Guide du développeur

Amazon Rekognition

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Rekognition: Guide du développeur

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques commerciales et la présentation commerciale d’Amazon ne peuvent pas être utilisées en relation avec un produit ou un service extérieur à Amazon, d’une manière susceptible d’entraîner une confusion chez les clients, ou d’une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d’Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Qu’est-ce qu’Amazon Rekognition ? ... 1
Capacités clés ... 1
Cas d’utilisation ... 2
Avantages ... 4
Amazon Rekognition et éligibilité à la loi HIPAA ... 5
Utilisez-vous Amazon Rekognition pour la première fois ? ... 5
Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition ... 6
Comment fonctionne Rekognition ... 7
Comprendre les types d'analyse de Rekognition ... 8
Étiquettes ... 10
Étiquettes personnalisées ... 11
Détection de la vivacité faciale ... 12
Détection et analyse faciales ... 12
Recherche faciale ... 13
Chemins de personnes ... 13
Equipement de protection individuelle ... 13
Célébrités ... 13
Détection de texte ... 14
Contenus inappropriés ou offensants ... 14
Personnalisation ... 14
Analyse en bloc ... 15
Comprendre les opérations d'imagerie et de vidéo de Rekognition ... 15
Fonctionnement d’Image Amazon Rekognition ... 15
Opérations de Vidéo Amazon Rekognition ... 16
Non-storage et opérations basées sur le stockage ... 16
Utilisation de l'AWS SDK ou de HTTP pour l’appel des opérations d’API Amazon Rekognition ... 17
Non-storage et opérations d'API de stockage ... 17
Non-storage opérations ... 18
Storage-based Opérations d'API ... 20
Comprendre le versionnement des modèles ... 21
Premiers pas ... 24
Étape 1 : configurer un compte AWS et créer un utilisateur ... 24
Créez un AWS Compte et utilisateur ... 25

Étape 2 : Configuration des AWS SDK AWS CLI et ... 26
Octroi d'un accès par programmation ... 28
Utilisation des AWS kits de développement logiciel ... 33
Étape 3 : Commencer à utiliser le AWS CLI and AWS API SDK ... 34
Formatage du AWS CLI exemples ... 35
Étape suivante ... 35
Étape 4 : mise en route à l'aide de la console ... 36
Configurer des autorisations de console ... 37
Exercice 1 : détecter des objets et des scènes (console) ... 40
Exercice 2 : analyser les visages (console) ... 48
Exercice 3 : comparer des visages (console) ... 50
Exercice 4 :afficher les métriques agrégées (console) ... 53
Utilisation d'images et de vidéos ... 56
Travail avec les images ... 56
Spécifications d'images ... 57
Analyse d'images dans un compartiment Amazon S3 ... 59
Utilisation d'un système de fichiers local ... 75
Affichage de cadres de délimitation ... 91
Obtention de l'orientation d'une image et des coordonnées du cadre de délimitation ... 104
Utilisation des opérations d'analyse vidéo stockées ... 115
Types d'analyse ... 116
Présentation de l'API Vidéo Amazon Rekognition ... 116
Appeler les opérations de Vidéo Amazon Rekognition ... 119
Configuration de Vidéo Amazon Rekognition ... 126
Analyse d'une vidéo stockée (SDK) ... 130
Analyse d'une vidéo (AWS CLI) ... 159
Référence : Notification des résultats d'une analyse vidéo ... 163
Résolution des problèmes liés à Vidéo Amazon Rekognition ... 164
Utilisation des vidéos streaming ... 166
Présentation des opérations du processeur de flux Vidéo Amazon Rekognition ... 167
Ajouter des balises à un nouveau processeur de flux ... 168
Ajouter des balises à un processeur de flux existant ... 169
Répertorier les balises dans un processeur de flux ... 170
Supprimer des balises d'un processeur de flux ... 171
Gestion des erreurs ... 171
Composants erreur ... 172

Codes et messages d'erreur ... 172
Gestion des erreurs dans votre application ... 178
Utilisation d'Amazon Rekognition avec FedRAMP ... 179
Bonnes pratiques pour les capteurs, images d'entrée et vidéos ... 183
Latence de fonctionnement d'Image Amazon Rekognition ... 184
Recommandations pour les images d'entrée de comparaison faciale ... 184
Recommandations générales pour la saisie d'images pour les opérations faciales ... 184
Recommandations pour rechercher des visages dans une collection ... 185
Recommandations pour la configuration de la caméra (image et vidéo) ... 186
Recommandations pour la configuration de la caméra (vidéos stockées et en streaming) ... 188
Recommandations pour la configuration de la caméra (vidéo en streaming) ... 189
Recommandations pour l'utilisation de Face Liveness ... 190
Détection d'objets et de concepts ... 191
Étiqueter les objets de réponse ... 193
Cadres de délimitation ... 193
Score de fiabilité ... 194
Parents ... 194
Catégories ... 194
les alias ; ... 194
Propriétés de l'image ... 195
Version de modèle ... 197
Filtres d'inclusion et d'exclusion ... 197
Tri et agrégation des résultats ... 198
Détection des étiquettes dans une image ... 198
DetectLabels demande d'opération ... 209
DetectLabels réponse ... 211
Transformation de la DetectLabels réponse ... 214
Détection des étiquettes dans une vidéo ... 218
StartLabelDetection Demande ... 219
GetLabelDetection Réponse à l'opération ... 220
Transformer la GetLabelDetection réponse ... 227
Détection d'étiquettes dans des événements vidéo en streaming ... 235
Configuration de vos ressources Vidéo Amazon Rekognition et Amazon Kinesis ... 237
Opérations de détection des étiquettes d'appel pour les événements vidéo en streaming ... 241
Détection des étiquettes personnalisées ... 248
Détection et analyse des visages ... 249

Vue d'ensemble de la détection des visages et de la comparaison faciale 250
Directives pour les attributs faciaux 252
Détection de visages sur une image 253
DetectFaces demande d'opération 265
DetectFaces réponse à l'opération 265
Comparaison de visages dans les images 273
CompareFaces demande d'opération 285
CompareFaces réponse à l'opération 285
Détection de visages dans une vidéo stockée 288
GetFaceDetection réponse à l'opération 298
Recherche de visages dans une collection 303
Gestion des collections de visages, des visages et des utilisateurs 306
Gérer une collection 307
Gestion des visages dans une collection 307
Gérer les utilisateurs dans une collection 308
Utilisation de seuils de similarité pour associer et faire correspondre des visages 308
Utilisation de seuils de similarité pour associer des visages 308
Utilisation des seuils de similarité pour faire correspondre les visages 308
Conseils pour l'indexation des visages dans les scénarios courants 309
Applications de sécurité publique ou critique 309
Applications de partage de photos et de réseaux sociaux 310
Utilisation générale 310
Recherche de visages et d’utilisateurs dans une collection 310
Cas d’utilisation impliquant la sécurité publique 311
Utiliser Amazon Rekognition pour renforcer la sécurité publique 313
Création d’une collection 314
CreateCollection demande d'opération 320
CreateCollection réponse à l'opération 321
Baliser des collections 321
Ajouter des balises à une nouvelle collection 321
Ajouter des balises à une collection existante 322
Répertorier les balises d’une collection 324
Supprimer des balises d’une collection 325
Créer une liste de collections 326
ListCollections demande d'opération 332
ListCollections réponse à l'opération 333

Description d'une collection 333
DescribeCollection demande d'opération 340
DescribeCollection réponse à l'opération 340
Suppression d'une collection 341
DeleteCollection demande d'opération 348
DeleteCollection réponse à l'opération 348
Ajout de visages à une collection 348
Filtrage des visages 349
IndexFaces demande d'opération 359
IndexFaces réponse à l'opération 359
Répertorier les visages et les utilisateurs associés dans une collection 368
ListFaces demande d'opération 374
ListFaces réponse à l'opération 374
Suppression de visages d'une collection 376
DeleteFaces demande d'opération 382
DeleteFaces réponse à l'opération 382
Création d'un utilisateur 383
Suppression d'un utilisateur 386
Associer des visages à un utilisateur 389
AssociateFaces réponse à l'opération 392
Dissocier des visages d'un utilisateur 393
DisassociateFaces réponse à l'opération 396
Création d'une liste de visages d'une collection 397
ListUsers réponse à l'opération 400
Recherche d'un visage (ID de visage) 401
SearchFaces demande d'opération 407
SearchFaces réponse à l'opération 408
Recherche d'un visage (image) 409
SearchFacesByImage demande d'opération 416
SearchFacesByImage réponse à l'opération 417
Recherche d'utilisateurs (identifiant facial/ID utilisateur) 418
SearchUsers demande d'opération 422
SearchUsers réponse à l'opération 423
Recherche d'utilisateurs (image) 424
SearchUsersByImage demande d'opération 428
SearchUsersByImage réponse à l'opération 428

Recherche de visages dans des vidéos stockées ... 430
GetFaceSearch réponse à l'opération ... 439
Recherche de visages dans une collection en vidéo streaming ... 444
Configuration de vos ressources Vidéo Amazon Rekognition et Amazon Kinesis ... 445
Recherche de visages dans une vidéo en streaming ... 449
Streaming à l'aide d'un plugin GStreamer ... 474
Résolution des problèmes de streaming vidéo ... 477
Tracé du parcours de personnes ... 484
GetPersonTracking réponse à l'opération ... 494
Détection des équipements de protection individuelle ... 498
Types d'EPI ... 499
Couvre-visage ... 499
Couvre-main ... 499
Couvre-tête ... 499
Fiabilité de détection des EPI ... 500
Présentation des EPI détectés dans une image ... 500
Tutoriel : Création d'un AWS Lambda fonction qui détecte les images avec le PPE ... 501
Comprendre l'API de détection des EPI ... 501
Fourniture d'une image ... 501
Comprendre la DetectProtectiveEquipment réponse ... 503
Détection de l'EPI dans une image ... 508
Exemple : cadres de délimitation et couvre-visages ... 520
Reconnaissance de célébrités ... 536
Comparaison entre reconnaissance de célébrités et recherche faciale ... 537
Reconnaissance de célébrités sur une image ... 537
Appel RecognizeCelebrities ... 538
RecognizeCelebrities demande d'opération ... 548
RecognizeCelebrities réponse à l'opération ... 548
Reconnaissance de célébrités dans une vidéo stockée ... 551
GetCelebrityRecognition réponse à l'opération ... 567
Obtention d'informations sur une célébrité ... 569
Appel GetCelebrityInfo ... 569
GetCelebrityInfo demande d'opération ... 575
GetCelebrityInfo réponse à l'opération ... 575
Modération du contenu ... 576
Utilisation des API de modération d'image et de vidéo ... 578

Catégories d'étiquette ... 579
Type de contenu ... 592
Fiabilité ... 593
Gestion des versions ... 593
Tri et agrégation ... 593
Statuts des adaptateurs de modération personnalisés ... 594
Test de la version 7 de Content Moderation et transformation de la réponse de l'API ... 594
AWS SDK et guide d'utilisation pour la modération de contenu version 7 ... 595
Mappages d'étiquettes pour les versions 6.1 à 7 ... 596
Détection d'images inappropriées ... 601
Détection d'un contenu inapproprié dans une image ... 601
DetectModerationLabels demande d'opération ... 608
DetectModerationLabels réponse à l'opération ... 609
Détection des vidéos stockées inappropriées ... 610
GetContentModeration réponse à l'opération ... 619
Amélioration de la précision grâce à la modération personnalisée ... 622
Création et utilisation d'adaptateurs ... 622
Préparation de vos jeux de données ... 626
Gestion des adaptateurs à l'aide du AWS CLI et SDK ... 628
Tutoriel sur l'adaptateur de modération personnalisé ... 635
Évaluation et amélioration de votre adaptateur ... 654
Formats de fichiers manifestes ... 656
Bonnes pratiques relatives aux adaptateurs d'entraînement ... 662
Configuration des AutoUpdate autorisations ... 662
AWS Notification du Health Dashboard pour Rekognition ... 665
Vérification de contenu inappropriée avec Amazon A2I ... 667
Détection de texte ... 673
Détection de texte sur une image ... 675
DetectText demande d'opération ... 684
DetectText réponse à l'opération ... 685
Détection de texte dans une vidéo stockée ... 690
Filtres ... 700
GetTextDetection réponse ... 701
Détection de segments vidéo ... 707
Repères techniques ... 708
Images noires ... 708

Génériques ... 708
Barres de couleur ... 709
Ardoises ... 709
Logos de studio ... 709
Contenu ... 709
Détection des plans ... 710
À propos de l'API de détection des segments Vidéo Amazon Rekognition ... 711
Utilisation de l'API de segmentation Amazon Rekognition ... 711
Démarrage de l'analyse de segments ... 712
Obtention des résultats de l'analyse des segments ... 713
Exemple : Détection de segments dans une vidéo stockée ... 718
Détection de la vivacité faciale ... 731
User-Side Exigences relatives à la vivacité du visage ... 733
Architecture et diagrammes de séquence ... 734
Conditions préalables ... 737
Étape 1 : Configurez un AWS compte ... 737
Étape 2 : configurer le Face Liveness AWS Kits SDK ... 737
Étape 3 : Configurer AWS Amplifier les ressources ... 738
Bonnes pratiques pour détecter la vivacité faciale ... 738
Programmation des API Face Liveness d'Amazon Rekognition ... 738
Étape 1 : CreateFaceLivenessSession ... 739
Étape 2 : StartFaceLivenessSession ... 740
Étape 3 : GetFaceLivenessSessionResults ... 741
Étape 4 : répondre aux résultats ... 742
Appeler les API Face Liveness ... 742
Configuration et personnalisation de votre application ... 748
Configuration de votre application ... 748
Personnaliser votre application ... 749
Modèle de responsabilité partagée Face Liveness ... 749
Mise à jour des directives de Face Liveness ... 753
Gestion des versions et délais ... 753
Version, publication et matrice de compatibilité ... 754
Communication des nouveautés ... 755
Face Liveness FAQ ... 755
Analyse en bloc ... 760
Traitement d'images en bloc ... 761

Pour créer une tâche d'analyse en bloc (CLI) ... 761
Manifestes de sortie d'analyse en bloc ... 762
Type de contenu ... 764
Vérification des prédictions et des adaptateurs d'entraînement ... 764
Tutoriels ... 766
Stockage des données Amazon Rekognition avec Amazon RDS et DynamoDB ... 766
Conditions préalables ... 767
Obtenir des étiquettes pour les images dans un compartiment Amazon S3 ... 767
Création d'un tableau Amazon DynamoDB ... 769
Chargement de données vers DynamoDB ... 770
Création d'une base de données MySQL dans Amazon RDS ... 772
Téléchargement de données vers un tableau MySQL Amazon RDS ... 773
Utilisation d'Amazon Rekognition et Lambda pour étiqueter les actifs dans un compartiment Amazon S3 ... 776
Conditions préalables ... 778
Configuration du rôle IAM Lambda ... 778
Création du projet ... 779
Ecriture du code ... 781
Package du projet ... 792
Déployez la fonction Lambda. ... 793
Test de la méthode Lambda ... 793
Création AWS applications d'analyse vidéo ... 795
Conditions préalables ... 795
Procédure ... 796
Création d'une fonction Lambda Amazon Rekognition ... 796
Conditions préalables ... 799
Création de la rubrique SNS ... 799
Créer la fonction Lambda ... 799
Configurer la fonction Lambda ... 800
Configuration du rôle IAM Lambda ... 801
Créez le AWS Toolkit for Eclipse Projet Lambda ... 802
Test de la fonction Lambda ... 806
Utilisation d'Amazon Rekognition pour la vérification d'identité ... 807
Conditions préalables ... 808
Création d'une collection ... 808
Enregistrement d'un nouvel utilisateur ... 809

Connexion utilisateur existante ... 818
Utilisation d'Amazon Rekognition (REK) pour détecter les étiquettes destinées aux applications marketing ... 821
Conditions préalables ... 822
Création et remplissage d'une table DynamoDB ... 822
Création d'une fonction Lambda avec journalisation et notification ... 825
Configuration des autorisations et de la fonction Lambda ... 827
Chargement d'images vers Amazon S3 avec des métadonnées ... 829
CloudWatch Journaux de sondage fournis par le client ... 831
Détection d'étiquettes dans une image à l'aide de Lambda et Python ... 832
Créer une fonction Lambda (console) ... 832
(Facultatif) Création d'une couche (console) ... 834
Ajouter le code Python (console) ... 836
Pour ajouter le code Python (console) ... 838
Exemples de code ... 842
Principes de base ... 843
Bonjour Amazon Rekognition ... 844
Actions ... 847
Scénarios ... 1020
Créez une collection et trouvez-y des visages ... 1021
Création d'une application sans serveur pour gérer des photos ... 1033
Détecter l'EPI dans des images ... 1038
Détecter et afficher des éléments dans des images ... 1038
Détecter des visages dans une image ... 1054
Détecter les informations contenues dans les vidéos ... 1055
Détecter des objets dans des images ... 1095
Détecter des personnes et des objets dans une vidéo ... 1098
Enregistrer des informations EXIF et d'autres informations sur les images ... 1100
Référence d'API ... 1101
Sécurité ... 1102
Gestion des identités et des accès ... 1102
Public ciblé ... 1103
Authentification par des identités ... 1103
Gestion de l'accès à l'aide de politiques ... 1104
Fonctionnement d'Amazon Rekognition avec IAM ... 1106
Politiques gérées par AWS ... 1113

Bonnes pratiques en matière de politiques ... 1121
Utilisation de la console ... 1122
Exemples d'utilisation d'exemples de politiques basées sur l'identité ... 1123
Exemples de Resource-based politiques ... 1126
Résolution des problèmes ... 1127
Protection des données ... 1129
Chiffrement des données ... 1130
Confidentialité du trafic inter-réseau ... 1133
Utilisation d'Amazon Rekognition avec les points de terminaison d'un VPC Amazon ... 1133
Création de points de terminaison d'un VPC pour Amazon Rekognition ... 1134
Créez une politique de point de terminaison d'un VPC pour Amazon Rekognition ... 1135
Validation de conformité ... 1136
Résilience ... 1137
Analyse de la configuration et des vulnérabilités ... 1138
Cross-service prévention confuse des adjoints ... 1138
Sécurité de l'infrastructure ... 1139
Contrôle ... 1141
Surveillance de la Rekognition avec Amazon CloudWatch ... 1142
Utilisation de CloudWatch métriques pour la Rekognition ... 1142
Métriques Access Rekognition ... 1143
Créer une alarme ... 1144
CloudWatch métriques pour Rekognition ... 1146
Journalisation des appels d'API Amazon Rekognition avec AWS CloudTrail ... 1151
Informations sur Amazon Rekognition dans CloudTrail ... 1152
Présentation des entrées du fichier journal d'Amazon Rekognition ... 1153
Consignes et quotas ... 1156
Régions prises en charge ... 1156
Définition des quotas. ... 1156
Image Amazon Rekognition ... 1156
Analyse globale des images Amazon Rekognition ... 1156
Vidéo stockée par Vidéo Amazon Rekognition ... 1158
Vidéo en streaming Vidéo Amazon Rekognition ... 1158
Quotas par défaut ... 1159
Calculer la modification de quota TPS ... 1159
Bonnes pratiques relatives aux quotas TPS ... 1159
Créez un dossier pour modifier les quotas TPS ... 1160

Historique de la documentation 1163
AWS Glossaire 1179
..... mclxxx

Qu’est-ce qu’Amazon Rekognition ?

