas globais do Rekognition para sua conta. Você pode usar métricas para monitorar a integridade de sua solução baseada no Rekognition e configurar alarmes para notificá-lo quando uma ou mais

métricas estiverem fora de um limite definido. Por exemplo, você pode ver métricas para o número de erros de servidor que ocorreram ou métricas para o número de faces que foram detectadas. Você também pode ver as métricas do número de vezes que uma operação específica do Rekognition foi bem-sucedida. Para ver as métricas, você pode usar a Amazon CloudWatch AWS Command Line Interface, a Amazon ou a CloudWatch API.

Você também pode ver métricas agregadas, por um determinado período de tempo, usando o console do Rekognition. Para obter mais informações, consulte Exercício 4: Ver métricas agregadas (console).

Usando CloudWatch métricas para Rekognition

Para usar métricas, você deve especificar as seguintes informações:

- A dimensão da métrica ou nenhuma dimensão. Uma dimensão é um par nome/valor, que ajuda a identificar com exclusividade uma métrica. O reconhecimento tem uma dimensão, denominada Operação. Ele fornece métricas para uma operação específica. Se você não especificar uma dimensão, a métrica terá como escopo todas as operações do Rekognition em sua conta.
- O nome da métrica, como `UserErrorCount`.

Você pode obter dados de monitoramento do Rekognition usando AWS Management Console a, a ou a API. AWS CLI CloudWatch Você também pode usar a CloudWatch API por meio de um dos kits de desenvolvimento de software da Amazon AWS (SDKs) ou das ferramentas de CloudWatch API. O console exibe uma série de gráficos com base nos dados brutos da CloudWatch API. Dependendo das necessidades, você pode preferir usar os gráficos exibidos no console ou recuperados da API.

A lista a seguir mostra alguns usos comuns para as métricas. Essas são sugestões para você começar, e não uma lista abrangente.

Como eu faço para...	Métricas relevantes
Como acompanho o número de faces reconhecidas?	Monitore a estatística `Sum` da métrica `DetectedFaceCount` .
Como sei se meu aplicativo atingiu o número máximo de solicitações por segundo?	Monitore a estatística `Sum` da métrica `ThrottledCount` .

Como eu faço para...	Métricas relevantes
Como posso monitorar os erros de solicitação?	Use a estatística `Sum` da métrica `UserErrorCount` .
Como posso encontrar o número total de solicitações?	Use as estatísticas `ResponseTime` e `DataSamples` da métrica `ResponseTime` . Isso inclui qualquer solicitação que resulte em um erro. Se você quiser ver apenas as chamadas de operação bem-sucedidas, use a métrica `SuccessfulRequestCount` .
Como posso monitorar a latência das chamadas de operação `Rekognition` ?	Use a métrica `ResponseTime` .
Como posso monitorar quantas vezes faces foram adicionadas à `IndexFaces` com sucesso às coleções do Rekognition?	Monitore a estatística `Sum` com a métrica `SuccessfulRequestCount` e a operação `IndexFaces` . Use a dimensão `Operation` para selecionar a operação e a métrica.

Você deve ter as CloudWatch permissões apropriadas para monitorar o Rekognition com. CloudWatch Para obter mais informações, consulte Autenticação e controle de acesso para o Amazon CloudWatch.

Métricas de reconhecimento de acesso

Os exemplos a seguir mostram como acessar as métricas do Rekognition usando o console, o e CloudWatch a API. AWS CLI CloudWatch

Para visualizar métricas (console)

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. Escolha Metrics, escolha a guia All Metrics e, em seguida, escolha Rekognition.
3. Escolha Metrics with no dimensions e, em seguida, escolha uma métrica.

   Por exemplo, escolha a métrica DetectedFace para medir quantas faces foram detectadas.
4. Escolha um valor para o intervalo de datas. A contagem de métricas exibidas no gráfico.

Para visualizar as métricas de chamadas bem-sucedidas da operação **`DetectFaces`** feitas durante um período (CLI).

- Abra o AWS CLI e digite o seguinte comando:

  ```
  aws cloudwatch get-metric-statistics --metric-name
  SuccessfulRequestCount --start-time 2017-1-1T19:46:20 --end-time
  2017-1-6T19:46:57 --period 3600 --namespace AWS/Rekognition --
  statistics Sum --dimensions Name=Operation,Value=DetectFaces --region
  us-west-2
  ```

  Este exemplo mostra as chamadas de operação `DetectFaces` bem-sucedidas feitas ao longo de um período. Para obter mais informações, consulte get-metric-statistics.

Para acessar métricas (CloudWatch API)

- Chame `GetMetricStatistics`. Para obter mais informações, consulte a Amazon CloudWatch API Reference.

Criar um alarme

Você pode criar um CloudWatch alarme que envia uma mensagem do Amazon Simple Notification Service (Amazon SNS) quando o alarme muda de estado. Um alarme observa uma única métrica ao longo de um período especificado por você e realiza uma ou mais ações com base no valor da métrica relativo a um determinado limite ao longo de vários períodos. A ação é uma notificação enviada a um tópico do Amazon SNS ou a uma política de Auto Scaling.

Os alarmes invocam ações somente para mudanças de estado sustentadas. CloudWatch os alarmes não invocam ações simplesmente porque estão em um estado específico. O estado deve ter sido alterado e mantido por um período especificado.