Amazon Rekognition est un service d'analyse d'images et de vidéos basé sur le cloud qui permet d'ajouter facilement des fonctionnalités avancées de vision par ordinateur à vos applications. Le service repose sur une technologie d'apprentissage en profondeur éprouvée et son utilisation ne nécessite aucune expertise en apprentissage automatique. Amazon Rekognition comprend une API simple et facile d’utilisation pouvant rapidement analyser tout fichier vidéo ou image stocké dans Amazon S3.

Vous pouvez ajouter des fonctionnalités qui détectent les objets, le texte, le contenu non sécurisé images/videos, analysent et comparent les visages à votre application à l'aide des API de Rekognition. Avec les API d’Amazon Rekognition vous pouvez détecter, analyser et comparer des visages pour une grande variété de cas d’utilisation, y compris la vérification d’utilisateurs, le catalogage, le comptage de personnes et la sécurité publique.

Le service est basé sur la même technologie d'apprentissage en profondeur éprouvée et hautement évolutive développée par les spécialistes de la vision par ordinateur d'Amazon, une technologie capable d'analyser des milliards d'images et de vidéos par jour. Rekognition apprend régulièrement à partir de nouvelles données, et nous ajoutons fréquemment de nouvelles étiquettes et fonctionnalités au service.

Pour plus d’informations, consultez FAQ Amazon Rekognition.

Capacités clés

Analyse d'image :

- Détection d'objets, de scènes et de concepts : détecte et classe les objets, les scènes, les concepts et les célébrités dans les images.
- Détection de texte - Détectez et reconnaissez le texte imprimé et manuscrit dans des images dans différentes langues.
- Contenu dangereux : détectez et filtrez le contenu et les images explicites, inappropriés et violents. Détecte les étiquettes granulaires contenant des contenus dangereux.
- Reconnaissance des célébrités - Reconnaissez des dizaines de milliers de célébrités sur vos images dans différentes catégories, telles que des politiciens, des athlètes, des acteurs et des musiciens.

- Analyse faciale - Détectez, analysez et comparez les visages, ainsi que les attributs faciaux, tels que le sexe, l'âge et les émotions. Les cas d'utilisation peuvent inclure la vérification des utilisateurs, le catalogage, le comptage des personnes et la sécurité publique.
- Étiquettes personnalisées - Créez des classificateurs personnalisés pour détecter les objets spécifiques à votre cas d'utilisation, tels que les logos, les produits, les personnages.
- Propriétés de l'image - Analysez les propriétés de l'image telles que la qualité, la couleur, la netteté et le contraste.

Analyse vidéo :

- Détection d'objets, de scènes et de concepts : détecte et classe les objets, les scènes, les concepts et les célébrités dans les vidéos.
- Détection de texte - Détectez et reconnaissez le texte imprimé et manuscrit dans des vidéos dans différentes langues.
- Trajectoire des personnes : suivez les personnes identifiées lorsqu'elles passent d'une image à l'autre.
- Analyse faciale : détectez, analysez et comparez les visages dans des vidéos en streaming ou stockées.
- Reconnaissance des célébrités - Reconnaissez des dizaines de milliers de célébrités dans vos vidéos stockées dans différentes catégories, telles que des politiciens, des athlètes, des acteurs et des musiciens.
- Détection des contenus non sécurisés : détectez les contenus explicites, inappropriés et violents dans les vidéos.
- Segmentation vidéo - Identifiez automatiquement les segments vidéo utiles, tels que les cadres noirs et le générique de fin.
- Visage vivant : détectez si un utilisateur en direct est présent lors de la vérification faciale.

Cas d’utilisation

Bibliothèques multimédias consultables - Rekognition détecte les étiquettes, les objets, les concepts et les scènes dans les images et les vidéos. Vous pouvez rendre ces étiquettes consultables sur la base de cette analyse de contenu visuel. Utile pour créer des bibliothèques d'images et de vidéos consultables.

Face-Based Vérification de l'identité de l'utilisateur : confirmez l'identité des utilisateurs en comparant les visages des images aux images des visages de référence. Utile pour la vérification d'identité dans les applications.

Détection de la vivacité faciale - Rekognition Face Liveness est une fonctionnalité d'apprentissage automatique (ML) entièrement gérée conçue pour aider les développeurs à prévenir les fraudes lors de la vérification d'identité basée sur le visage. Cette fonctionnalité vous permet de vérifier qu'un utilisateur est physiquement présent devant la caméra, et qu'il ne s'agit pas d'un mauvais acteur qui usurpe son visage. L'utilisation de Rekognition Face Liveness peut vous aider à détecter les attaques frauduleuses présentées à un appareil photo, telles que des photos imprimées, des masques numériques ou 3D. photos/videos Il permet également de détecter les attaques frauduleuses qui contournent une caméra, telles que les vidéos préenregistrées ou les vidéos deepfake injectées directement dans le sous-système de capture vidéo.

Recherche faciale - Avec Rekognition, vous pouvez rechercher dans des images, des vidéos stockées et des vidéos en streaming des visages correspondant à ceux stockés dans un conteneur appelé collection de visages. Une collection de visages est un index de visages que vous possédez et gérez. Pour rechercher des personnes en fonction de leur visage, vous devez indexer les visages, puis rechercher les visages.

Détection des contenus dangereux : détectez et filtrez les contenus explicites, inappropriés et violents dans les images et les vidéos. Utilise des étiquettes pour un filtrage granulaire en fonction des besoins de l'entreprise. L'API de modération du contenu renvoie également une liste hiérarchique de toutes les étiquettes détectées (objets et concepts), ainsi que des scores de confiance. Ils objects/labels indiquent des catégories spécifiques de contenus dangereux, ce qui permet un filtrage granulaire et une gestion de gros volumes de contenu généré par les utilisateurs (UGC). Vous pouvez personnaliser le résultat de l'API de modération de contenu à l'aide d'adaptateurs, qui améliorent les performances des images telles que celles que vous fournissez sous forme de données d'entraînement.

Détection des équipements de protection individuelle - Détectez les équipements de protection individuelle sur des images afin de contrôler le respect des normes de sécurité dans divers secteurs. Vous pouvez signaler automatiquement les conditions dangereuses en détectant les équipements inappropriés et en recevant des alertes concernant ces conditions, ce qui peut améliorer la conformité et la formation.

Reconnaissance des célébrités - Reconnaissez les célébrités présentes dans vos images et vidéos dans toutes les catégories, telles que les politiciens, les athlètes, les acteurs et les musiciens. Vous pouvez identifier les apparitions de célébrités sans avoir à fournir de noms.

Détection de texte : détectez et extrayez du texte dans des images pour une recherche visuelle ou pour extraire des métadonnées. Cela fonctionne sur différentes polices et styles. Détecte l'orientation pour gérer le texte sur les panneaux et les bannières.

Étiquettes personnalisées : identifiez des objets, des concepts et des scènes personnalisés spécifiques aux cas d'utilisation professionnels, tels que la détection de logos. Vous pouvez entraîner des classificateurs personnalisés à gérer des objets spécifiques ou propriétaires, ce qui améliore la précision des objets clés par rapport aux classificateurs généraux. Pour de plus amples informations, veuillez consulter Qu'est-ce qu'Étiquettes personnalisées Amazon Rekognition ? dans le manuel Guide du développeur d'Étiquettes personnalisées Amazon Rekognition.

Avantages

Intégrer une puissante analyse d'images et de vidéos à votre application - Ajoutez une analyse précise des images et des vidéos aux applications sans expertise. L'API Amazon Rekognition permet d'effectuer des analyses via le deep learning sans nécessiter de connaissances en machine learning. Vous pouvez rapidement intégrer la vision par ordinateur dans les applications Web, mobiles et pour appareils.

Analyse d'images et de vidéos basée sur le deep learning : analyse les images et les vidéos à l'aide du deep learning pour une grande précision. Amazon Rekognition' peut détecter des étiquettes, des objets, des scènes, des visages, des célébrités. Filtrez les résultats selon include/exclude des étiquettes spécifiques.

Analyse d'image évolutive : analyse des millions d'images pour organiser d'énormes ensembles de données visuelles. S'adapte à la croissance des bibliothèques d'images et du trafic. Vous n'avez pas besoin de planifier la capacité et vous ne payez que pour ce que vous utilisez.

Analyser et filtrer les images en fonction des propriétés : analysez et filtrez les images par propriétés, telles que la qualité, la couleur et le contenu visuel, et détectez la netteté, la luminosité et le contraste des images.

Intégration avec d'autres AWS services : Amazon Rekognition s'intègre immédiatement à S3 et Lambda. Vous pouvez appeler les API d'Amazon Rekognition depuis Lambda et traiter des images dans Amazon S3 sans déplacer de données. Rekognition intègre l'évolutivité et la sécurité grâce à l'IAM. AWS

Faible coût : Pay-as-you-go prix, pas de minimum ni d'engagement. Un niveau gratuit est disponible pour commencer. Économisez davantage au fur et à mesure que l'utilisation augmente grâce à une tarification échelonnée. Rentable par rapport aux solutions internes.

Personnalisation simple - Personnalisez la précision en fonction de votre cas d'utilisation avec des adaptateurs. Fournissez des exemples d'images pour entraîner les adaptateurs. Améliore la détection des objets et des étiquettes pour des domaines donnés. Un moyen facile de personnaliser l'analyse sans expertise en machine learning.

Pour plus d'informations, consultez FAQ Amazon Rekognition.

Amazon Rekognition et éligibilité à la loi HIPAA

Il s'agit d'un service admissible en vertu de la loi HIPAA. Pour plus d'informations sur AWS, U.S. Health Insurance Portability and Accountability Act de 1996 (HIPAA), et utilisation de AWS services pour traiter, stocker et transmettre des informations de santé protégées (PHI), voir la vue d'ensemble de la HIPAA.

Utilisez-vous Amazon Rekognition pour la première fois ?

Si vous utilisez Amazon Rekognition pour la première fois, nous vous recommandons de lire les sections suivantes pour :

1. Fonctionnement d'Amazon Rekognition : cette section présente les nombreux composants d'Amazon Rekognition que vous utilisez pour créer une expérience de bout en bout.
2. Premiers pas avec Amazon Rekognition : dans cette section, vous configurez votre compte, installez le SDK correspondant à la langue de votre choix et testez l'API Amazon Rekognition. Pour obtenir la liste des langages de programmation pris en charge par Amazon Rekognition, consultez Utiliser Rekognition avec un AWS Kit SDK.
3. Travail avec les images : cette section fournit des informations sur l'utilisation d'Amazon Rekognition avec des images stockées dans des compartiments Amazon S3 et des images chargées à partir d'un système de fichiers locaux.
4. Utilisation des opérations d'analyse vidéo stockées : cette section fournit des informations sur l'utilisation d'Amazon Rekognition avec des vidéos stockées dans un compartiment Amazon S3.
5. Utilisation des vidéos streaming : cette section fournit des informations sur l'utilisation d'Amazon Rekognition avec des vidéos en streaming.

Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition

> **Note**
>
> Les fonctionnalités de streaming vidéo et d'analyse d'images en masse ne seront plus disponibles pour les nouveaux clients à compter du 30 avril 2026.

Après mûre réflexion, nous avons décidé que le streaming vidéo et l'analyse d'images en masse ne seraient plus disponibles pour les nouveaux clients à compter du 30 avril 2026. Si vous souhaitez utiliser ces fonctionnalités avec de nouveaux comptes, veuillez le faire avant cette date. Aucune action n'est requise pour les comptes qui ont utilisé ces fonctionnalités au cours des 12 derniers mois ; ils y auront toujours accès.

Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Ressources pour faciliter la migration vers des solutions alternatives

- Pour plus d'informations sur l'utilisation de l'API Rekognition Image pour traiter les images provenant d'une entrée vidéo en streaming, consultez cette section Créer une solution sans serveur pour l'analyse et les alertes d'images vidéo sur le blog. AWS
- Pour plus d'informations sur l'utilisation de l'API Rekognition Image pour traiter un lot d'images stockées dans S3, consultez la section Création d'un service de floutage facial sans serveur pour les photos dans Amazon S3 sur le blog. AWS

Si vous avez d'autres questions, veuillez contacter le AWS Support.

Fonctionnement d’Amazon Rekognition

Amazon Rekognition fournit deux ensembles d'API pour l'analyse visuelle :

- Amazon Rekognition Image pour l'analyse d'images
- Amazon Rekognition Video pour l'analyse vidéo

Analyse d'image

Avec Amazon Rekognition Image, vos applications peuvent :

- Détectez des objets, des scènes et des concepts dans les images
- Reconnaître les célébrités
- Détecter du texte dans différentes langues
- Détectez le contenu ou les images explicites, inappropriés ou violents
- Détectez, analysez et comparez les visages et leurs attributs tels que l'âge et les émotions
- Détecter la présence d'EPI

Les cas d'utilisation incluent l'amélioration des applications photo, le catalogage d'images et la modération du contenu.

Analyse vidéo

Avec Amazon Rekognition Video, vos applications peuvent :

- Suivez des personnes et des objets sur des images vidéo
- Reconnaître les objets
- Reconnaître les célébrités
- Rechercher des vidéos stockées et diffusées en streaming pour les personnes d'intérêt
- Analysez les visages pour détecter des attributs tels que l'âge et les émotions
- Détectez le contenu ou les images explicites, inappropriés ou violents
- Agréger et trier les résultats d'analyse par horodatage et par segment
- Détectez les personnes, les animaux domestiques et les colis dans les vidéos en streaming

Les cas d'utilisation incluent l'analyse vidéo, le catalogage de vidéos et le filtrage de contenus inappropriés.

Fonctions principales

- Puissante analyse du deep learning
- Détection de haute précision pour les objets, les scènes, les visages, le texte
- API facile à utiliser pour l'intégration dans les applications
- Modèles personnalisables adaptés à vos données
- Analyse évolutive des médiathèques

Amazon Rekognition vous permet d'améliorer la précision de certains modèles d'apprentissage profond en formant un adaptateur personnalisé. Par exemple, avec Amazon Rekognition Custom Moderation, vous pouvez adapter le modèle d'analyse d'image de base d'Amazon Rekognition en formant un adaptateur personnalisé avec vos images. Voir Améliorer la précision grâce à la modération personnalisée pour plus d'informations.

Les sections suivantes décrivent les types d'analyses proposés par Amazon Rekognition et présentent un aperçu des opérations Amazon Rekognition Image et Amazon Rekognition Video. La différence entre les opérations hors stockage et de stockage est également expliquée.

Pour faire une démonstration des API Amazon Rekognition, vous pouvez consulter l'étape 3 : Commencer à utiliser l'API AWS CLI et le SDK AWS, qui explique comment essayer Rekognition dans la console. AWS

Rubriques

- Comprendre les types d'analyse de Rekognition
- Comprendre les opérations d'imagerie et de vidéo de Rekognition
- Comprendre les opérations non liées au stockage et aux API de stockage
- Comprendre le versionnement des modèles

Comprendre les types d'analyse de Rekognition

Les types d'analyse que l'API Image Amazon Rekognition et l'API Vidéo Amazon Rekognition peuvent effectuer sont les suivants. Pour en savoir plus sur les API, consultez Comprendre les opérations d'imagerie et de vidéo de Rekognition.

Le tableau suivant répertorie les opérations que vous devez utiliser en fonction du type de média avec lequel vous travaillez et de votre cas d'utilisation :

Cas d'utilisation	Type de support	Opérations
Modération du contenu	Images	DetectModerationLabels, StartMediaAnalysisJob, GetMediaAnalysisJob, ListMediaAnalysisJobs
	Vidéo stockée	StartContentModeration, GetContentModeration
Vérification d'identité	Images	CreateCollection, CreateUser, IndexFaces, AssociateFaces, SearchFacesByImage, SearchUsersByImage
	Vidéo stockée	CreateCollection, IndexFaces, StartFaceSearch, GetFaceSearch
	Vidéo en streaming (Détection de la vivacité faciale)	CreateFaceLivenessSession, StartFaceLivenessSession, GetFaceLivenessSessionResults,
Analyse faciale	Images	DetectFaces, CompareFaces
	Vidéo stockée	StartFaceDetection, GetFaceDetection
	Vidéo en streaming	CreateStreamProcessor, StartStreamProcessor
Reconnaissance d'objets et d'activités	Images	DetectLabels

Cas d'utilisation	Type de support	Opérations
	Vidéos stockées	StartLabelDetection, GetLabelDetection
Maison connectée	Vidéo en streaming	StartStreamProcessor
Analyse des médias	Vidéo stockée	StartSegmentDetection, GetSegmentDetection
Sécurité au travail	Images	DetectProtectiveEquipment
Détection de texte	Images	DetectText
	Vidéo	StartTextDetection, GetTextDetection
Tracé du chemin de personnes	Vidéo	StartPersonTracking, GetPersonTracking
Reconnaissance de célébrités	Images	RecognizeCelebrities
	Vidéo	StartCelebrityRecognition, GetCelebrityRecognition
Détection d'étiquettes personnalisées	Images	DetectCustomLabels
	Entraînement d'un modèle	Voir le guide du développeur de Custom Labels

Étiquettes

Sont considérés comme étiquettes les éléments suivants : les objets (ex. : une fleur, un arbre, une table), les événements (ex : un mariage, une cérémonie de remise de diplômes, un anniversaire), les concepts (ex. : un paysage, une soirée, la nature) ou les activités (ex. : courir ou jouer au basket). Amazon Rekognition peut détecter les étiquettes dans les images et les vidéos. Pour de plus amples informations, veuillez consulter Détection d'objets et de concepts.

Rekognition peut détecter une longue liste d'étiquettes dans les images et les vidéos enregistrées. Rekognition peut également détecter un petit nombre d'étiquettes dans les vidéos en streaming.

Utilisez les opérations suivantes pour détecter les étiquettes en fonction des cas d'utilisation :

- Pour détecter les étiquettes dans les images : utilisez DetectLabels. Vous pouvez identifier les propriétés de l'image, telles que les couleurs dominantes et la qualité de l'image. Pour ce faire, utilisez DetectLabelswith `IMAGE_PROPERTIES` comme paramètre d'entrée.
- Pour détecter les libellés dans les vidéos enregistrées : utilisez StartLabelDetection. La détection des couleurs et de la qualité d'image dominantes n'est pas prise en charge pour les vidéos stockées.
- Pour détecter les libellés dans une vidéo en streaming : utilisez CreateStreamProcessor. La détection des couleurs et de la qualité d'image dominantes n'est pas prise en charge pour le streaming vidéo.

Vous pouvez spécifier les types d'étiquettes que vous souhaitez renvoyer pour la détection d'étiquettes d'images et de vidéos stockées en utilisant des options de filtrage inclusives et exclusives.

Étiquettes personnalisées

Étiquettes personnalisées Amazon Rekognition peuvent identifier les objets et les scènes dans des images spécifiques aux besoins de votre entreprise en formant un modèle de machine learning. Par exemple, vous pouvez former un modèle de sorte qu'il détecte des logos ou des pièces de machines d'ingénierie sur une ligne d'assemblage.

> **Note**
>
> Pour des informations sur les Étiquettes personnalisées Amazon Rekognition, veuillez consulter le manuel du développeur Étiquettes personnalisées Amazon Rekognition.

Amazon Rekognition fournit une console que vous utilisez pour créer, former, évaluer et exécuter un modèle de machine learning. Pour de plus amples informations, veuillez consulter Mise en route sur Étiquettes personnalisées Amazon Rekognition dans le manuel Guide du développeur Étiquettes personnalisées Amazon Rekognition. Vous pouvez également utiliser l'API Étiquettes personnalisées Amazon Rekognition pour former et exécuter un modèle. Pour plus d'informations, consultez Getting

Started with the Amazon Rekognition Custom Labels SDK dans le manuel du développeur Amazon Rekognition. CustomLabels

Pour analyser des images à l'aide d'un modèle entraîné, utilisez DetectCustomLabels.

Détection de la vivacité faciale

Amazon Rekognition Face Liveness peut vous aider à vérifier qu'un utilisateur soumis à une vérification d'identité basée sur le visage est physiquement présent devant la caméra et qu'il ne s'agit pas d'un mauvais acteur qui usurpe le visage de l'utilisateur. Il détecte également les attaques frauduleuses présentées à une caméra ou qui tentent de contourner une caméra. Un utilisateur peut effectuer un test de vivacité faciale en prenant un court selfie vidéo, et un score de vivacité est renvoyé pour le contrôle. La vivacité faciale d'un visage est déterminée à l'aide d'un calcul probabiliste, et un score de confiance (compris entre 0 et 100) est renvoyé après le contrôle. Plus le score est élevé, plus on est sûr que la personne qui reçoit le chèque est en direct.

Pour plus d'informations sur la vivacité faciale, consultezDétection de la vivacité faciale.

Détection et analyse faciales

Amazon Rekognition peut détecter les visages dans les images et les vidéos. Avec Amazon Rekognition, vous pouvez obtenir des informations sur :

- Où des visages sont détectés dans une image ou une vidéo
- Des repères faciaux tels que la position des yeux
- La présence d'une occlusion faciale dans les images
- Émotions détectées, telles que le bonheur ou la tristesse
- Orientation du regard d'une personne dans les images

Vous pouvez également interpréter des informations démographiques telles que le sexe ou l'âge. Vous pouvez comparer un visage sur une image à des visages détectés sur une autre image. Les informations sur les visages peuvent également être stockées pour une récupération ultérieure. Pour de plus amples informations, veuillez consulter Détection et analyse des visages.

Pour détecter des visages dans des images, utilisez DetectFaces. Pour détecter des visages dans des vidéos stockées, utilisez StartFaceDetection.

Recherche faciale

Amazon Rekognition peut rechercher des visages. Les informations sur les visages sont indexées dans un conteneur appelé « collection ». Les informations sur les visages de la collection peuvent ensuite être comparées aux visages détectés dans les images, les vidéos stockées et les vidéos en streaming. Pour plus d'informations, veuillez consulter Recherche de visages dans une collection.

Pour rechercher des visages connus dans des images, utilisez DetectFaces. Pour rechercher des visages connus dans des vidéos stockées, utilisez StartFaceDetection. Pour rechercher des visages connus dans des vidéos en streaming, utilisez CreateStreamProcessor.

Chemins de personnes

Amazon Rekognition peut retracer les chemins des personnes détectées dans une vidéo stockée. Vidéo Amazon Rekognition fournit un suivi du chemin, des informations sur les visages et la localisation de trames pour les personnes détectées dans une vidéo. Pour de plus amples informations, veuillez consulter Tracé du parcours de personnes.

Pour détecter des personnes dans des vidéos stockées, utilisez StartPersonTracking.

Equipement de protection individuelle

Amazon Rekognition peut détecter les équipements de protection individuelle (EPI) portés par les personnes détectées sur une image. Amazon Rekognition détecte les couvre-mains, les couvre-visages et les couvre-chefs. Amazon Rekognition prédit si un article d'EPI couvre la partie du corps appropriée. Vous pouvez également vous procurer des cadres de délimitations pour les personnes détectées et des éléments de protection individuelle. Pour de plus amples informations, veuillez consulter Détection des équipements de protection individuelle.

Pour détecter les équipements de protection individuelle sur les images, utilisez DetectProtectiveEquipment.

Célébrités

Amazon Rekognition peut reconnaître des milliers de célébrités dans des images et des vidéos stockées. Vous pouvez obtenir des informations sur la place du visage d'une célébrité sur une image, sur des repères faciaux et sur l'expression du visage d'une célébrité. Vous pouvez obtenir des informations de suivi sur des célébrités, à mesure qu'elles apparaissent dans une vidéo stockée.

Vous pouvez également obtenir de plus amples informations sur une célébrité reconnue, comme l'émotion exprimée et la présentation du genre. Pour de plus amples informations, veuillez consulter Reconnaissance de célébrités.

Pour reconnaître des célébrités sur des images, utilisez RecognizeCelebrities. Pour reconnaître des célébrités dans des vidéos stockées, utilisez StartCelebrityRecognition.

Détection de texte

La fonction Amazon Rekognition Texte dans les images peut détecter du texte dans des images et le convertir en texte lisible par une machine. Pour de plus amples informations, veuillez consulter Détection de texte.

Pour détecter du texte dans des images, utilisez DetectText.

Contenus inappropriés ou offensants

Amazon Rekognition peut analyser des images et des vidéos stockées à la recherche de contenus pour adultes et violents. Pour de plus amples informations, veuillez consulter Modération du contenu.

Pour détecter des images inappropriées, utilisez DetectModerationLabels. Pour détecter des vidéos stockées inappropriées, utilisez StartContentModeration.

Personnalisation

Certaines API d'analyse d'images proposées par Rekognition vous permettent d'améliorer la précision des modèles deep learning en créant des adaptateurs personnalisés entraînés sur vos propres données. Les adaptateurs sont des composants qui s'intègrent au modèle deep learning préentraîné de Rekognition, améliorant ainsi sa précision grâce à une connaissance du domaine basée sur vos images. Vous formez un adaptateur pour répondre à vos besoins en fournissant et en annotant des exemples d'images.

Une fois que vous avez créé un adaptateur, vous recevez un AdapterId. Vous pouvez le fournir AdapterId à une opération pour spécifier que vous souhaitez utiliser l'adaptateur que vous avez créé. Par exemple, vous fournissez le AdapterId à l'DetectModerationLabelsAPI pour l'analyse synchrone des images. AdapterId En les fournissant dans le cadre de la demande, Rekognition les utilisera automatiquement pour améliorer les prédictions relatives à vos images. Cela vous permet de tirer parti des fonctionnalités de Rekognition tout en les personnalisant en fonction de vos besoins.

Vous avez également la possibilité d'obtenir des prédictions pour des images en masse avec l'StartMediaAnalysisJobAPI. Pour plus d'informations, consultez Analyse en bloc.

Vous pouvez évaluer la précision des opérations de Rekognition en téléchargeant des images sur la console Rekognition et en exécutant une analyse sur ces images. Rekognition annote vos images à l'aide de la fonction sélectionnée, et vous pouvez ensuite revoir les prédictions, en utilisant les prédictions vérifiées pour déterminer quelles étiquettes bénéficieraient de la création d'un adaptateur.

À l'heure actuelle, vous pouvez utiliser des adaptateurs avec le DetectModerationLabels. Pour plus d'informations sur la création et l'utilisation d'adaptateurs, consultez Amélioration de la précision grâce à la modération personnalisée.

Analyse en bloc

Rekognition Bulk Analysis vous permet de traiter une grande collection d'images de manière asynchrone en utilisant un fichier manifeste en même temps que l'opération. StartMediaAnalysisJob Pour plus d'informations, consultez Analyse en bloc.

Comprendre les opérations d'imagerie et de vidéo de Rekognition

Amazon Rekognition propose deux ensembles d'API principaux pour l'analyse d'images et de vidéos :

- Amazon Rekognition Image : cette API est conçue pour analyser des images.
- Amazon Rekognition Video : cette API se concentre sur l'analyse des vidéos stockées et en streaming.

Les deux API peuvent détecter diverses entités telles que des visages et des objets. Pour une compréhension complète des types de comparaison et de détection pris en charge, reportez-vous à la section surComprendre les types d'analyse de Rekognition.

Fonctionnement d'Image Amazon Rekognition

Les opérations Amazon Rekognition Image sont synchrones. L'entrée et la réponse sont au format JSON. Les opérations Image Amazon Rekognition analysent une image d'entrée au format .jpg ou .png. L'image transmise à une opération Image Amazon Rekognition peut être stockée dans un compartiment Amazon S3. Si vous n'utilisez pas l'AWS CLI, vous pouvez également transmettre des octets d'images codés en Base64 directement à une opération Amazon Rekognition. Pour plus d'informations, consultez la section Utilisation des images.

Opérations de Vidéo Amazon Rekognition

L'API Amazon Rekognition Video facilite l'analyse des vidéos stockées dans un compartiment Amazon S3 ou diffusées via Amazon Kinesis Video Streams.

Pour les opérations vidéo enregistrées, notez ce qui suit :

- Les opérations sont asynchrones.
- L'analyse doit être lancée par une opération « Démarrer » (par exemple, StartFaceDetectionpour la détection de visages dans des vidéos enregistrées).
- L'état d'achèvement de l'analyse est publié dans une rubrique Amazon SNS.
- Pour récupérer les résultats d'une analyse, utilisez l'opération « Obtenir » correspondante (par exemple, GetFaceDetection).
- Pour plus d'informations, voir Utilisation de l'analyse vidéo stockée.

Pour l'analyse des vidéos en streaming :

- Les fonctionnalités incluent la recherche faciale dans les collections de vidéos de Rekognition et la détection d'étiquettes (objets ou concepts).
- Les résultats d'analyse des étiquettes sont envoyés sous forme de notifications Amazon SNS et Amazon S3.
- Les résultats de la recherche faciale sont envoyés dans un flux de données Kinesis.
- La gestion de l'analyse des vidéos en streaming est effectuée via un processeur de streaming Amazon Rekognition Video (par exemple, créez un processeur à l'aide de). CreateStreamProcessor
- Pour plus d'informations, voir Utilisation des événements vidéo en streaming.

Chaque opération d’analyse vidéo renvoie des métadonnées relatives à la vidéo analysée, ainsi qu’un identifiant de tâche et une étiquette de tâche. Des opérations telles que la détection d'étiquettes et la modération du contenu pour les vidéos permettent de trier par horodatage ou nom d'étiquette, et d'agréger les résultats par horodatage ou par segment.

Non-storage et opérations basées sur le stockage

Les opérations d’Amazon Rekognition sont regroupées selon les catégories suivantes.

- Non-storage Opérations d'API — Dans le cadre de ces opérations, Amazon Rekognition ne conserve aucune information. Vous fournissez des images et des vidéos d'entrée, l'opération exécute l'analyse et renvoie les résultats, mais rien n'est enregistré par Amazon Rekognition. Pour de plus amples informations, veuillez consulter Non-storage opérations.
- Storage-based Opérations d'API : les serveurs Amazon Rekognition peuvent stocker les informations faciales détectées dans des conteneurs appelés collections. Grâce à d'autres opérations API, Amazon Rekognition permet de rechercher des informations conservées sur un visage, et de réaliser une correspondance de visages. Pour de plus amples informations, veuillez consulter Storage-based Opérations d'API.

Utilisation de l'AWS SDK ou de HTTP pour l'appel des opérations d'API Amazon Rekognition

Vous pouvez appeler les opérations d'API Amazon Rekognition à l'aide de l'AWS SDK ou directement en utilisant HTTP. Sauf si vous avez une bonne raison de ne pas le faire, vous devez toujours utiliser l'AWS SDK. Les exemples Java de cette section utilisent l'AWS SDK . Il n'est pas fourni de fichier de projet Java, mais vous pouvez utiliser AWS Toolkit for Eclipse pour développer des applications AWS à l'aide de Java.

Les exemples .NET présentés dans cette section utilisent le kit AWS SDK pour .NET. Vous pouvez utiliser le kit AWS Toolkit for Visual Studio pour développer des applications AWS avec .NET. Vous y trouverez des modèles utiles, ainsi qu'AWS Explorer pour le déploiement d'applications et la gestion des services.

La Référence de l'API du présent guide couvre l'appel des opérations Amazon Rekognition à l'aide de HTTP. Pour en savoir plus sur la référence Java, consultez AWS SDK pour Java.

Les points de terminaison de service d'Amazon Rekognition que vous pouvez utiliser sont définis dans Régions et points de terminaison AWS.

Lorsque vous appelez Amazon Rekognition avec HTTP, utilisez les opérations HTTP POST.

Comprendre les opérations non liées au stockage et aux API de stockage

Amazon Rekognition offre deux types d'opérations API. Il s'agit des opérations hors stockage, au cours desquelles aucune information n'est stockée par Amazon Rekognition, et des opérations

de stockage au cours desquelles certaines informations de visages sont stockées par Amazon Rekognition.

Non-storage opérations

Amazon Rekognition offre des opérations API hors stockage pour les images suivantes :

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

Amazon Rekognition offre des opérations API hors stockage pour les vidéos suivantes :

- StartLabelDetection
- StartFaceDetection
- StartPersonTracking
- StartCelebrityRecognition
- StartContentModeration

Ces opérations sont appelées opérations d'API hors stockage, car lorsque vous appelez l'opération, Amazon Rekognition ne conserve aucune information relative au sujet de l'image d'entrée. Comme toutes les autres opérations d'API Amazon Rekognition, aucun octet d'image d'entrée n'est conservé par les opérations d'API hors stockage.