Para definir um alarme (console)

1. Faça login no AWS Management Console e abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. Escolha Create Alarm. Isso executa o Create Alarm Wizard.
3. Na lista de métricas Metrics with no dimensions, escolha Rekognition Metrics e escolha uma métrica.

Por exemplo, escolha DetectedFaceCount a fim de definir um alarme para um número máximo de faces detectadas.

4. Na área Time Range, selecione um valor de intervalo de datas que inclua as operações de detecção de faces chamadas por você. Escolha Próximo.
5. Preencha o Name e a Description. Em Whenever, escolha **>=** e digite um valor máximo de sua escolha.
6. Se você quiser CloudWatch enviar um e-mail quando o estado do alarme for atingido, em Sempre que este alarme:, escolha Estado é ALARME. Para enviar alarmes para um tópico existente do Amazon SNS, em Enviar notificação para:, escolha um tópico existente do SNS. Para definir o nome e os endereços de e-mail para uma nova lista de assinaturas de e-mail, escolha Criar tópico para CloudWatch salvar a lista e exibi-la no campo para que você possa usá-la para definir futuros alarmes.

   > **Note**
   >
   > Se você usar Criar tópico para criar um novo tópico do Amazon SNS, os endereços de e-mail devem ser verificados antes que os destinatários pretendidos recebam as notificações. O Amazon SNS envia e-mails somente quando o alarme entra em um estado de alarme. Se essa alteração no estado de alarme acontecer antes dos endereços de e-mail serem verificados, os destinatários desejados não receberão uma notificação.

7. Visualize o alarme na seção Prévia do alarme. Escolha Create Alarm.

Para definir um alarme (AWS CLI)

- Abra o AWS CLI e digite o seguinte comando. Altere o valor do parâmetro `alarm-actions` para referenciar um tópico do Amazon SNS que você criou anteriormente.

```
aws cloudwatch put-metric-alarm --alarm-name UserErrors --
alarm-description "Alarm when more than 10 user errors occur"
--metric-name UserErrorCount --namespace AWS/Rekognition --
statistic Average --period 300 --threshold 10 --comparison-
operator GreaterThanThreshold --evaluation-periods 2 --alarm-actions
arn:aws:sns:us-west-2:111111111111:UserError --unit Count
```

Este exemplo mostra como criar um alerta para quando mais de 10 erros de usuário ocorrem em 5 minutos. Para obter mais informações, consulte put-metric-alarm.

Para definir um alarme (CloudWatch API)

- Chame `PutMetricAlarm`. Para obter mais informações, consulte Amazon CloudWatch API Reference.

CloudWatchmétricas para Rekognition

Esta seção contém informações sobre as CloudWatch métricas da Amazon e a dimensão Operação disponíveis para o Amazon Rekognition.

Você também pode ver uma visão agregada das métricas do Rekognition no console do Rekognition. Para obter mais informações, consulte Exercício 4: Ver métricas agregadas (console).

CloudWatch métricas para Rekognition

A tabela a seguir resume as métricas do Rekognition.

Métrica	Descrição
SuccessfulRequestCount	O número de solicitações bem-sucedidas. O intervalo de códigos de resposta para uma solicitação bem-sucedida vai de 200 até 299. Unidade: contagem Estatísticas válidas: `Sum,Average`
ThrottledCount	O número de solicitações limitadas. O Rekognition limita uma solicitação quando recebe mais solicitações do que o limite de transações por segundo definido para sua conta. Se o limite definido para a conta for frequentemente excedido, você poderá solicitar um aumento no limite. Para solicitar um aumento, consulte Limites de serviço da AWS. Unidade: contagem Estatísticas válidas: `Sum,Average`

Métrica	Descrição
ResponseTime	O tempo em milissegundos para o Rekognition computar a resposta. Unidades: 1. Contagem de estatísticas `Data Samples` 2. Milissegundos para estatísticas `Average` Estatísticas válidas: `Data Samples,Average` **Note** A métrica `ResponseTime` não está incluída no painel de métricas do Rekognition.
DetectedFaceCount	O número de faces detectadas com a operação `IndexFaces` ou `DetectFaces` . Unidade: contagem Estatísticas válidas: `Sum,Average`
DetectedLabelCount	O número de rótulos detectados com a operação `DetectLabels` . Unidade: contagem Estatísticas válidas: `Sum,Average`
ServerErrorCount	O número de erros do servidor. O intervalo de códigos de resposta para um erro de servidor vai de 500 até 599. Unidade: contagem Estatísticas válidas: `Sum, Average`

Métrica	Descrição
UserErrorCount	O número de erros de usuário (parâmetros inválidos, imagem inválida, sem permissão etc). O intervalo de códigos de resposta para um erro de usuário vai de 400 até 499. Unidade: contagem Estatísticas válidas: `Sum,Average`
MinInferenceUnits	O número mínimo de unidades de inferência especificadas durante a solicitação `StartProjectVersion` . Unidade: contagem Estatísticas válidas: `Average`
MaxInferenceUnits	O número máximo de unidades de inferência especificadas durante a solicitação `StartProjectVersion` . Unidade: contagem Estatísticas válidas: `Average`
DesiredInferenceUnits	O número de unidades de inferência para as quais o Rekognition está aumentando ou diminuindo. Unidade: contagem Estatísticas válidas: `Average`
InServiceInferenceUnits	O número de unidades de inferência que o modelo está usando. Unidade: contagem Estatísticas válidas: `Average` É recomendável usar a estatística Média para obter a média de 1 minuto de quantas instâncias são usadas.