Les exemples de scénarios suivants montrent où peuvent être intégrées les opérations d'API hors stockage dans votre application. Ces scénarios partent du principe que vous disposez d'un référentiel d'images local.

Example 1 : Une application trouve dans votre référentiel local des images contenant des étiquettes spécifiques

Tout d'abord, vous détectez des étiquettes à l'aide de l'opération `DetectLabels` d'Amazon Rekognition dans chacune des images de votre référentiel, puis vous créez un index côté client, comme indiqué ci-dessous :

```
Label        ImageID

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

Ensuite, votre application peut consulter cet index pour rechercher dans votre référentiel local des images qui contiennent une étiquette spécifique. Par exemple, afficher des images qui contiennent un arbre.

Chaque étiquette détectée par Amazon Rekognition est associée à une valeur de fiabilité. Celle-ci indique le niveau de fiabilité correspondant à la présence de l'étiquette dans l'image d'entrée. Vous pouvez utiliser cette valeur pour exécuter un filtrage supplémentaire côté client des étiquettes. Le filtrage dépend des exigences de votre application en matière de fiabilité de la détection. Par exemple, si vous avez besoin d'étiquettes précises, vous pouvez filtrer et sélectionner uniquement les étiquettes ayant une fiabilité plus élevée (95 % ou plus, par exemple). Si votre application ne nécessite pas une valeur de fiabilité très élevée, vous pouvez choisir de filtrer les étiquettes ayant une valeur de fiabilité plus faible (plus proche de 50 %).

Example 2 : Une application permettant d'afficher des images de visage optimisées

Tout d'abord, vous pouvez détecter les visages dans chacune des images de votre référentiel local à l'aide de l'opération `DetectFaces` Amazon Rekognition, puis créer un index côté client. Pour chaque visage, l'opération renvoie des métadonnées qui incluent un cadre de délimitation, des repères faciaux (la position de la bouche et des oreilles, par exemple) et des attributs du visage (masculin ou féminin, par exemple). Vous pouvez stocker ces métadonnées dans un index local côté client, comme indiqué ci-dessous :

```
ImageID     FaceID     FaceMetaData
```

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

Dans cet index, la clé primaire est une combinaison de `ImageID` et de `FaceID`.

Ensuite, vous pouvez utiliser les informations contenues dans l'index pour améliorer les images de votre référentiel local lorsque votre application les affiche. Par exemple, vous pouvez ajouter un cadre de délimitation autour du visage ou mettre en évidence les caractéristiques faciales.

Storage-based Opérations d'API

Amazon Rekognition Image IndexFacesprend en charge cette opération, que vous pouvez utiliser pour détecter les visages sur une image et conserver les informations relatives aux traits faciaux détectés dans une collection Amazon Rekognition. Voici un exemple d'opération d'API de stockage, car le service conserve les informations sur le serveur.

Image Amazon Rekognition offre des opérations API hors stockage pour les images suivantes :

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

Vidéo Amazon Rekognition offre des opérations API de stockage suivantes :

- StartFaceSearch

- CreateStreamProcessor

Pour stocker des informations sur les visages, vous devez d'abord créer une collection de visages dans l'une des régions AWS de votre compte. Vous spécifiez cette collection de visages lorsque vous appelez l'opération `IndexFaces`. Une fois que vous avez créé une collection de visages et stocké les informations des caractéristiques faciales de tous les visages, vous pouvez explorer la collection à la recherche de correspondances de visage. Par exemple, vous pouvez détecter le plus grand visage dans une image et rechercher des visages semblables dans un ensemble en appelant `searchFacesByImage`.

Les informations de visage stockées dans les collections par `IndexFaces` sont accessibles pour les opérations Vidéo Amazon Rekognition. Par exemple, dans une vidéo, vous pouvez rechercher des personnes dont les visages correspondent à ceux contenus dans une collection existante, en appelant StartFaceSearch.

Pour en savoir plus sur la création et la gestion des collections, consultez Recherche de visages dans une collection.

> **Note**
>
> Les collections stockent des vecteurs de visages, qui sont des représentations mathématiques de visages. Les collections ne stockent pas d'images de visages.

Example 1 : Une application qui authentifie l'accès à un bâtiment

Vous commencez par la création d'une collection de visages pour stocker les images de badges numérisés à l'aide de l'opération `IndexFaces`, qui extrait les visages et les stocke comme vecteurs d'image explorables. Puis, lorsqu'un employé entre dans le bâtiment, une image de son visage est capturée et envoyée à l'opération `SearchFacesByImage`. Si la correspondance des visages offre un score de similarité suffisamment élevé (par exemple, 99 %), vous pouvez authentifier l'employé.

Comprendre le versionnement des modèles

Amazon Rekognition utilise des modèles de deep learning pour détecter des visages dans des collections. Les retours des clients et les avancées de la recherche en matière d'apprentissage approfondi permettent d'améliorer en permanence la précision de ses modèles. Ces améliorations sont envoyées en tant que mises à jour de modèle. Par exemple, avec la version 1.0 du modèle,

IndexFaces peut indexer les 15 plus grands visages dans une image. Les versions ultérieures du modèle permettent à `IndexFaces` d'indexer les 100 plus grandes visages d'une image.

Lorsque vous créez une nouvelle collection, elle est associée à la dernière version du modèle. Pour améliorer la précision, le modèle est régulièrement mis à jour.

Lorsqu'une nouvelle version du modèle est publiée, voici ce qui se passe :

- Les nouvelles collections que vous créez sont associées au dernier modèle. Les visages que vous avez ajoutés aux nouvelles collections en utilisant IndexFaces sont détectés à l'aide du dernier modèle.
- Vos collections existantes continuent d'utiliser la version du modèle avec lequel elles ont été créées. Les vecteurs faciaux stockés dans ces collections ne sont pas automatiquement mis à jour vers la dernière version du modèle.
- Les nouveaux visages ajoutés à une collection existante sont détectés à l'aide du modèle déjà associé à la collection.

Les différentes versions du modèle ne sont pas compatibles les unes avec les autres. Plus précisément, si une image est indexée dans plusieurs collections utilisant différentes versions du modèle, les identificateurs de visage d'un même visage détecté seront différents. Si une image est indexée dans plusieurs collections associées au même modèle, les identificateurs de visage seront les mêmes.

Votre application peut rencontrer des problèmes si la gestion de votre collection ne constitue pas de mises à jour du modèle. Vous pouvez déterminer la version du modèle utilisée par une collection grâce au champ `FaceModelVersion` renvoyé par les réponses de l'opération de collecte (par exemple, `CreateCollection`). Vous pouvez obtenir la version du modèle d'une collection existante en appelant DescribeCollection. Pour de plus amples informations, veuillez consulter Description d'une collection.

Les vecteurs faciaux existant dans une collection ne peuvent pas être mis à jour vers une version plus récente du modèle. Dans la mesure où Amazon Rekognition ne stocke pas d'octets d'images sources, il ne peut pas automatiquement réindexer d'images en utilisant une version ultérieure du modèle.

Pour utiliser le dernier modèle sur des visages stockés dans une collection existante, créez une nouvelle collection (CreateCollection) et réindexez les images sources dans la nouvelle collection (`Indexfaces`). Mettez à jour tous les identificateurs de visage stockés par votre application, car ceux

de la nouvelle collection et de l’ancienne sont différents. Si vous n'avez plus besoin de l'ancienne collection, vous pouvez la supprimer grâce à l'opération DeleteCollection.

Pour les opérations sans état, telles que DetectFaces, utilisez la dernière version du modèle.

Premiers pas avec Amazon Rekognition

Cette section inclut des rubriques vous permettant de commencer à utiliser Amazon Rekognition. Si vous découvrez Amazon Rekognition, nous vous recommandons dans un premier temps de passer en revue les concepts et la terminologie présentés dans Fonctionnement d'Amazon Rekognition.

Avant de pouvoir utiliser Rekognition, vous devez créer un compte AWS et obtenir un identifiant de compte AWS. Vous devez également créer un utilisateur, qui permet au système Amazon Rekognition de déterminer si vous disposez des autorisations nécessaires pour accéder à ses ressources.

Après avoir créé vos comptes, vous souhaiterez installer et configurer les AWS SDK AWS CLI et. Il vous AWS CLI permet d'interagir avec Amazon Rekognition et d'autres services via la ligne de commande, tandis que les AWS kits SDK vous permettent d'utiliser des langages de programmation tels que Java et Python pour interagir avec Amazon Rekognition.

Une fois que vous avez configuré AWS les SDK AWS CLI et, vous pouvez consulter quelques exemples d'utilisation des deux. Vous pouvez également consulter des exemples d'interaction avec Amazon Rekognition à l'aide de la console.

La console Amazon Rekognition vous permet d'utiliser les flux de travail d'analyse en masse et de modération personnalisée. Au-delà de ces fonctionnalités, vous pouvez également utiliser la console pour faire une démonstration de différentes fonctionnalités d'Amazon Rekognition. Notez que l'utilisation de l'API est requise pour utiliser la plupart des fonctionnalités d'Amazon Rekognition.

Rubriques

- Étape 1 : configurer un compte AWS et créer un utilisateur
- Étape 2 : configurer le AWS CLI and AWS Kits SDK
- Étape 3 : Commencer à utiliser le AWS CLI and AWS API SDK
- Étape 4 : mise en route à l'aide de la console Amazon Rekognition

Étape 1 : configurer un compte AWS et créer un utilisateur

Avant d'utiliser Amazon Rekognition pour la première fois, vous devez effectuer les tâches suivantes :

1. Ouvrez un AWS compte.

2. Créer un utilisateur.

Cette section du guide du développeur explique pourquoi et comment vous allez créer un compte AWS et un utilisateur.

Rubriques

- Créez un AWS Compte et utilisateur

Créez un AWS Compte et utilisateur

Comptes AWS

Lorsque vous vous inscrivez à Amazon Web Services (AWS), votre compte AWS est automatiquement inscrit à tous les services d'AWS, y compris Amazon Rekognition. Seuls les services que vous utilisez vous sont facturés.

Avec Amazon Rekognition, vous ne payez que les ressources que vous utilisez.

Si vous êtes un nouveau AWS client, vous pouvez commencer à utiliser Amazon Rekognition gratuitement. Pour plus d'informations, consultez la page Niveau d'offre gratuite d'AWS.

Reportez-vous à la section Inscrivez-vous pour un Compte AWS suivante pour les instructions de création de compte.

Si vous possédez déjà un AWS compte, ignorez la configuration du compte et créez un utilisateur administratif.

Utilisateurs

Les services dans AWS, tels que Amazon Rekognition, nécessitent que vous fournissiez vos informations d'identification lors de l'accès. Cela permet au service de déterminer si vous avez les autorisations nécessaires pour accéder aux ressources détenues par ce service.

Vous pouvez créer des clés d'accès pour que votre AWS compte puisse accéder aux API AWS CLI ou lorsque l'utilisation de la console nécessite votre mot de passe. Cependant, nous vous déconseillons d'accéder à AWS en utilisant les informations d'identification de l'utilisateur root du compte AWS. Nous vous recommandons plutôt d'utiliser Gestion des identités et des accès AWS (IAM) pour créer un utilisateur administratif.

Vous pouvez alors accéder à AWS à l'aide d'une URL spéciale et des informations d'identification de cet utilisateur administratif.

Si vous êtes inscrit à AWS, mais que vous n'avez pas créé d'utilisateur pour vous-même, vous pouvez le faire avec la console IAM. Reportez-vous à la prochaine section Créer un utilisateur avec accès administratif pour obtenir des instructions sur la création d'un utilisateur administratif.

Inscrivez-vous pour un Compte AWS

Pour commencer AWS, vous avez besoin d'un Compte AWS. Pour plus d'informations sur la création d'un Compte AWS, consultez la section Démarrage avec un Compte AWS dans le Guide de Gestion de compte AWS référence.

Étape 2 : configurer le AWS CLI and AWS Kits SDK

Rubriques

- Octroi d'un accès par programmation
- Utiliser Rekognition avec un AWS Kit SDK

Les étapes suivantes vous montrent comment installer le AWS Command Line Interface (AWS CLI) et AWS les kits SDK utilisés dans les exemples de cette documentation. Il existe différentes manières d'authentifier les appels du AWS SDK. Les exemples de ce guide supposent que vous utilisez un profil d'informations d'identification par défaut pour appeler des AWS CLI commandes et effectuer des opérations d'API du AWS SDK.

Pour une liste des AWS régions disponibles, voir Régions et points de terminaison dans le Référence générale d'Amazon Web Services.

Suivez les étapes pour télécharger et configurer les AWS kits de développement logiciel.

Pour configurer le AWS CLI et le AWS Kits SDK

1. Téléchargez et installez le AWS SDK AWS CLI et les kits de développement logiciel que vous souhaitez utiliser. Ce guide fournit des exemples pour Java AWS CLI, Python, Ruby Node.js, PHP, .NET et JavaScript. Pour plus d'informations sur l'installation AWS des kits SDK, consultez la section Outils pour Amazon Web Services.
2. Créez une clé d'accès pour l'utilisateur que vous avez créé dans Créez un AWS Compte et utilisateur.

a. Connectez-vous au Console de gestion AWS et ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.

b. Dans le panneau de navigation, choisissez utilisateurs.

c. Choisissez le nom de l'utilisateur que vous avez créé à l'étape Créez un AWS Compte et utilisateur.

d. Choisissez l'onglet Informations d'identification de sécurité.

e. Choisissez Create access key (Créer une clé d'accès). Choisissez ensuite Download .csv file (Télécharger un fichier .csv) pour enregistrer l'ID de clé d'accès et la clé d'accès secrète dans un fichier CSV sur votre ordinateur. Stockez le fichier dans un emplacement sûr. Vous ne pourrez pas accéder à la clé d'accès secrète à nouveau une fois que cette boîte de dialogue se sera fermée. Après avoir téléchargé le fichier CSV, choisissez Close.

3. Si vous avez installé le AWS CLI, vous pouvez configurer les informations d'identification et la région pour la plupart des AWS SDK en les saisissant `aws configure` à l'invite de commande. Sinon, suivez les instructions suivantes.

4. Sur votre ordinateur, créez le répertoire `.aws` dans le répertoire de base. Sur Unix-based les systèmes tels que Linux ou macOS, il se trouve à l'emplacement suivant :

```
~/.aws
```

Sur les systèmes Windows, il se trouve à l'emplacement suivant :

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

5. Dans le répertoire `.aws`, créez un fichier appelé `credentials`.

6. Ouvrez le fichier CSV d'informations d'identification que vous avez créé à l'étape 2, puis copiez son contenu dans le fichier `credentials`, en respectant le format suivant :

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

Substituez votre ID de clé d'accès et votre clé d'accès secrète pour your_access_key_id et your_secret_access_key.

7. Enregistrez le fichier `Credentials` et supprimez le fichier CSV.

8. Dans le répertoire `.aws`, créez un fichier appelé `config`.

9. Ouvrez le fichier `config` et entrez votre région au format suivant.

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

Indiquez la région AWS souhaitée (par exemple `us-west-2`) pour your_aws_region.

> **Note**
>
> Si vous ne sélectionnez pas une région, la région us-east-1 sera utilisée par défaut.

10. Enregistrez le fichier `config`.

Octroi d'un accès par programmation

Vous pouvez exécuter les exemples de code AWS CLI et de code de ce guide sur votre ordinateur local ou dans d'autres AWS environnements, tels qu'une instance Amazon Elastic Compute Cloud. Pour exécuter les exemples, vous devez autoriser l'accès aux opérations du AWS SDK qu'ils utilisent.

Rubriques

- Exécuter du code sur votre ordinateur local
- Exécuter du code dans AWS environnements

Exécuter du code sur votre ordinateur local

Pour exécuter du code sur un ordinateur local, nous vous recommandons d'utiliser des informations d'identification à court terme pour autoriser un utilisateur à accéder aux opérations du AWS SDK. Pour des informations spécifiques sur l'exécution des exemples de code AWS CLI et sur un ordinateur local, consultezUtiliser un profil sur votre ordinateur local.

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec des AWS personnes extérieures au Console de gestion AWS. La manière d'accorder l'accès par programmation dépend du type d'utilisateur qui accède AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l'une des options suivantes.

Quel utilisateur a besoin d'un accès programmatique ?	À	Méthode
IAM	(Recommandé) Utilisez les informations d'identification de la console comme informati ons d'identification temporair es pour signer les AWS CLI demandes programmatiques adressées aux AWS SDK ou AWS aux API.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour cela AWS CLI, consultez la section Connexion pour le développement AWS local dans le Guide de AWS Command Line Interface l'utilisateur. • Pour les AWS kits de développement, consultez la section Connexion pour le développement AWS local dans le Guide de référence AWS des kits de développe ment logiciel et des outils.
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées aux AWS CLI AWS SDK ou AWS aux API.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour cela AWS CLI, reportez-vous AWS CLI à la section Configuration de l'utilisation AWS IAM Identity Center dans le Guide de AWS Command Line Interface l'utilisateur. • Pour les AWS kits de développement logiciel, les outils et les AWS API, consultez la section

Quel utilisateur a besoin d'un accès programmatique ?	À	Méthode
		Authentification IAM Identity Center dans le Guide de référence AWS des kits de développement logiciel et des outils.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées aux AWS CLI AWS SDK ou AWS aux API.	Suivez les instructions de la section Utilisation des informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	À	Méthode
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes programmatiques adressées aux AWS CLI AWS SDK ou AWS aux API.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour cela AWS CLI, consultez la section Authentification à l'aide des informations d'identification utilisateur IAM dans le Guide de l'AWS Command Line Interface utilisateur. • Pour les AWS kits de développement logiciel et les outils, consultez la section Authentification à l'aide d'informations d'identification à long terme dans le Guide de référence AWS des kits de développement logiciel et des outils. • Pour les AWS API, consultez la section Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Utiliser un profil sur votre ordinateur local

Vous pouvez exécuter les exemples de code AWS CLI et de ce guide avec les informations d'identification à court terme que vous créez dansExécuter du code sur votre ordinateur local. Pour obtenir les informations d'identification et autres informations relatives aux paramètres, les exemples utilisent un profil nommé `profile-name`. Par exemple :

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

L'utilisateur représenté par le profil doit être autorisé à appeler les opérations du SDK Rekognition et les autres opérations du AWS SDK requises par les exemples.

Pour créer un profil qui fonctionne avec les exemples de code AWS CLI et, choisissez l'une des options suivantes. Assurez-vous que le nom du profil que vous créez est `profile-name`.

- Utilisateurs gérés par IAM : suivez les instructions de la section Basculer vers un rôle IAM (AWS CLI).
- Identité du personnel (utilisateurs gérés par AWS IAM Identity Center) : suivez les instructions de la section Configuration de l'interface de ligne de commande AWS à utiliser AWS IAM Identity Center. Pour les exemples de code, nous recommandons d'utiliser un environnement de développement intégré (IDE), qui prend en charge le kit d'outils AWS permettant l'authentification via IAM Identity Center. Pour les exemples Java, voir Commencer à créer avec Java. Pour les exemples Java, voir Commencer à créer avec Python. Pour plus d'informations, consultez informations d'identification du IAM Identity Center.

> **Note**
>
> Vous pouvez utiliser du code pour obtenir des informations d'identification à court terme. Pour plus d'informations, consultez Basculement vers un rôle IAM (AWS API). Pour IAM Identity Center, obtenez les informations d'identification à court terme pour un rôle en suivant les instructions de la section Obtenir les informations d'identification du rôle IAM pour l'accès à la CLI.

Exécuter du code dans AWS environnements

Vous ne devez pas utiliser les informations d'identification utilisateur pour signer des appels au AWS SDK dans AWS des environnements, tels que le code de production exécuté dans une AWS Lambda fonction. Au lieu de cela, configurez un rôle qui définit les autorisations dont votre code a besoin. Vous attachez ensuite le rôle à l'environnement dans lequel votre code s'exécute. La manière dont vous attachez le rôle et mettez à disposition les informations d'identification temporaires varie en fonction de l'environnement dans lequel votre code s'exécute :

- AWS Lambda function : utilisez les informations d'identification temporaires que Lambda fournit automatiquement à votre fonction lorsqu'elle assume le rôle d'exécution de la fonction Lambda. Les informations d'identification sont disponibles dans les variables d'environnement Lambda. Vous n'avez pas besoin d'indiquer de profil. Pour plus d'informations, consultez Rôle d'exécution Lambda.
- Amazon EC2 : utilisez le fournisseur d'informations d'identification du point de terminaison des métadonnées de l'instance Amazon EC2. Le fournisseur génère et actualise automatiquement les informations d'identification pour les rattacher à l'instance Amazon EC2 à l'aide du profil d'instance Amazon EC2. Pour plus d'informations, consultez Utilisation d'un rôle IAM pour accorder des autorisations à des applications s'exécutant sur des instances Amazon EC2.
- Amazon Elastic Container Service : utilisez le fournisseur d'informations d'identification du conteneur. Amazon ECS envoie et actualise les informations d'identification à un point de terminaison de métadonnées. Un rôle IAM de tâche que vous spécifiez fournit une stratégie pour gérer les informations d'identification utilisées par votre application. Pour plus d'informations consultez Interaction avec les services AWS.

Pour plus d'informations sur les fournisseurs d'informations d'identification, consultez la section Fournisseurs d'informations d'identification standardisés.

Utiliser Rekognition avec un AWS Kit SDK

AWS des kits de développement logiciel (SDK) sont disponibles pour de nombreux langages de programmation courants. Chaque kit SDK fournit une API, des exemples de code et de la documentation qui facilitent la création d'applications par les développeurs dans leur langage préféré.

Documentation SDK	Exemples de code
AWS SDK pour C++	AWS SDK pour C++ exemples de code
AWS CLI	AWS CLI exemples de code
AWS SDK pour Go	AWS SDK pour Go exemples de code
AWS SDK pour Java	AWS SDK pour Java exemples de code
AWS SDK pour JavaScript	AWS SDK pour JavaScript exemples de code
AWS SDK pour Kotlin	AWS SDK pour Kotlin exemples de code

Documentation SDK	Exemples de code
AWS SDK pour .NET	AWS SDK pour .NET exemples de code
AWS SDK pour PHP	AWS SDK pour PHP exemples de code
Outils AWS pour PowerShell	Outils AWS pour PowerShell exemples de code
AWS SDK pour Python (Boto3)	AWS SDK pour Python (Boto3) exemples de code
AWS SDK pour Ruby	AWS SDK pour Ruby exemples de code
AWS SDK pour Rust	AWS SDK pour Rust exemples de code
AWS SDK pour SAP ABAP	AWS SDK pour SAP ABAP exemples de code
AWS SDK pour Swift	AWS SDK pour Swift exemples de code

Pour plus d'exemples spécifiques à Rekognition, consultez Exemples de code pour Amazon Rekognition à l'aide de AWS Kits SDK.

> **Exemple de disponibilité**
>
> Vous n'avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code en utilisant le lien Faire un commentaire en bas de cette page.

Étape 3 : Commencer à utiliser le AWS CLI and AWS API SDK

Après avoir configuré les AWS kits SDK AWS CLI et que vous souhaitez utiliser, vous pouvez créer des applications utilisant Amazon Rekognition. La plupart des interactions avec Amazon Rekognition se font à l'aide des opérations d'API, bien que certaines fonctionnalités d'Amazon Rekognition disposent de flux de travail sur console.

Les rubriques suivantes vous montrent comment démarrer avec Amazon Rekognition Image et Amazon Rekognition Video via le ou les kits de développement logiciel (SDK). AWS CLI AWS

- Travail avec les images— Couvre le processus d'analyse des images avec Amazon Rekognition Image.
- Utilisation des opérations d'analyse vidéo stockées- Couvre le processus d'analyse des vidéos stockées et non diffusées avec Amazon Rekognition Video.
- Utilisation des vidéos streaming- Couvre le processus d'analyse des vidéos en streaming avec Amazon Rekognition Video.

Les sections répertoriées ci-dessus contiennent des exemples utilisant le AWS CLI. Si vous avez l'intention d'utiliser le AWS CLI, consultez la section suivante pour savoir comment formater vos appels d'API.

Formatage du AWS CLI exemples

Les AWS CLI exemples de ce guide sont formatés pour le système d'exploitation Linux. Pour utiliser les exemples avec Microsoft Windows, vous devez modifier le format JSON du paramètre `--image` et remplacer les barres obliques inverses (\) de saut de ligne par des accents circonflexes (^). Pour plus d'informations sur le format JSON, consultez Spécification des valeurs des paramètres pour l'interface de ligne de commande AWS.

Voici un exemple de AWS CLI commande formatée pour Microsoft Windows (notez que ces commandes ne s'exécuteront pas telles quelles, ce ne sont que des exemples de mise en forme) :

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

Vous pouvez également fournir une version JSON abrégée qui fonctionne à la fois sur Microsoft Windows et sur Linux.

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

Pour plus d'informations, consultez Utilisation de la syntaxe raccourcie avec l'interface de ligne de commande AWS.

Étape suivante

Étape 4 : mise en route à l'aide de la console Amazon Rekognition

Étape 4 : mise en route à l'aide de la console Amazon Rekognition

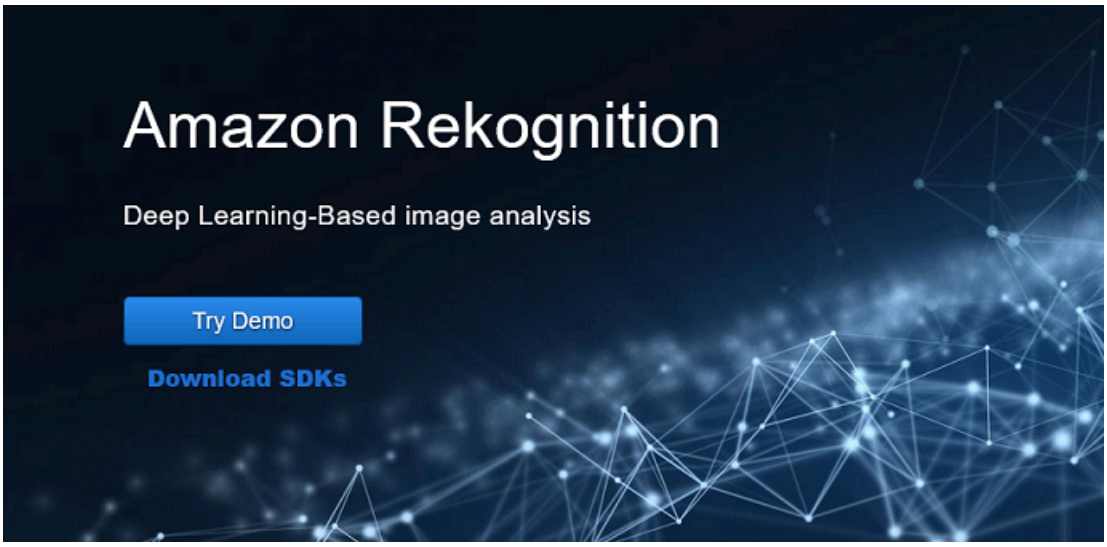
La console Amazon Rekognition vous permet de gérer les ressources liées aux étiquettes personnalisées Rekognition et à la fonctionnalité de modération personnalisée. La console propose uniquement des démonstrations des autres fonctionnalités de Rekognition.

Cette section vous montre comment utiliser un sous-ensemble des fonctionnalités d'Amazon Rekognition telles que la détection d'objet et de scène, l'analyse faciale et la comparaison des visages dans un ensemble d'images. Pour de plus amples informations, veuillez consulter Fonctionnement d'Amazon Rekognition. Vous pouvez également utiliser l' AWS CLI API Amazon Rekognition ou pour détecter des objets et des scènes, détecter des visages, comparer et rechercher des visages. Pour de plus amples informations, veuillez consulter Étape 3 : Commencer à utiliser le AWS CLI and AWS API SDK.

Cette section explique également comment consulter les CloudWatch métriques Amazon agrégées pour Rekognition à l'aide de la console Rekognition.

Rubriques

- Configurer des autorisations de console
- Exercice 1 : détecter des objets et des scènes (console)
- Exercice 2 : analyser les visages d'une image (console)
- Exercice 3 : comparer des visages dans des images (console)
- Exercice 4 :afficher les métriques agrégées (console)



Configurer des autorisations de console

Pour utiliser la console Rekognition, vous devez disposer des autorisations appropriées pour le rôle ou le compte accédant à la console. Pour certaines opérations, Rekognition créera automatiquement un compartiment Amazon S3 pour stocker les fichiers traités pendant le fonctionnement. Si vous souhaitez stocker vos fichiers d'entraînement dans un compartiment autre que ce compartiment de console, vous aurez besoin d'autorisations supplémentaires.

Autorisation d'accès à la console

Pour utiliser la console Rekognition, vous pouvez utiliser une politique IAM telle que la suivante, qui couvre Amazon S3 et la console de Rekognition. Pour plus d'informations sur l'attribution d'autorisations, consultez la section Attribution d'autorisations.

JSON

```
{
    "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:PutBucketCors",
                "s3:GetBucketCors"
            ],
            "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: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": "*"
        }
    ]
}
```

Accès aux compartiments Amazon S3 externes

Lorsque vous ouvrez la console Rekognition pour la première fois dans une nouvelle AWS région, Rekognition crée un bucket (bucket de console) qui est utilisé pour stocker les fichiers de projet. Vous pouvez également utiliser votre propre compartiment Amazon S3 (compartiment externe) pour télécharger les images ou le fichier manifeste sur la console. Pour utiliser un compartiment externe, ajoutez le bloc de stratégie suivant à la stratégie précédente.