CloudWatch métricas para Rekognition Streaming

O Rekognition também tem um segundo namespace usado para operações de streaming, "Rekognition Streaming". A tabela a seguir resume as métricas do Rekognition Streaming.

Métrica	Descrição
SuccessfulRequestCount	O número de solicitações bem-sucedidas. O intervalo de códigos de resposta para uma solicitação bem-sucedida vai de 200 até 299. Unidade: contagem Estatísticas válidas: `Sum,Average`
CallCount	O número de operações especificadas executadas em sua conta. Estatísticas válidas: `Sum,Average`
ThrottledCount	O número de solicitações limitadas. O Rekognition limita uma solicitação quando recebe mais solicitações do que o limite de transações por segundo definido para sua conta. Se o limite definido para a conta for frequentemente excedido, você poderá solicitar um aumento no limite. Para solicitar um aumento, consulte Limites de serviço da AWS. Unidade: contagem Estatísticas válidas: `Sum,Average`
ServerErrorCount	O número de erros do servidor. O intervalo de códigos de resposta para um erro de servidor vai de 500 até 599. Unidade: contagem Estatísticas válidas: `Sum, Average`
UserErrorCount	O número de erros de usuário (parâmetros inválidos, imagem inválida, sem permissão etc). O intervalo de códigos de resposta para um erro de usuário vai de 400 até 499. Unidade: contagem

Métrica	Descrição
	Estatísticas válidas: `Sum,Average`
ConcurrentSessions	O número de sessões simultâneas que sua conta está usando atualmente. Unidade: contagem Estatísticas válidas: `Max,Average`

CloudWatch dimensão para Rekognition

Para recuperar métricas específicas da operação, use o namespace `Rekognition` e forneça uma dimensão de operação.

Para obter mais informações sobre dimensões, consulte Dimensões no Guia do CloudWatch usuário da Amazon.

CloudWatch dimensão para rótulos personalizados do Rekognition

A tabela a seguir mostra as CloudWatch dimensões disponíveis para uso com as etiquetas personalizadas do Rekognition:

Dimensão	Descrição
ProjectName	O nome do projeto Rekognition Custom Labels que você criou com o `CreateProject` .
VersionName	O nome da versão do projeto Rekognition Custom Labels com a qual você criou `CreateProjectVersion` .

Para obter mais informações sobre dimensões, consulte Dimensões no Guia do CloudWatch usuário da Amazon.

Registro de chamadas da API Amazon Rekognition com AWS CloudTrail

O Amazon Rekognition é AWS CloudTrail integrado ao Amazon Rekognition, um serviço que fornece um registro das ações realizadas por um usuário, função ou serviço no Amazon Rekognition. AWS CloudTrail captura todas as chamadas de API para o Amazon Rekognition como eventos. As chamadas capturadas incluem chamadas do console do Amazon Rekognition e chamadas de código para as operações da API do Amazon Rekognition. Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon S3, incluindo eventos para o Amazon Rekognition. Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita ao Amazon Rekognition, o endereço IP a partir do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre isso CloudTrail, consulte o Guia AWS CloudTrail do usuário.

Informações sobre o Amazon Rekognition em CloudTrail

CloudTrail é ativado em sua AWS conta quando você cria a conta. Quando a atividade ocorre no Amazon Rekognition, essa atividade é registrada CloudTrail em um evento junto AWS com outros eventos de serviço no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes em sua AWS conta. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para um registro contínuo dos eventos em sua AWS conta, incluindo eventos do Amazon Rekognition, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as regiões da AWS. A trilha registra eventos de todas as regiões na AWS partição e entrega os arquivos de log ao bucket do Amazon S3 que você especificar. Além disso, você pode configurar outros AWS serviços para analisar e agir com base nos dados de eventos coletados nos CloudTrail registros. Para obter mais informações, consulte:

- Visão Geral para Criar uma Trilha
- CloudTrail Serviços e integrações compatíveis
- Configurando notificações do Amazon SNS para CloudTrail
- Recebendo arquivos de CloudTrail log de várias regiões e recebendo arquivos de CloudTrail log de várias contas

Todas as ações do Amazon Rekognition são CloudTrail registradas e documentadas na referência da API Amazon Rekognition. Por exemplo, chamadas para o `CreateCollection` `CreateStreamProcessor` e `DetectCustomLabels` as ações geram entradas nos arquivos de CloudTrail log.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar o seguinte:

- Se a solicitação foi feita com credenciais de usuário root ou AWS Identity and Access Management (IAM).
- Se a solicitação foi feita com credenciais de segurança temporárias de uma função ou de um usuário federado.
- Se a solicitação foi feita por outro AWS serviço.

Para obter mais informações, consulte Elemento userIdentity do CloudTrail .

Entendendo as entradas do arquivo de log do Amazon Rekognition

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log para um bucket do Amazon S3 que você especificar. CloudTrail os arquivos de log contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros da solicitação e assim por diante. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API, portanto, eles não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra uma entrada de CloudTrail registro com ações para a seguinte API: `StartLabelDetection` `DetectLabels` e.

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

Diretrizes e cotas no Amazon Rekognition

As seções a seguir fornecem diretrizes e cotas ao usar o Amazon Rekognition. Há dois tipos de cotas. As cotas definidas, como o tamanho máximo da imagem, não podem ser alteradas. As Cotas padrão listado na página AWS Service Quotas pode ser alterada seguindo o procedimento descrito na seção Cotas padrão.

Tópicos

- Regiões compatíveis
- Definir cotas
- Cotas padrão

Regiões compatíveis

Para obter uma lista das AWS regiões em que o Amazon Rekognition está disponível, consulte Regiões e endpoints da AWS na Referência geral da Amazon Web Services.

Definir cotas

A seguir está uma lista de limites no Amazon Rekognition que não podem ser alterados. Para obter informações sobre limites que você pode alterar, como limites de transações por segundo (TPS), consulte Cotas padrão.

Para conhecer os limites de Amazon Rekognition Custom Labels, consulte Diretrizes e cotas em Amazon Rekognition Custom Labels.

Amazon Rekognition Image

- O tamanho máximo da imagem armazenada como um objeto do Amazon S3 é limitado a 15 MB.
- A dimensão máxima da imagem para `DetectModerationLabels` é de 10K pixels para largura e altura.
- A dimensão máxima da imagem para `DetectLabels` é de 10K pixels para largura e altura.
- Para ser detectado, um rosto não deve ser menor do que 40 x 40 pixels em uma imagem com resolução de 1920 x 1080 pixels. Imagens com dimensões maiores do que 1920 x 1080 pixels precisam de um tamanho mínimo de face proporcionalmente maior.

- As dimensões mínimas da imagem são 80 pixels para altura e largura. A dimensão mínima da imagem para `DetectProtectiveEquipment` é de 64 pixels para altura e largura.
- A dimensão máxima da imagem `DetectProtectiveEquipment` é de 4096 pixels para largura e altura.
- Para ser detectada por `DetectProtectiveEquipment`, uma pessoa não deve ter menos de 100x100 pixels em uma imagem com 800x1300. Imagens com dimensões superiores a 800x1300 pixels precisarão de um tamanho mínimo de pessoa maior proporcionalmente.
- O tamanho máximo das imagens como bytes brutos passados como parâmetro para uma API é de 5 MB. O limite é de 4 MB para a API `DetectProtectiveEquipment`.
- O Amazon Rekognition oferece suporte aos formatos de imagem PNG e JPEG. Ou seja, as imagens fornecidas por você como entrada para diversas operações de API, como `DetectLabels` e `IndexFaces` devem estar em um dos formatos compatíveis.
- O número máximo de vetores faciais que você pode armazenar em uma única coleção de faces é de 20 milhões.
- O número máximo padrão de vetores de usuário que você pode armazenar em uma única coleção de faces é 10 milhões.
- O máximo de vetores faciais correspondentes que a API de pesquisa retorna é 4096.
- O máximo de vetores de usuário correspondentes que a API de pesquisa retorna é 4096.
- `DetectText` pode detectar até 100 palavras em uma imagem.
- `DetectProtectiveEquipment` pode detectar equipamentos de proteção individual em até 15 pessoas.

Para obter informações sobre as melhores práticas para imagens e comparação facial, consulte Melhores práticas para sensores, imagens de entrada e vídeos.

Análise em massa do Amazon Rekognition Image

- A análise em massa do Amazon Rekognition Image pode analisar lotes de imagens com um tamanho de até 10 mil imagens.
- A análise em massa do Amazon Rekognition Image é compatível com manifestos de entrada com um tamanho de até 50 MB.

Vídeo armazenado no Amazon Rekognition Video

- O Amazon Rekognition Video pode analisar vídeos armazenados de até 10 GB.
- O Amazon Rekognition Video pode analisar vídeos armazenados com até 6 horas de duração.
- O Amazon Rekognition Video suporta no máximo 20 trabalhos simultâneos por conta.
- Os vídeos armazenados devem ser codificados usando o codec H.264. Os formatos de arquivo compatíveis são MPEG-4 e MOV.
- Qualquer API do Amazon Rekognition Video que analise dados de áudio só oferece suporte a codecs de áudio AAC.
- O período de vida útil (TTL – Time To Live) para tokens de paginação é de 24 horas. Os tokens de paginação tokens estão no campo `NextToken` retornado por operações Get, como `GetLabeldetection`.

Streaming de vídeo com o Amazon Rekognition Video

- Um stream de entrada do Kinesis Video pode ser associado a no máximo 1 processador de stream do Amazon Rekognition Video.
- Um stream de saída do Kinesis Data pode ser associado a no máximo 1 processador de stream do Amazon Rekognition Video.
- O stream de entrada do Kinesis Video e o stream de saída do Kinesis Data associados a um processador de stream do Amazon Rekognition Video não podem ser compartilhados por vários processadores.
- Qualquer API do Amazon Rekognition Video que analise dados de áudio só oferece suporte a codecs de áudio ACC.