```
{
            "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*"
```

```
            ]
}
```

Attribution d'autorisations

Pour activer l'accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

- Utilisateurs et groupes dans AWS IAM Identity Center (successeur d'AWS Single Sign-On) :

  Créez un jeu d'autorisations. Suivez les instructions de la section Création d'un ensemble d'autorisations dans le guide de l'utilisateur d'AWS IAM Identity Center (successeur d'AWS Single Sign-On).
- Utilisateurs gérés dans IAM par un fournisseur d'identité :

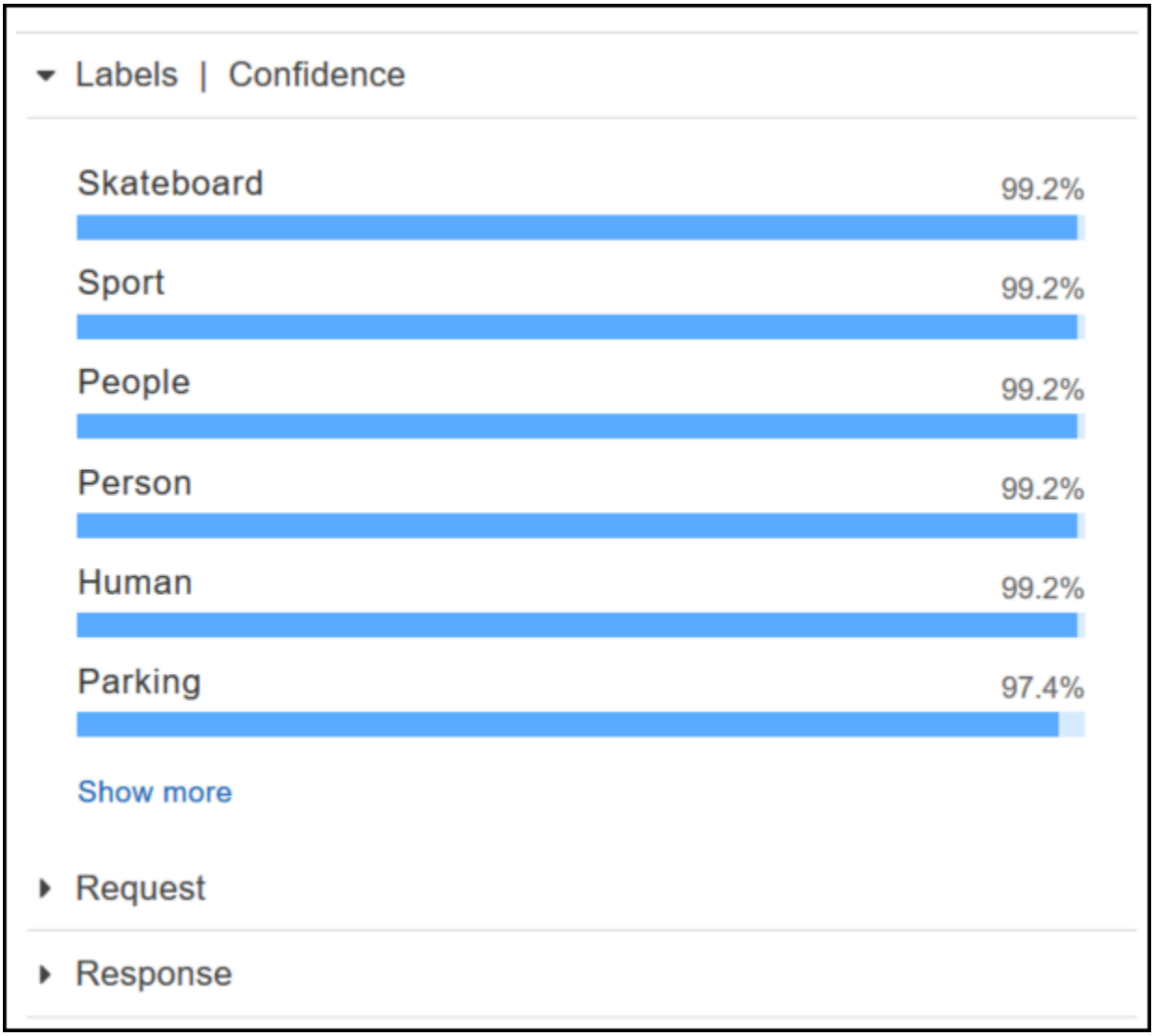
  Créez un rôle pour la fédération d'identité. Pour plus d'informations, voir la rubrique Création d'un rôle pour un fournisseur d'identité tiers (fédération) du Guide de l'utilisateur IAM.
- Utilisateurs IAM :
  - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d'un rôle pour un utilisateur IAM du Guide de l'utilisateur IAM.
  - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique Ajout d'autorisations à un utilisateur (console) du Guide de l'utilisateur IAM.

Exercice 1 : détecter des objets et des scènes (console)

Cette section montre comment fonctionne à un très haut niveau la détection des objets et des scènes par Amazon Rekognition. Lorsque vous spécifiez une image comme entrée, le service détecte les objets et les scènes de l'image, puis les retourne avec un score de pourcentage de fiabilité pour chaque objet et chaque scène.

Par exemple, Amazon Rekognition détecte les objets et scènes suivants dans l'exemple d'image : skateboard, sport, personne, auto, car et véhicule.

Amazon Rekognition retourne aussi un score de confiance pour chaque objet détecté dans l'exemple d'image, comme illustré dans l'exemple de réponse suivant.



Pour afficher tous les scores de fiabilité de la réponse, choisissez Show more (Afficher plus) dans le volet Labels | Confidence (Étiquettes | Fiabilité).

Vous pouvez aussi examiner la demande adressée à l'API et la réponse de l'API à titre de référence.

Demande

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

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

Réponse

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

Pour de plus amples informations, veuillez consulter Fonctionnement d'Amazon Rekognition.

Détection des objets et des scènes dans une image que vous fournissez

Vous pouvez charger une image qui vous appartient ou fournir l'URL d'une image comme entrée de la console Amazon Rekognition. Amazon Rekognition retourne les objets et les scènes, les scores de fiabilité de chaque objet et scène qu'il détecte dans l'image que vous avez fournie.

> **Note**
>
> La taille de l'image doit être inférieure à 5 MO et être au format JPEG ou PNG.

Pour détecter des objets et des scènes dans une image que vous fournissez

1. Ouvrez la console Amazon Rekognition à l'adresse. https://console.aws.amazon.com/rekognition/
2. Choisissez Détection d'étiquettes.
3. Effectuez l'une des actions suivantes :
   - Charger une image – Choisissez Charger, accédez à l'emplacement où vous avez stocké votre image et sélectionnez-la.
   - Utiliser une URL – Tapez l'URL dans la zone de texte, puis choisissez Go.
4. Affichez le score de fiabilité de chaque étiquette détectée dans le volet Etiquettes | Fiabilité.

Pour plus d'options d'analyse d'image, voir the section called "Travail avec les images".

Détecter des personnes et des objets dans une vidéo

Vous pouvez charger une vidéo qui vous appartient comme entrée de la console Amazon Rekognition. Amazon Rekognition renvoie les personnes, les objets et les étiquettes détectés dans la vidéo.

> **Note**
>
> La vidéo de démonstration ne doit pas durer plus d'une minute ni dépasser 30 Mo. Il doit être au format de fichier MP4 et encodé à l'aide du H.264 codec.

Pour détecter des personnes et des objets dans une vidéo

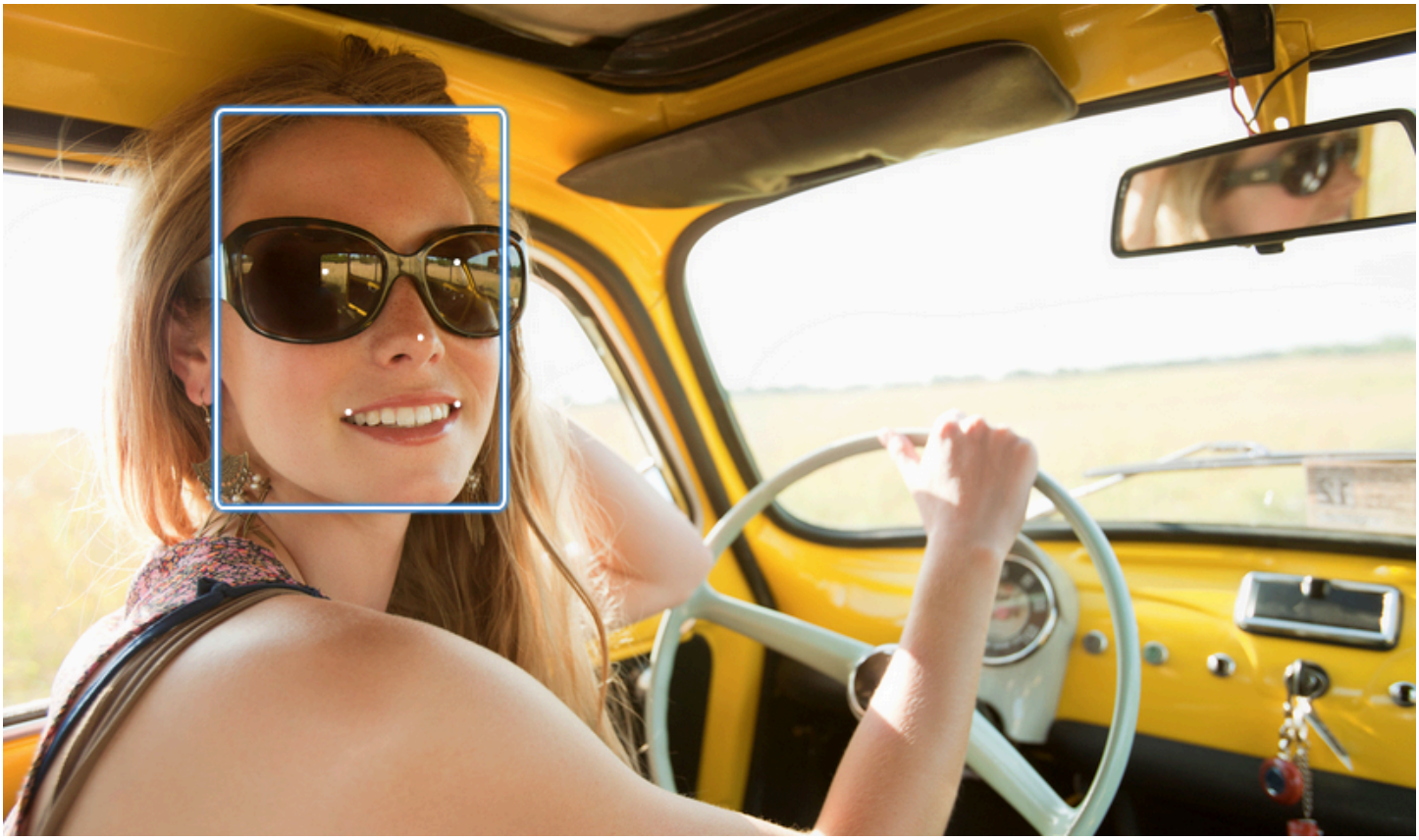
1. Ouvrez la console Amazon Rekognition à l'adresse. https://console.aws.amazon.com/rekognition/
2. Choisissez Stored Video Analysis dans la barre de navigation.
3. Sous Choisissez un extrait ou téléchargez le vôtre, sélectionnez Votre propre vidéo dans le menu déroulant.
4. Glissez et déposez votre vidéo ou sélectionnez-la depuis l'emplacement où vous l'avez stockée.

Pour plus d'options d'analyse vidéo, voir the section called "Utilisation des opérations d'analyse vidéo stockées" ou the section called "Utilisation des vidéos streaming".

Exercice 2 : analyser les visages d'une image (console)

Cette section vous montre comment utiliser la console Amazon Rekognition pour détecter les visages et analyser les attributs faciaux dans une image. Lorsque vous fournissez comme entrée une image qui contient un visage, le service détecte le visage dans l'image, analyse les attributs faciaux et retourne sous forme de pourcentage un score de fiabilité pour le visage et les attributs faciaux détectés dans l'image. Pour de plus amples informations, veuillez consulter Fonctionnement d'Amazon Rekognition.

Par exemple, si vous choisissez comme entrée l'exemple d'image suivant, Amazon Rekognition la détecte comme visage, et retourne les scores de confiance pour le visage et les attributs faciaux détectés.



L'exemple de réponse est le suivant.

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

Si l'image en entrée contient plusieurs visages, Rekognition en détecte jusqu'à 100. Chaque visage détecté est marqué avec un carré. Lorsque vous cliquez sur la zone marquée avec un carré sur un visage, Rekognition affiche le score de fiabilité du visage et de ses attributs détectés dans le volet Visages | Fiabilité.

Analyse des visages dans une image que vous fournissez

Vous pouvez charger votre propre image ou fournir l'URL de l'image dans la console Amazon Rekognition.

> **Note**
>
> La taille de l'image doit être inférieure à 5 MO et être au format JPEG ou PNG.

Pour analyser un visage dans une image que vous fournissez

1. Ouvrez la console Amazon Rekognition à l'adresse. https://console.aws.amazon.com/rekognition/
2. Choisissez Analyse faciale.
3. Effectuez l'une des actions suivantes :
   - Charger une image – Choisissez Charger, accédez à l'emplacement où vous avez stocké votre image et sélectionnez-la.
   - Utiliser une URL – Tapez l'URL dans la zone de texte, puis choisissez Go.
4. Afficher le score de fiabilité de l'un des visages détectés et de ses attributs faciaux dans le volet Visages | Fiabilité.
5. S'il y a plusieurs visages dans l'image, choisissez l'un des autres visages pour afficher ses attributs et scores.

Exercice 3 : comparer des visages dans des images (console)

Cette section vous montre comment utiliser la console Amazon Rekognition pour comparer des visages au sein d'un ensemble d'images comportant plusieurs visages. Lorsque vous spécifiez un visage de référence (source) et des visages de comparaison (cible), Rekognition compare le plus grand visage de l'image source (c'est-à-dire le visage de référence) à 100 visages maximum détectés dans l'image cible (c'est-à-dire les visages de comparaison), puis recherche le degré de

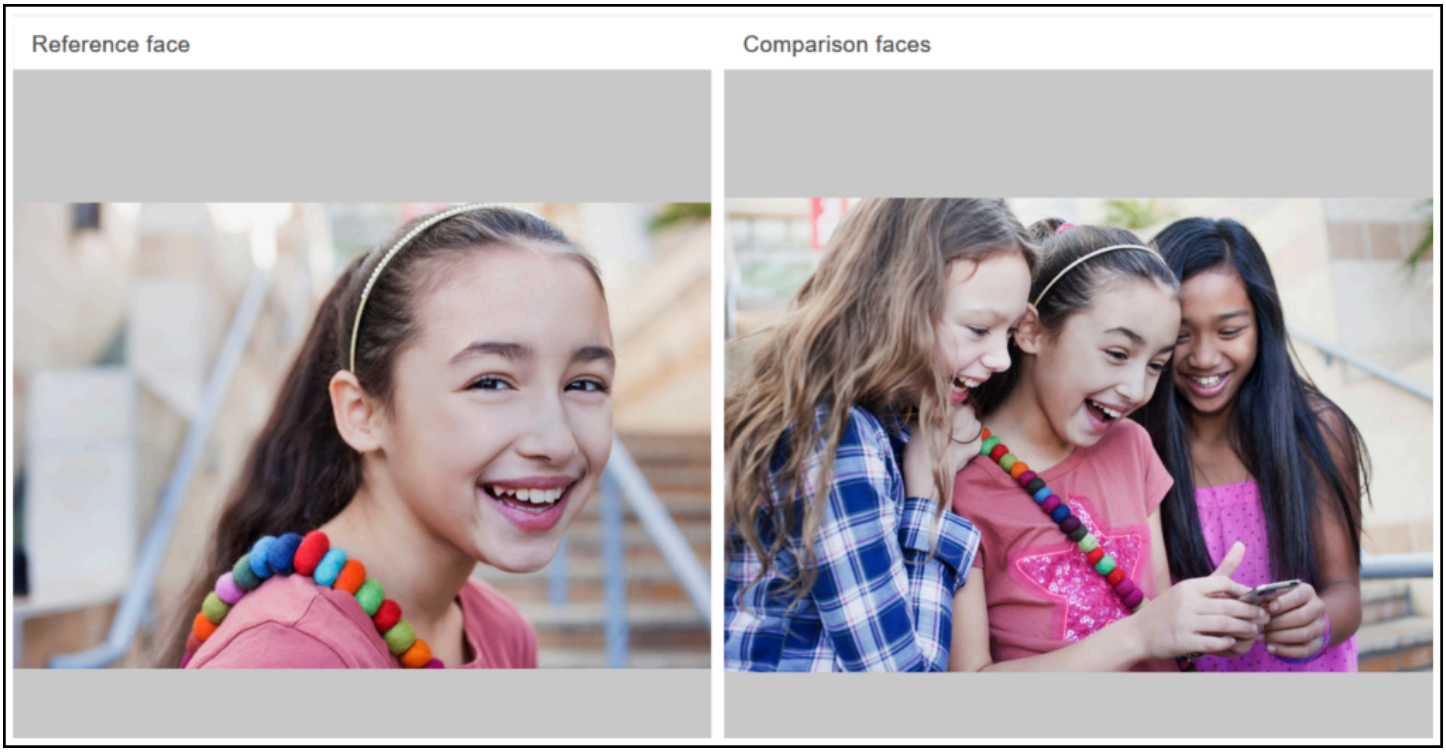
ressemblance entre le visage de l'image source et les visages de l'image cible. Le score de similarité pour chaque comparaison s'affiche dans le volet Résultats.

Si l'image cible contient plusieurs visages, Rekognition compare le visage de l'image source avec les 100 visages détectés dans l'image cible, puis affecte un score de similarité à chaque correspondance.

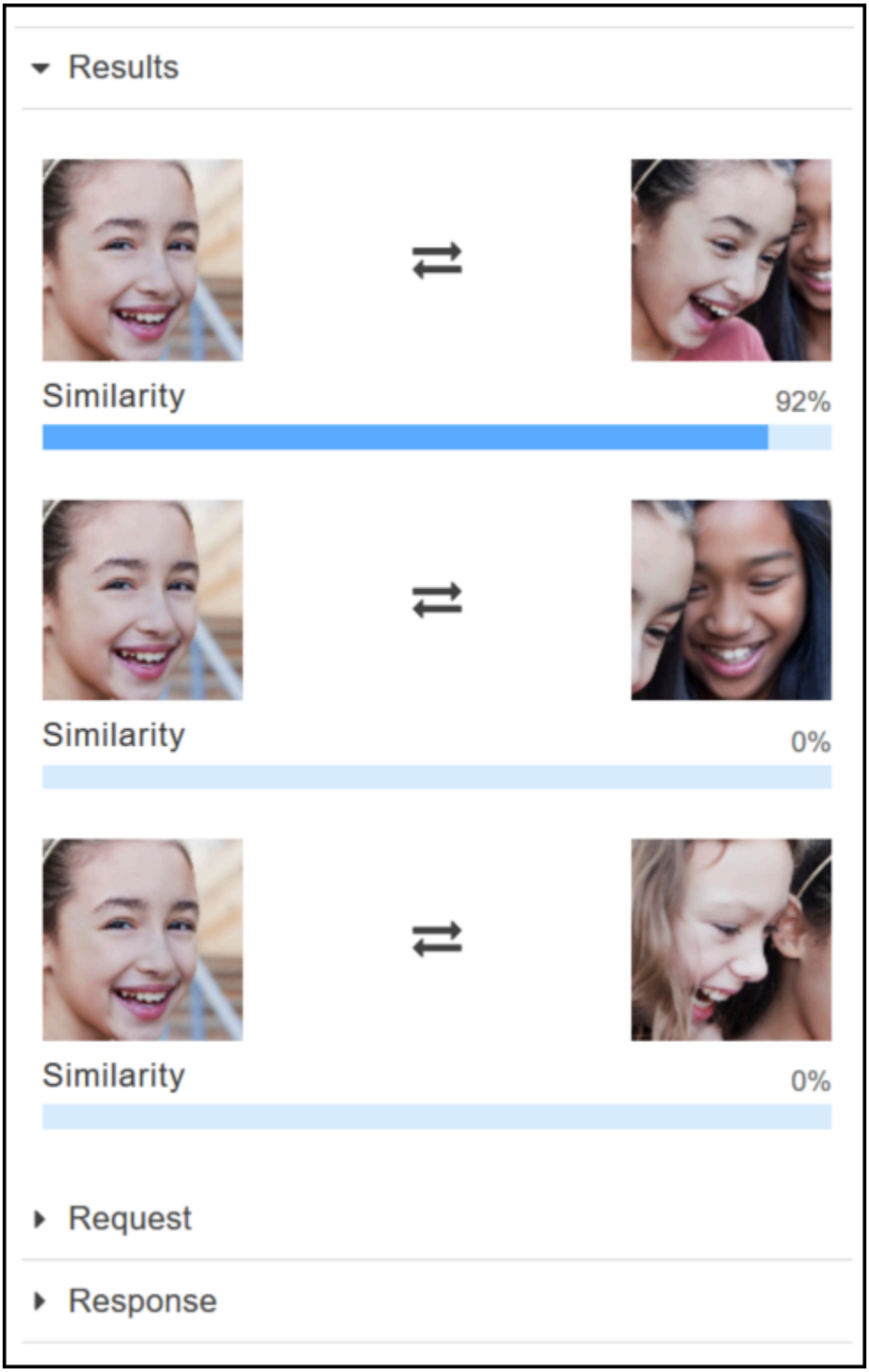
Si l'image source contient plusieurs visages, le service détecte le visage le plus grand de l'image source et le compare à chaque visage détecté dans l'image cible.

Pour de plus amples informations, veuillez consulter Comparaison de visages dans les images.

Par exemple, dans l'exemple d'image sur la gauche comme image source et l'exemple d'image sur la droite comme image cible, Rekognition détecte le visage de l'image source, le compare avec chaque visage détecté de l'image cible et affiche un score de similarité pour chaque couple.



Le tableau suivant indique les visages détectés dans l'image cible et le score de similarité de chaque visage.



Comparaison des visages dans une image que vous fournissez

Vous pouvez charger vos propres images source et cible pour que Rekognition puisse comparer les visages dans les images ou vous pouvez spécifier une URL pour l'emplacement des images.

> **Note**
>
> La taille de l'image doit être inférieure à 5 MO et être au format JPEG ou PNG.

Pour comparer les visages de vos images

1. Ouvrez la console Amazon Rekognition à l'adresse. https://console.aws.amazon.com/rekognition/
2. Choisissez Comparaison faciale.
3. Pour votre image source, effectuez l'une des actions suivantes :
   - Charger une image – Choisissez Charger sur la gauche, accédez à l'emplacement où vous avez stocké votre image source et sélectionnez-la.
   - Utiliser une URL – Tapez l'URL de votre image source dans la zone de texte, puis choisissez Go.
4. Pour votre image cible, effectuez l'une des actions suivantes :
   - Charger une image – Choisissez Charger sur la droite, accédez à l'emplacement où vous avez stocké votre image source et sélectionnez-la.
   - Utiliser une URL – Tapez l'URL de votre image source dans la zone de texte, puis choisissez Go.
5. Rekognition compare le plus grand visage de votre image source aux 100 visages (maximum) de l'image cible, puis il affiche le score de similarité pour chaque paire du volet Résultats.

Exercice 4 :afficher les métriques agrégées (console)

Le volet des métriques Amazon Rekognition affiche les graphiques d'activité pour un regroupement de métriques Rekognition sur une période de temps spécifiée. Par exemple, les métriques agrégées `SuccessfulRequestCount` indiquent le nombre total de demandes adressées à toutes les opérations d'API Rekognition ayant réussi durant les sept derniers jours.

Le tableau suivant répertorie les graphiques affichés dans le volet des métriques Rekognition et la métrique Rekognition correspondante. Pour de plus amples informations, veuillez consulter CloudWatch métriques pour Rekognition.

Graphe	Métrique agrégée
Appels réussis	SuccessfulRequestCount
Erreurs de client	UserErrorCount
Erreurs de serveur	ServerErrorCount
Limité	ThrottledCount
Etiquettes détectées	DetectedLabelCount
Visages détectés	DetectedFaceCount

Chaque graphique présente les données des métriques agrégées collectées sur une période de temps spécifiée. Le nombre total de données de métriques agrégées pour la période est également affiché. Pour afficher les métriques d'appels d'API individuels, choisissez le lien sous chaque graphique.

Pour autoriser les utilisateurs à accéder au volet des métriques de Rekognition, assurez-vous qu'ils disposent des autorisations de Rekognition appropriées. CloudWatch Par exemple, un utilisateur avec les autorisations de stratégie gérée `AmazonRekognitionReadOnlyAccess` `CloudWatchReadOnlyAccess` peut afficher le volet des métriques. Si un utilisateur ne dispose pas des autorisations requises, lorsqu'il ouvre le volet des métriques, aucun graphique ne s'affiche. Pour de plus amples informations, veuillez consulter Gestion des identités et des accès pour Amazon Rekognition.

Pour plus d'informations sur la surveillance de Rekognition à l'aide de. CloudWatch Surveillance de la Rekognition avec Amazon CloudWatch

Pour afficher les métriques agrégées (console)

1. Ouvrez la console Amazon Rekognition à l'adresse. https://console.aws.amazon.com/rekognition/
2. Dans le panneau de navigation, sélectionnez Métriques.
3. Dans la liste déroulante, sélectionnez la période pour laquelle vous voulez les métriques.
4. Pour mettre à jour les graphiques, choisissez le bouton Refresh.

5. Pour voir CloudWatch les statistiques détaillées d'une métrique agrégée spécifique, choisissez Voir les détails CloudWatch sous le graphique des métriques.

Utilisation d’images et de vidéos

Vous pouvez utiliser les opérations de l'API Amazon Rekognition avec trois types de médias différents : les images, les vidéos stockées et les vidéos en streaming. Cette section fournit des informations générales sur l'écriture de code permettant d'accéder à Amazon Rekognition pour traiter les différents types de médias. Pour obtenir des conseils sur les meilleures pratiques et les considérations à prendre en compte, consultez les sections respectives répertoriées ci-dessous, en fonction du type de média que vous traitez.

Les autres rubriques de ce guide fournissent des informations sur des types spécifiques d’analyse d’images et de vidéos, telles que la détection des visages.

Rubriques

- Travail avec les images
- Utilisation des opérations d'analyse vidéo stockées
- Utilisation des vidéos streaming
- Gestion des erreurs
- Utilisation d’Amazon Rekognition en tant que service autorisé FedRAMP

Travail avec les images

Cette section couvre les types d’analyse qu’Image Amazon Rekognition peut effectuer sur les images.

- Détection des objets et des scènes
- Détection et comparaison de visages
- Recherche de visages dans une collection
- Reconnaissance de célébrités
- Modération des images
- Détection de texte dans les images

Ces analyses sont effectuées par des opérations d’API hors stockage où Image Amazon Rekognition ne conserve pas les informations découvertes par l’opération. Les octets d’image en entrée ne sont

pas conservés par les opérations d'API hors stockage. Pour de plus amples informations, veuillez consulter Comprendre les opérations non liées au stockage et aux API de stockage.

Image Amazon Rekognition peut également stocker les métadonnées faciales dans des collections afin de les récupérer ultérieurement. Pour de plus amples informations, veuillez consulter Recherche de visages dans une collection.

Dans cette section, vous utilisez les opérations d'API Image Amazon Rekognition pour analyser les images stockées dans un compartiment S3, ainsi que les octets d'image chargés à partir du système de fichiers local. Cette section traite également l'obtention d'informations d'orientation d'image à partir d'une image .jpg.

Rekognition utilise uniquement les canaux RGB pour effectuer des inférences. AWS recommande aux utilisateurs de supprimer le canal Alpha avant d'utiliser un écran pour inspecter visuellement (manuellement par un humain) la comparaison.

Rubriques

- Spécifications d'images
- Analyse d'images stockées dans un compartiment Amazon S3
- Analyse d'une image chargée à partir d'un système de fichiers local
- Affichage de cadres de délimitation
- Obtention de l'orientation d'une image et des coordonnées du cadre de délimitation

Spécifications d'images

Les opérations Image Amazon Rekognition peuvent analyser les images au format .jpg ou .png.

Vous transmettez les octets des images à une opération Image Amazon Rekognition dans le cadre de l'appel ou vous référencez un objet Amazon S3 existant. Pour obtenir un exemple d'analyse d'une image stockée dans un compartiment Amazon S3, consultez Analyse d'images stockées dans un compartiment Amazon S3. Pour obtenir un exemple de transmission d'octets d'image à une opération d'API Image Amazon Rekognition consultez Analyse d'une image chargée à partir d'un système de fichiers local.

Si vous utilisez HTTP et transmettez les octets d'image dans le cadre d'une opération Image Amazon Rekognition, ils doivent former une chaîne encodée en base64. Si vous utilisez le kit de développement logiciel AWS SDK et transmettez les octets d'image dans le cadre de l'appel d'opération d'API, la nécessité de les encoder en base64 dépend du langage que vous utilisez.

Les AWS SDK courants suivants encodent automatiquement les images en base64, et vous n'avez pas besoin de coder des octets d'image avant d'appeler une opération d'API Amazon Rekognition Image.

- Java
- JavaScript
- Python
- PHP

Si vous utilisez un autre kit AWS SDK et obtenez une erreur de format d'image en appelant une opération Rekognition API, essayez d'encoder les octets d'image en base64 avant de les transmettre à une opération Rekognition API.

Si vous utilisez le AWS CLI pour appeler les opérations Amazon Rekognition Image, le transfert d'octets d'image dans le cadre de l'appel n'est pas pris en charge. Vous devez d'abord charger l'image dans un compartiment Amazon S3, puis appeler l'opération faisant référence à l'image chargée.

> **Note**
>
> Il n'est pas nécessaire d'encoder l'image en base64 si vous transmettez une image stockée dans une propriété `S3Object` plutôt que des octets d'image.

Pour savoir comment limiter autant que possible la latence pour les opérations Image Amazon Rekognition, consultez Latence de fonctionnement d'Image Amazon Rekognition.

Correction de l'orientation d'une image

Dans plusieurs opérations Rekognition API l'orientation d'une image analysée est renvoyée. Il est important de connaître l'orientation d'une image, car elle vous permet de réorienter les images à afficher. Les opérations Rekognition API chargées d'analyser les visages renvoient également des cadres de délimitation correspondant à l'emplacement des visages dans une image. Vous pouvez utiliser des cadres de délimitation pour afficher un cadre autour d'un visage sur une image. Les coordonnées de cadre de délimitation renvoyées variant en fonction de l'orientation de l'image, vous pouvez être amené à traduire ces coordonnées pour afficher correctement un cadre autour d'un visage. Pour de plus amples informations, veuillez consulter Obtention de l'orientation d'une image et des coordonnées du cadre de délimitation.

Redimensionnement d'images

Au cours de l'analyse, Amazon Rekognition redimensionne les images en interne à l'aide d'un ensemble de plages prédéfinies qui conviennent le mieux à un modèle ou à un algorithme en particulier. De ce fait, Amazon Rekognition peut détecter un nombre différent d'objets ou fournir des résultats différents en fonction de la résolution de l'image d'entrée. Par exemple, supposons que vous avez deux images. La première image a une résolution de 1024 x 768 pixels. La deuxième image, une version redimensionnée de la première image, a une résolution de 640 x 480 pixels. Si vous soumettez les images à DetectLabels, les réponses aux deux appels `DetectLabels` peuvent légèrement différer.

Analyse d'images stockées dans un compartiment Amazon S3

Image Amazon Rekognition peut analyser des images stockées dans un compartiment Amazon S3, ou des images fournies sous forme d'octets d'image.

Dans cette rubrique, vous utilisez l'opération DetectLabelsAPI pour détecter des objets, des concepts et des scènes dans une image (JPEG ou PNG) stockée dans un compartiment Amazon S3. Pour transmettre une image à une opération d'API Image Amazon Rekognition, utilisez le paramètre d'entrée Image. Dans `Image`, vous spécifiez la propriété d'objet S3Object pour référencer une image stockée dans un compartiment S3. Il n'est pas nécessaire d'encoder en base64 les octets d'image pour les images stockées dans des compartiments Amazon S3. Pour de plus amples informations, veuillez consulter Spécifications d'images.

Exemple de demande

Dans cet exemple de demande JSON `DetectLabels`, l'image source (`input.jpg`) est chargée à partir d'un compartiment Amazon S3 nommé `amzn-s3-demo-bucket`. La région du compartiment S3 contenant l'objet S3 doit correspondre à la région que vous utilisez pour les opérations Image Amazon Rekognition.

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