Cotas padrão

Uma lista de cotas padrão pode ser encontrada em Service Quotas da AWS. Esses limites são padrões e podem ser alterados. Para solicitar um aumento de limite, você cria um caso. Para ver seus limites de cota atuais (valores de cota aplicados), consulte Service Quotas do Amazon Rekognition. Para visualizar seu histórico de utilização do TPS para o Amazon Rekognition Image, APIs consulte a página de Cotas do Amazon Rekognition Service e escolha uma operação de API específica para ver o histórico dessa operação.

Tópicos

- Calcular a alteração da cota de TPS
- Melhores práticas para cotas de TPS
- Crie um caso para alterar as cotas de TPS

Calcular a alteração da cota de TPS

Qual é o novo limite que você está solicitando? As transações por segundo (TPS) são mais relevantes no pico de uma workload esperada. É importante entender o máximo de chamadas de API simultâneas no pico de uma workload e o tempo de resposta (5 a 15 segundos). Observe que 5 segundos deve ser o mínimo. Abaixo estão dois exemplos:

- Exemplo 1: O número máximo de usuários simultâneos de Autenticação Facial (CompareFaces API) que eu espero no início do meu horário mais movimentado é de 1000. Essas respostas serão distribuídas por um período de 10 segundos. Portanto, o TPS necessário é 100 (1000/10) para a CompareFaces API na minha região relevante.
- Exemplo 2: O máximo de chamadas simultâneas de Detecção de Objetos (DetectLabels API) esperadas no início do meu horário mais movimentado é 250. Essas respostas serão distribuídas por um período de 5 segundos. Portanto, o TPS necessário é 50 (250/5) para a DetectLabels API na minha região relevante.

Melhores práticas para cotas de TPS

As melhores práticas recomendadas para transações por segundo (TPS) incluem atenuar picos de tráfego, configurar novas tentativas e configurar recuo exponencial e instabilidade.

1. Tráfego suave e pontiagudo. O tráfego intenso afeta a throughput. Para obter a máxima de throughput para as transações alocadas por segundo (TPS), use uma arquitetura sem servidor de filas ou outro mecanismo para "suavizar" o tráfego e torná-lo mais consistente. Para exemplos de código e referências para processamento de imagens e vídeos em grande escala sem servidor com o Rekognition, consulte Processamento de imagens e vídeos em grande escala com o Amazon Rekognition.
2. Configure novas tentativas. Siga as diretrizes em the section called “Gerenciamento de erros” para configurar novas tentativas para os erros que as permitem.

3. Configure o recuo exponencial e a instabilidade. Configurar o recuo exponencial e a instabilidade ao configurar novas tentativas permite melhorar o throughput que pode ser alcançado. Consulte Tentativas de erro e recuo exponencial em. AWS

Crie um caso para alterar as cotas de TPS

Para criar um caso, acesse Criar caso e responda às seguintes perguntas:

- Você implementou o the section called “Melhores práticas para cotas de TPS” para suavizar seus picos de tráfego e configurar novas tentativas, recuo exponencial e instabilidade?
- Você calculou a alteração de cota de TPS de que precisa? Se não, consulte the section called “Calcular a alteração da cota de TPS”.
- Você verificou seu histórico de uso do TPS para prever com mais precisão suas necessidades futuras? Para visualizar seu histórico de uso do TPS, consulte a página Service Quotas do Amazon Rekognition.
- Qual é o seu caso de uso?
- O APIs que você planeja usar?
- Em quais regiões você planeja usá-los APIs ?
- Você é capaz de distribuir a carga em várias regiões?
- Quantas imagens você processa diariamente?
- Por quanto tempo você espera manter esse volume (é um pico único ou contínuo)?
- Como você é bloqueado pelo limite padrão? Examine a tabela de exceções a seguir para confirmar o cenário que você está enfrentando.

Código de erro	Exceção	Mensagem	O que isso significa?	Pode ser tentado novamente?
Código de status HTTP 400	ProvisionedThroughputExceededException	Taxa provisionada excedida.	Indica controle de utilização. Você pode tentar novamente ou avaliar uma solicitação de	Sim

Código de erro	Exceção	Mensagem	O que isso significa?	Pode ser tentado novamente?
			aumento de limite.	
Código de status HTTP 400	Throttlin gException	Diminuir a velocidad e; aumento repentino na taxa de solicitaç ões.	Você pode estar enviando tráfego intenso e usar o controle de utilização. Você deve moldar o tráfego e torná-lo mais suave e consistente. Em seguida, configure novas tentativa s. Consulte as melhores práticas.	Sim
Código de status HTTP 5xx	Throttlin gException (HTTP 500)	Serviço indisponível	Indica que o back-end está sendo ampliado para dar suporte à ação. Você deve tentar fazer a solicitação novamente.	Sim

Para obter uma compreensão detalhada dos códigos de erro, consulte the section called “Gerenciamento de erros”.

> **Note**
>
> Esses limites dependem da região em que você está. Fazer um caso para alterar um limite afeta a operação de API que você solicita, na região em que você a solicita. Outras operações e regiões da API não são afetadas.

Histórico de documentos do Amazon Rekognition

A tabela a seguir descreve mudanças importantes em cada versão do Guia do desenvolvedor do Amazon Rekognition. Para receber notificações sobre atualizações dessa documentação, é possível inscrever-se em um feed RSS.

- Última atualização da documentação: 15 de junho de 2023

Alteração	Descrição	Data
O Amazon Rekognition agora oferece suporte a novos rótulos de moderação e mais precisão para moderação de conteúdo de imagens	O atributo Moderação de conteúdo do Amazon Rekognition foi aprimorado para melhorar a precisão, a detecção de novos rótulos e a capacidade de identific ar conteúdo animado e/ou ilustrado.	1.º de fevereiro de 2024
O Amazon Rekognition agora oferece suporte à análise de imagens em massa	O Amazon Rekognition agora suporta o processamento de uma grande coleção de imagens de forma assíncron a usando um arquivo de manifesto com a operação. StartMediaAnalysisJob	23 de outubro de 2023
O Amazon Rekognition agora oferece suporte à moderação de conteúdo personalizado com adaptadores	O Amazon Rekognition agora oferece suporte à maior precisão da API usando adaptadores que ampliam os recursos DetectModerationLa bels dos modelos existentes de aprendizado profundo do Rekognition.	12 de outubro de 2023

O Rekognition agora oferece suporte a vetores de usuário com coleções	As coleções de faces do Rekognition agora suportam a criação de vetores do usuário. Os vetores do usuário agregam vários vetores faciais do mesmo usuário, melhorando a precisão com representações mais robustas de um usuário.	12 de junho de 2023
As ações para envolver o gerenciamento de usuários foram adicionadas às seguintes políticas gerenciadas: AmazonRekognitionReadOnlyAccess	O Amazon Rekognition adicionou as seguintes ações às políticas gerenciadas `AmazonRekognitionReadOnlyAccess`: `ListUsers`, `SearchUsers`, `SearchUsersByImage`	12 de junho de 2023
Amazon Rekognition Image agora pode inferir a direção do olhar	Foram feitas melhorias nas operações de detecção facial do Amazon Rekognition Image, que agora podem inferir a direção do olhar em uma face detectada.	31 de maio de 2023

API de moderação de conteúdo do Rekognition aprimorada	O Rekognition aprimorou o modelo de moderação de conteúdo para moderação de imagens e vídeos. A melhoria expande significativamente a detecção de conteúdo explícito, violento e sugestivo . Agora, os clientes podem detectar conteúdo explícito e violento com maior precisão para melhorar a experiência do usuário final, proteger sua identidade de marca e garantir que todo o conteúdo esteja em conformidade com as regulamentações e políticas do setor.	9 de maio de 2023
O Amazon Rekognition Image agora pode detectar faces ocluídas	O Amazon Rekognition Image agora pode detectar a oclusão de faces. Um novo FaceOcclu ded atributo é retornado pelo Amazon Rekognition DetectFaces IndexFaces APIs Image's and, que indica se o rosto em uma imagem foi parcialmente capturado ou não está totalmente visível devido à sobreposição de objetos, roupas e partes do corpo.	5 de maio de 2023

O Rekognition agora pode detectar a vivacidade facial	Agora, o Amazon Rekogniti on Video pode ser usado para detectar a vivacidade de um vídeo, verificando se um usuário na frente da câmera está fisicamente presente. O detector Face Liveness também detecta ataques falsos apresentados a uma câmera ou tentando contornar uma câmera.	11 de abril de 2023
Atualização para o Amazon Rekognition Video.	Agora, o Amazon Rekogniti on Video pode detectar mais rótulos e retornar mais informações sobre atributos de imagens e rótulos. A GetLabelDetection API agora retorna informações sobre aliases e categorias. As informações da etiqueta retornada podem ser filtradas com opções de filtro inclusiva s e exclusivas. Os resultado s podem ser agregados por timestamps ou segmentos de vídeo.	7 de dezembro de 2022

Atualização para o Amazon Rekognition Image.	O Amazon Rekognition Image agora pode detectar mais rótulos e agora retorna mais informações sobre atributos de imagens e rótulos. A DetectLabels API agora retorna informações sobre aliases, categorias e proprieda des da imagem, como cores dominantes. As informações da etiqueta retornada podem ser filtradas com opções de filtro inclusivas e exclusivas.	11 de novembro de 2022
As ações ProjectPolicy e a cópia personalizada do modelo de etiquetas foram adicionadas às seguintes políticas gerenciadas: AmazonRekognitionR eadOnlyAccess	O Amazon Rekognition adicionou as seguintes ações à política gerenciad a `AmazonRekognitionR eadOnlyAccess` : `ListProjectPolicies`	21 de julho de 2022
As ações ProjectPolicy e a cópia do modelo de etiquetas personalizadas foram adicionadas às seguintes políticas gerenciad as: AmazonRekognitionF ullAccess, AmazonRek ognitionCustomLabelsFullAcc ess	O Rekognition adicionou as seguintes ações às políticas gerenciadas `AmazonRek ognitionCustomLabe lsFullAccess` e `AmazonRekognitionF ullAccess` : `CopyProje ctVersion` , `PutProjec tPolicy` , `ListProje ctPolicies` , `DeletePro jectPolicy`	21 de julho de 2022