```
}
```

Les exemples suivants utilisent différents AWS SDK et le AWS CLI to call`DetectLabels`. Pour en savoir plus sur la réponse d'opération `DetectLabels`, consultez DetectLabels réponse.

Pour détecter des étiquettes dans une image

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur avec `AmazonRekognitionFullAccess` et autorisations `AmazonS3ReadOnlyAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK. Assurez-vous que vous avez accordé à l'utilisateur qui appelle les opérations d'API les autorisations appropriées pour l'accès par programmation. Consultez Octroi d'un accès par programmation les instructions sur la procédure à suivre.

2. Chargez une image qui contient un ou plusieurs objets - par exemple des arbres, des maisons et un bateau - dans votre compartiment S3. L'image doit être au format .jpg ou .png.

   Pour en savoir plus, consultez Chargement d'objets dans Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

3. Utilisez les exemples suivants pour appeler l'opération `DetectLabels`.

   Java

   Cet exemple affiche la liste des étiquettes qui ont été détectées dans l'image d'entrée. Remplacez les valeurs de `bucket` et `photo` par le nom du compartiment Amazon S3 et le nom de l'image utilisés à l'étape 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

Cet exemple affiche la sortie JSON de l'opération `detect-labels` de l'interface de ligne de commande (CLI). Remplacez les valeurs de `bucket` et de `photo` par le nom du compartiment Amazon S3 et le nom de l'image utilisés à l'étape 2. Remplacez la valeur de

`profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
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
```

Si vous utilisez Windows, vous devrez peut-être éviter les guillemets, comme indiqué dans l'exemple ci-dessous.

```
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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
//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

Cet exemple affiche les étiquettes qui ont été détectées dans l'image d'entrée. Remplacez les valeurs de `bucket` et de `photo` par le nom du compartiment Amazon S3 et le nom de l'image utilisés à l'étape 2. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

Cet exemple affiche des informations sur les étiquettes qui sont détectées dans une image.

Remplacez la valeur de `photo` par le chemin et le nom d'un fichier image qui contient un ou plusieurs visages de célébrités. Remplacez la valeur de `bucket` par le nom du compartiment S3 qui contient le fichier image. Remplacez la valeur de `REGION` par le nom de la région associée à votre utilisateur. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
// 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

Cet exemple affiche la liste des étiquettes qui ont été détectées dans l'image d'entrée. Remplacez les valeurs de `bucket` et `photo` par le nom du compartiment Amazon S3 et le nom de l'image utilisés à l'étape 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

Cet exemple affiche la liste des étiquettes qui ont été détectées dans l'image d'entrée. Remplacez les valeurs de `bucket` et de `photo` par le nom du compartiment Amazon S3 et le nom de l'image utilisés à l'étape 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
```

Exemple de réponse

La réponse de `DetectLabels` est un tableau qui recense les étiquettes détectées dans l'image et qui indique le niveau de fiabilité de la détection.

Lorsque vous effectuez l'opération `DetectLabels` sur une image, Amazon Rekognition renvoie un résultat similaire à l'exemple de réponse suivant.

La réponse montre que l'opération a détecté plusieurs étiquettes, notamment Person (Personne), Vehicle (Véhicule) et Car (Voiture). A chaque étiquette est associé un niveau de fiabilité. Par exemple, l'algorithme de détection est certain à 98,991432 % que l'image contient une personne.

La réponse comprend également les étiquettes d'ancêtre d'une étiquette dans le tableau `Parents`. Par exemple, l'étiquette Automobile possède deux étiquettes parents nommées Vehicle et Transportation.

La réponse pour les étiquettes d'objets courants comprend des informations de cadre de délimitation pour l'emplacement de l'étiquette sur l'image d'entrée. Par exemple, l'étiquette Personne comporte un tableau d'instances contenant deux cadres de délimitation. Il s'agit des emplacements de deux personnes détectées dans l'image.

Le champ `LabelModelVersion` contient le numéro de version du modèle de détection utilisé par `DetectLabels`.

Pour plus d'informations sur l'utilisation de l'opération `DetectLabels`, consultez Détection d'objets et de concepts.

```
{

    {
    "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"
}


}
```

Analyse d’une image chargée à partir d’un système de fichiers local

Les opérations Image Amazon Rekognition peuvent analyser les images fournies sous forme d’octets d’image ou les images stockées dans un compartiment Amazon S3.

Ces rubriques proposent des exemples où des octets d'image sont fournis aux opérations d'API Image Amazon Rekognition en utilisant un fichier chargé à partir d'un système de fichiers local. Pour transmettre des octets d'image à une opération d'API Amazon Rekognition, vous utilisez le paramètre d'entrée Image. Dans `Image`, vous spécifiez la propriété `Bytes` pour transmettre des octets d'image encodés en base64.

Les octets d'image transmis à une opération d'API Amazon Rekognition à l'aide du paramètre d'entrée `Bytes` doivent être encodés en base64. Les kits AWS SDK utilisés dans ces exemples encodent automatiquement les images en base64. Vous n'avez pas besoin de coder les octets d'image avant d'appeler une opération d'API Amazon Rekognition. Pour de plus amples informations, veuillez consulter Spécifications d'images.

Dans cet exemple de demande JSON `DetectLabels`, les octets d'image source sont transmis dans le paramètre d'entrée `Bytes`.

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

Les exemples suivants utilisent différents AWS SDK et le AWS CLI to call`DetectLabels`. Pour en savoir plus sur la réponse d'opération `DetectLabels`, consultez DetectLabels réponse.

Pour un JavaScript exemple côté client, voir. En utilisant JavaScript

Pour détecter des étiquettes dans une image locale

1. Si vous ne l'avez pas déjà fait :
   a. Créez ou mettez à jour un utilisateur avec `AmazonRekognitionFullAccess` et autorisations `AmazonS3ReadOnlyAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.
   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.
2. Utilisez les exemples suivants pour appeler l'opération `DetectLabels`.