Amazon Rekognition Video agora pode detectar rótulos em streaming de vídeo	O Amazon Rekognition Video pode detectar rótulos como animais de estimação e pacotes em streaming de vídeo. Isso é feito usando as configurações `ConnectedHome` dos processadores de stream criados com a operação `CreateStreamProcessor`.	28 de abril de 2022
A referência da API foi retirada do guia do desenvolvedor do Amazon Rekognition	A referência da API Amazon Rekognition já está disponível em Referência da API Amazon Rekognition.	24 de fevereiro de 2022
Atualização do gerenciamento do conjunto de dados para as seguintes políticas gerenciadas: Política gerenciada pela AWS: AmazonRekognitionReadOnlyAccess,Política gerenciada pela AWS: AmazonRekognitionFullAccess,Política gerenciada pela AWS: AmazonRekognitionCustomLabelsFullAccess	O Amazon Rekognition adicionou as seguintes ações `AmazonRekognitionReadOnlyAccess` às `AmazonRekognitionFullOnlyAccess` políticas ,, e `CreateDataset` gerenciou:,,`AmazonRekognitionCustomLabelsFullAccess` ,,, `ListDatasetEntries` `ListDatasetLabels` `DescribeDataset` `UpdateDatasetEntries` `DistributeDatasetEntries` DeleteDataset	1º de novembro de 2023

Um novo nó no sumário mostra exemplos do Amazon Rekognition hospedados em GitHub	Os exemplos de código atualizados do repositór io de exemplos de AWS código agora aparecem em um nó separado no guia do desenvolvedor do Amazon Rekognition para facilitar o acesso.	22 de outubro de 2021
O Amazon Rekognition pode detectar quadros pretos e conteúdo primário do programa em segmentos de vídeo	O Amazon Rekognition pode identificar quadros pretos, barras coloridas, créditos de abertura, créditos finais, logotipos de estúdio e conteúdo primário do programa como dicas técnicas em um vídeo usando as operações `StartSegmentDetection` e `GetSegmentDetection` .	7 de junho de 2021
Atualização do gerenciamento do conjunto de dados para as seguintes políticas gerenciadas:	Você pode usar a operação `DetectText` do Amazon Rekognition para detectar até 100 palavras em uma imagem.	21 de maio de 2021
Atualização de marcação para e AmazonRekognitionReadOnlyAccessAmazonRekognitionFullAccess	O Rekognition adicionou novas ações de marcação às políticas `AmazonRekognitionFullAccess` e `AmazonRekognitionReadOnlyAccess` .	2 de abril de 2021

O Amazon Rekognition agora oferece suporte à marcação	Agora você pode usar tags para identificar, organizar, pesquisar e filtrar coleções, processadores de stream e modelos de Amazon Rekogniti on Custom Labels.	25 de março de 2021
Amazon Rekognition agora pode detectar equipamentos de proteção pessoal	Agora, o Amazon Rekogniti on pode detectar capas de mãos, de rosto e de cabeça em pessoas em uma imagem.	15 de outubro de 2020
Amazon Rekognition tem novas categorias de moderação de conteúdo	As categorias de moderação de conteúdo do Amazon Rekognition agora incluem 6 novas categorias: Drogas, tabaco, álcool, jogos de azar, gestos rudes e símbolos de ódio.	12 de outubro de 2020
Novo tutorial para exibir os resultados do Amazon Rekognition Video do Kinesis Video Streams localmente	Você pode exibir a saída do Amazon Rekognition Video de um streaming de vídeo no Kinesis Video Streams em um feed de vídeo local.	20 de julho de 2020
Novo tutorial do Amazon Rekognition para usar o Gstreamer	Usando o Gstreamer, você pode ingerir um vídeo de transmissão ao vivo de uma fonte de câmera do dispositi vo para o Amazon Rekogniti on Video por meio do Kinesis Video Streams.	17 de julho de 2020

O Amazon Rekognition agora oferece suporte à segmentação de vídeos armazenados	Com a API assíncrona de segmentação de vídeo Amazon Rekognition, você pode detectar quadros pretos, barras de cores, créditos finais e capturas em vídeos armazenados.	22 de junho de 2020
O Amazon Rekognition agora oferece suporte às políticas de endpoint do Amazon VPC	Ao especificar uma política, você pode restringir o acesso a um endpoint Amazon Rekognition Amazon VPC.	3 de março de 2020
O Amazon Rekognition agora suporta a detecção de texto em vídeos armazenados	Você pode usar a API Amazon Rekognition Video para detectar texto de forma assíncrona em um vídeo armazenado.	17 de fevereiro de 2020
O Amazon Rekognition agora oferece suporte à IA aumentada (versão prévia) e aos Amazon Rekognition Custom Labels	Com os rótulos personalizados do Amazon Rekognition, você pode detectar objetos, cenas e conceitos especializados em imagens criando seu próprio modelo de aprendizado de máquina. DetectModerationLabels agora é compatível com Amazon Augmented AI (versão prévia).	3 de dezembro de 2019
O Amazon Rekognition agora oferece suporte à AWS PrivateLink	Com a AWS, PrivateLink você pode estabelecer uma conexão privada entre sua VPC e o Amazon Rekognition.	12 de setembro de 2019

Filtragem facial do Amazon Rekognition	O Amazon Rekognition adiciona suporte aprimorado à filtragem facial à operação da API e introduz a filtragem facial IndexFaces para as operações da API. CompareFaces SearchFacesByImage	12 de setembro de 2019
Exemplos do Amazon Rekognition Video atualizados	Código de exemplo do Amazon Rekognition Video atualizado para criar e configurar o tópico do Amazon SNS e a fila do Amazon SQS.	5 de setembro de 2019
Exemplos de Ruby e Node.js adicionados	Exemplos de Amazon Rekognition Image Ruby e Node.js adicionados para detecção síncrona de rótulos e faces.	19 de agosto de 2019
Detecção de conteúdo não seguro atualizada	A detecção de conteúdo inseguro do Amazon Rekognition agora pode detectar conteúdo violento.	9 de agosto de 2019
GetContentModeration operação atualizada	GetContentModeration agora retorna a versão do modelo de detecção de moderação usado para detectar conteúdo não seguro.	13 de fevereiro de 2019

GetLabelDetection e DetectModerationLabels operações atualizadas	GetLabelDetection agora retorna informações da caixa delimitadora para objetos comuns e uma taxonomia hierárquica dos rótulos detectados. A versão do modelo usada para detecção de rótulos agora é retornada. DetectModerationLabels agora retorna a versão do modelo usado para detectar conteúdo não seguro.	17 de janeiro de 2019
DetectFaces e IndexFaces operação atualizada	Esta versão atualiza DetectFaces a IndexFace s operação e. Quando o parâmetro de entrada Attribute s é definido como ALL, os pontos de referência da localização do rosto incluem 5 novos pontos de referência: upperJawlineLeft, midJawlin eLeft, ChinBottom,, midJawlin eRight. upperJawlineRight	19 de novembro de 2018
DetectLabels operação atualizada	As caixas delimitadoras agora são retornadas para determina dos objetos. Uma taxonomia hierárquica já está disponíve l para os rótulos. Agora, você pode obter a versão do modelo de detecção usado para a detecção.	1 de novembro de 2018

IndexFaces operação atualizada	Agora IndexFaces você pode usar o parâmetro QualityFilter de entrada para filtrar faces detectadas com baixa qualidade. Você também pode usar o parâmetro MaxFaces de entrada para reduzir o número de faces retornadas com base na qualidade da detecção facial e no tamanho da face detectada.	18 de setembro de 2018
DescribeCollection operação adicionada	Agora você pode obter informações sobre uma coleção existente chamando a DescribeCollection operação.	22 de agosto de 2018
Novos exemplos de Python	Exemplos de Python foram adicionados ao conteúdo do Amazon Rekognition Video junto com algumas reorganizações de conteúdo.	26 de junho de 2018
Layout de conteúdo atualizado	O conteúdo do Amazon Rekognition Image foi reorganizado junto com novos exemplos de Python e C#.	29 de maio de 2018

O Amazon Rekognition oferece suporte AWS CloudTrail	O Amazon Rekognition é AWS CloudTrail integrado com, um serviço que fornece um registro das ações realizada s por um usuário, função ou serviço no Amazon Rekognition. AWS Para obter mais informações, consulte Registrar chamadas de API do Amazon Rekognition com a AWS. CloudTrail	6 de abril de 2018
Analisar vídeos armazenados e de streaming. Novo índice	Para obter informações sobre a análise de vídeos armazenados, consulte Trabalhar com vídeos armazenados. Para obter informações sobre a análise de vídeos de streaming, consulte Trabalhar com vídeos de streaming. O índice da documentação do Amazon Rekognition foi reorganizado para acomodar operações de imagem e vídeo.	29 de novembro de 2017

Texto em modelos de detecção de imagem e rosto	O Amazon Rekognition agora pode detectar texto em imagens. Para obter mais informações, consulte Detectar texto. O Amazon Rekognition apresenta o versionamento para o modelo de aprendizado profundo de detecção facial. Para obter mais informações, consulte Versionamento de modelo.	21 de novembro de 2017
Reconhecimento de celebridades	O Amazon Rekognition agora pode analisar imagens de celebridades. Para obter mais informações, consulte Reconhecer celebridades.	8 de junho de 2017
Moderação de imagem	Agora, o Amazon Rekogniti on pode determinar se uma imagem contém conteúdo adulto explícito ou sugestivo. Para obter mais informações, consulte Detectar conteúdo não seguro.	19 de abril de 2017

Faixa etária para faces detectadas. Painel de métricas de reconhecimento agregado	Agora, o Amazon Rekogniti on retorna a faixa etária estimada, em anos, para faces detectadas pela API Rekogniti on. Para obter mais informaçõ es, consulte AgeRange. O console do Rekognition agora tem um painel de métricas mostrando gráficos de atividades de um conjunto de métricas da Amazon CloudWatch para o Rekogniti on durante um período de tempo especificado. Para obter mais informaçõ es, consulte Exercício 4: Consultar métricas agregadas (console).	9 de fevereiro de 2017
Serviço e guia novos	Esta é a versão inicial do serviço de análise de imagens, Amazon Rekognition, e o Guia de desenvolvedor do Amazon Rekognition.	30 de novembro de 2016

AWS Glossário

Para obter a AWS terminologia mais recente, consulte o AWS glossário na Glossário da AWS Referência.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.