Java

L'exemple Java suivant montre comment charger une image à partir du système de fichiers local et comment détecter les étiquettes à l'aide de l'opération detectLabels de l'AWS SDK. Remplacez la valeur de `photo` par le chemin et le nom d'un fichier image (au format .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

L'exemple AWS SDK pour Python suivant montre comment charger une image à partir du système de fichiers local et comment appeler l'opération detect_labels. Remplacez la valeur de `photo` par le chemin et le nom d'un fichier image (au format .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

L'exemple suivant montre comment charger une image à partir du système de fichiers local et détecter les étiquettes à l'aide de l'opération `DetectLabels`. Remplacez la valeur de `photo` par le chemin et le nom d'un fichier image (au format .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

L'exemple de SDK AWS pour PHP suivant montre comment charger une image depuis le système de fichiers local et appeler DetectFacesl'opération d'API. Remplacez la valeur de `photo` par le chemin et le nom d'un fichier image (au format .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

Cet exemple affiche la liste des étiquettes qui ont été détectées dans l'image d'entrée. Remplacez la valeur de `photo` par le chemin et le nom d'un fichier image (au format .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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
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);
        }
    }
}
```

En utilisant JavaScript

L'exemple de JavaScript page Web suivant permet à un utilisateur de choisir une image et de visualiser l'âge estimé des visages détectés sur l'image. Les âges estimés sont renvoyés par un appel à DetectFaces.

L'image choisie est chargée à l'aide de la JavaScript `FileReader.readAsDataURL` fonction qui encode l'image en base64. Cela permet d'afficher l'image sur un canevas HTML. Mais cela signifie que les octets d'image doivent être décodés avant d'être transmis à une opération Image Amazon Rekognition. Cet exemple montre comment décoder les octets d'image chargés. Si les octets d'image codés ne sont pas utiles pour vous, utilisez plutôt `FileReader.readAsArrayBuffer`, car l'image chargée n'est pas codée. Cela signifie que les opérations Image Amazon Rekognition peuvent

être appelées sans que les octets d'image soient décodés au préalable. Pour obtenir un exemple, consultez Utilisation de la lecture AsArrayBuffer.

Pour exécuter l' JavaScript exemple

1. Chargez l'exemple de code source dans un éditeur.
2. Obtenez l'ID de la réserve d'identités Amazon Cognito. Pour de plus amples informations, veuillez consulter Obtenir l'ID de la réserve d'identités Amazon Cognito.
3. Dans la fonction `AnonLog` de l'exemple de code, remplacez `IdentityPoolIdToUse` et `RegionToUse` par les valeurs que vous avez notées à l'étape 9 de Obtenir l'ID de la réserve d'identités Amazon Cognito.
4. Dans la fonction `DetectFaces`, remplacez `RegionToUse` par la valeur que vous avez utilisée à l'étape précédente.
5. Enregistrez l'exemple de code source en tant que fichier `.html`.
6. Chargez le fichier dans votre navigateur.
7. Choisissez le bouton Parcourir..., puis choisissez une image qui contient un ou plusieurs visages. La table qui s'affiche contient les âges estimés pour chaque visage détecté dans l'image.

> **Note**
>
> L'exemple de code suivant utilise deux scripts qui ne font plus partie d' Amazon Cognito. Pour obtenir ces fichiers, suivez les liens pour aws-cognito-sdk.min.js et amazon-cognito-identity.min.js, puis enregistrez le texte de chaque fichier en tant que fichiers individuels `.js`.

JavaScript exemple de code

L'exemple de code suivant utilise la JavaScript version V2. Pour un exemple dans la JavaScript version 3, consultez l'exemple dans le référentiel d'exemples GitHub du SDK de AWS documentation.

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

Utilisation de la lecture AsArrayBuffer

L'extrait de code suivant est une implémentation alternative de la `ProcessImage` fonction dans l'exemple de code, à l'aide JavaScript de la version V2. Il utilise `readAsArrayBuffer` pour charger une image et appeler `DetectFaces`. Comme `readAsArrayBuffer` n'encode pas le fichier chargé en base64, il n'est pas nécessaire de décoder les octets d'image avant d'appeler une opération Image Amazon Rekognition.

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

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

    // Load base64 encoded image for display
    var reader = new FileReader();
    reader.onload = (function (theFile) {
      return function (e) {
        //Call Rekognition
        AWS.region = "RegionToUse";
        var rekognition = new AWS.Rekognition();
        var params = {
          Image: {
          Bytes: e.target.result
```

```
        },
        Attributes: [
        'ALL',
      ]
    };
    rekognition.detectFaces(params, function (err, data) {
      if (err) console.log(err, err.stack); // an error occurred
      else {
       var table = "<table><tr><th>Low</th><th>High</th></tr>";
        // show each face and build out estimated age table
        for (var i = 0; i < data.FaceDetails.length; i++) {
          table += '<tr><td>' + data.FaceDetails[i].AgeRange.Low +
            '</td><td>' + data.FaceDetails[i].AgeRange.High + '</td></tr>';
        }
        table += "</table>";
        document.getElementById("opResult").innerHTML = table;
      }
    });

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

Obtenir l’ID de la réserve d'identités Amazon Cognito

Pour plus de simplicité, l’exemple utilise une réserve d'identités anonyme Amazon Cognito pour fournir un accès non authentifié à l’API Image Amazon Rekognition. Cela peut répondre à vos besoins. Par exemple, vous pouvez utiliser un accès non authentifié pour offrir un accès gratuit ou un accès test à votre site web avant que les utilisateurs ne s’inscrivent. Pour fournir un accès authentifié, utilisez un groupe d’utilisateurs Amazon Cognito. Pour plus d’informations, consultez Groupe d’utilisateurs dans Amazon Cognito.

La procédure suivante montre comment créer une réserve d'identités qui permet d’accéder à des identités non authentifiées, et comment obtenir l’identifiant de la réserve d'identités requis dans l’exemple de code.

Pour obtenir l’identifiant de la réserve d'identités

1. Ouvrez la console Amazon Cognito.
2. Sélectionnez Create new identity pool.

3. Pour Identity pool name* (Nom de la réserve d'identités*), tapez un nom pour votre réserve d'identités.
4. Dans Identités non authentifiées, choisissez Activer l'accès aux identités non authentifiées.
5. Sélectionnez Créer une réserve.
6. Choisissez View Details (Afficher les détails) et notez le nom du rôle pour les identités non authentifiées.
7. Sélectionnez Allow (Autoriser).
8. Dans Plateforme, sélectionnez JavaScript.
9. Dans Obtenir les informations d'identification AWS, notez les valeurs de `AWS.config.region` et `IdentityPooldId` qui sont affichées dans l'extrait de code.
10. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
11. Dans le panneau de navigation, choisissez Rôles.
12. Choisissez le nom du rôle que vous avez noté à l'étape 6.
13. Dans l'onglet Autorisations, choisissez Attacher des stratégies.
14. Sélectionnez AmazonRekognitionReadOnlyAccess.
15. Choisissez Attach Policy (Attacher une politique).

Affichage de cadres de délimitation

Les opérations Image Amazon Rekognition peuvent renvoyer les coordonnées des cadres de délimitation pour les éléments détectés dans les images. Par exemple, l'opération DetectFaces renvoie un cadre de délimitation (BoundingBox) pour chaque visage détecté dans une image. Vous pouvez utiliser les coordonnées du cadre de délimitation pour afficher un cadre autour des éléments détectés. Par exemple, l'image suivante illustre un cadre de délimitation autour d'un visage.

La propriété `BoundingBox` dispose des valeurs suivantes :

- Height : hauteur du cadre de délimitation sous forme de ratio de la hauteur d'image globale.
- Left : coordonnée gauche du cadre de délimitation sous forme de ratio de la largeur d'image globale.
- Top : coordonnée supérieure du cadre de délimitation sous forme de ratio de la hauteur d'image globale.
- Width : largeur du cadre de délimitation sous forme de ratio de la largeur d'image globale.

Chaque BoundingBox propriété possède une valeur comprise entre 0 et 1. Chaque valeur de propriété est un ratio de la largeur d'image globale (`Left` et `Width`) ou de la hauteur d'image globale (`Height` et `Top`). Par exemple, si l'image d'entrée mesure 700 x 200 pixels et que les coordonnées en haut à gauche du cadre de délimitation sont 350 x 50 pixels, l'API renvoie une `Left` valeur de 0,5 (350/700) et une `Top` valeur de 0,25 (). 50/200

Le schéma suivant illustre la plage d'une image couverte par chaque propriété du cadre de délimitation.

Pour afficher le cadre de sélection à l'emplacement et à la taille corrects, vous devez multiplier les BoundingBox valeurs par la largeur ou la hauteur de l'image (selon la valeur souhaitée) pour obtenir les valeurs en pixels. Vous utilisez les valeurs en pixels pour afficher le cadre de délimitation. Par exemple, les dimensions en pixels de l'image précédente sont de 608 (largeur) x 588 (hauteur). Les valeurs du cadre de délimitation pour le visage sont les suivantes :

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

L'emplacement du cadre de délimitation du visage en pixels est calculé comme suit :

```
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
```

Vous utilisez ces valeurs pour afficher un cadre de délimitation autour du visage.

> **Note**
>
> Une image peut être orientée de diverses manières. Pour qu'une image s'affiche avec l'orientation corrigée, il se peut que l'application ait besoin de la faire pivoter. Les coordonnées du cadre de délimitation sont affectées par l'orientation de l'image. Vous aurez peut-être besoin de convertir les coordonnées pour pouvoir afficher un cadre de délimitation

à l'emplacement approprié. Pour de plus amples informations, veuillez consulter Obtention de l'orientation d'une image et des coordonnées du cadre de délimitation.

Les exemples suivants indiquent comment afficher un cadre de délimitation autour de visages détectés en appelant DetectFaces. Les exemples supposent que l'orientation des images est de 0 degré. Ils montrent aussi comment télécharger l'image à partir d'un compartiment Amazon S3.

Pour afficher un cadre de délimitation

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur avec `AmazonRekognitionFullAccess` et autorisations `AmazonS3ReadOnlyAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Utilisez les exemples suivants pour appeler l'opération `DetectFaces`.

   Java

   Remplacez la valeur de `bucket` par le compartiment Amazon S3 qui contient le fichier image. Remplacez la valeur de `photo` par le nom d'un fichier image (au format .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

Remplacez la valeur de `bucket` par le compartiment Amazon S3 qui contient le fichier image. Remplacez la valeur de `photo` par le nom d'un fichier image (au format .jpg ou .png). Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

Notez que `s3` fait référence au client Amazon S3 de l'AWS SDK et `rekClient` au client Amazon Rekognition de l'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]
```

```
}
```

Obtention de l’orientation d’une image et des coordonnées du cadre de délimitation

Il est courant que les applications qui utilisent Image Amazon Rekognition soient appelées à afficher les images détectées par les opérations Image Amazon Rekognition et les cadres entourant les visages détectés. Pour afficher correctement une image dans votre application, vous devez connaître son orientation. Vous devez peut-être corriger cette orientation. Pour certains fichiers .jpg, l’orientation de l’image est contenue dans ses métadonnées Exif (Exchangeable Image File Format).

Pour afficher un cadre autour d’un visage, vous avez besoin des coordonnées du cadre de délimitation du visage. Si la boîte n’est pas orientée correctement, vous devez peut-être ajuster ces coordonnées. Les opérations de détection de visages Image Amazon Rekognition renvoient les coordonnées du cadre de délimitation pour chaque visage détecté, mais elles n’estiment pas les coordonnées des fichiers .jpg sans métadonnées Exif.

Les exemples suivants montrent comment obtenir les coordonnées des cadres de délimitation des visages détectés dans une image.

Servez-vous de cet exemple pour déterminer si vos images sont correctement orientées et si les cadres de délimitation sont affichés à l’emplacement approprié dans votre application.

Comme le code permettant de faire pivoter et d’afficher les images et les cadres de délimitation dépend du langage et de l’environnement utilisés, nous n’expliquons pas comment afficher les images et les cadres de délimitation dans votre code ni comment obtenir les informations d’orientation à partir des métadonnées Exif.

Détermination de l’orientation d’une image

Pour qu’une image s’affiche correctement dans votre application, il peut être nécessaire de la faire pivoter. L’image suivante présente une orientation de 0 degré et est correctement affichée.

Cependant, l'image suivante a faire l'objet d'une rotation de 90 degrés dans le sens inverse des aiguilles d'une montre. Pour l'afficher correctement, vous devez déterminer l'orientation de l'image et utiliser ces informations dans votre code pour la faire pivoter à 0 degré.

Pour certaines images au format .jpg, les informations d'orientation sont contenues dans leurs métadonnées Exif. Si elles sont disponibles, les métadonnées Exif de l'image contiennent l'orientation. Dans les métadonnées Exif, l'orientation de l'image est indiquée dans le champ `orientation`. Même si Image Amazon Rekognition identifie la présence d'informations sur l'orientation de l'image dans les métadonnées Exif, il n'y donne pas accès. Pour accéder aux métadonnées Exif d'une image, vous devez utiliser une bibliothèque tierce ou écrire votre propre code. Pour en savoir plus, consultez Exif Version 2.32.

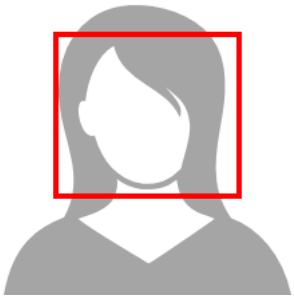
Du moment où vous connaissez l'orientation d'une image, vous pouvez écrire du code pour la faire pivoter et l'afficher correctement.

Affichage de cadres de délimitation

Les opérations Image Amazon Rekognition qui analysent les visages dans une image renvoient également les coordonnées des cadres de délimitation qui entourent les visages. Pour de plus amples informations, veuillez consulter BoundingBox.

Pour afficher un cadre de délimitation autour d'un visage, à l'instar de celui figurant dans l'image suivante, dans votre application, utilisez les coordonnées de ce cadre dans votre code. Les

coordonnées du cadre de délimitation renvoyées par une opération reflètent l'orientation de l'image. Si vous devez faire pivoter l'image pour l'afficher correctement, vous pouvez être amené à traduire les coordonnées du cadre de délimitation.



Affichage des cadres de délimitation en présence d'informations d'orientation dans les métadonnées Exif

Si l'orientation d'une image est incluse dans les métadonnées Exif, les opérations Image Amazon Rekognition produisent les résultats suivants :

- Retour de la valeur null dans le champ de correction de l'orientation dans la réponse de l'opération. Pour faire pivoter l'image, utilisez l'orientation fournie dans les métadonnées Exif dans votre code.
- Retour des coordonnées du cadre de délimitation qui présente déjà une orientation de 0 degré. Pour afficher le cadre de délimitation à l'emplacement approprié, utilisez les coordonnées renvoyées. Vous n'avez pas besoin de les traduire.

Exemple : obtention de l'orientation d'une image et des coordonnées du cadre de délimitation associé

L'exemple suivant montre comment utiliser l'AWS SDK pour obtenir l'orientation d'une image Exif et les coordonnées des cadres de délimitation des célébrités détectées par l'opération `RecognizeCelebrities`.

> **Note**
>
> L'assistance pour estimer l'orientation de l'image à l'aide du champ `OrientationCorrection` a cessé en août 2021. Toutes les valeurs renvoyées pour ce champ inclus dans une réponse d'API seront toujours NULL.

Java

Cet exemple charge une image à partir du système de fichiers local, appelle l'opération `RecognizeCelebrities`, détermine la hauteur et la largeur de l'image et calcule enfin les coordonnées du cadre de délimitation du visage pour l'image ayant fait l'objet d'une rotation. L'exemple ne montre pas comment traiter les informations d'orientation stockées dans les métadonnées Exif.

Dans la fonction `main`, remplacez la valeur de `photo` par le nom et le chemin d'une image stockée localement au format .png ou .jpg.

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

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

public class RotateImage {

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

  String photo = "photo.png";

  //Get Rekognition client
```

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


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

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

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

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

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

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

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

 //Call GetCelebrities

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


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

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

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

            System.out.println();
       }

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

}


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

  float left = 0;
  float top = 0;

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

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

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

  }
}
```

Python

Cet exemple utilise la bibliothèque PIL/Pillow d'images pour obtenir la largeur et la hauteur de l'image. Pour en savoir plus, consultez Pillow. Cet exemple conserve les métadonnées Exif, dont vous pourriez avoir besoin ailleurs dans votre application.

Dans la fonction `main`, remplacez la valeur de `photo` par le nom et le chemin d'une image stockée localement au format .png ou .jpg.

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

import boto3
import io
from PIL import Image
```

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

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


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

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

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

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

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

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

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

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

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

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


def main():
    photo = 'photo'

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


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

Java V2

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

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

Utilisation des opérations d'analyse vidéo stockées

Vidéo Amazon Rekognition est une API que vous pouvez utiliser pour analyser des vidéos. Avec Vidéo Amazon Rekognition, vous pouvez détecter des étiquettes, des visages, des personnes, des célébrités et du contenu pour adulte (suggestif et explicite) dans des vidéos stockées dans un compartiment Amazon Simple Storage Service (Amazon S3). Vous pouvez utiliser Amazon Rekognition Video dans des catégories telles que la sécurité publique. media/entertainment Auparavant, l’analyse de vidéos en vue de la détection d’objets ou de personnes pouvait nécessiter plusieurs heures de visionnage manuel, source d’erreurs potentielles. Vidéo Amazon Rekognition automatise la détection d’éléments et du moment où ceux-ci apparaissent dans une vidéo.

Cette section présente les types d’analyse que Vidéo Amazon Rekognition peut effectuer, une présentation de l’API et des exemples d’utilisation de Vidéo Amazon Rekognition.

Rubriques

- Types d'analyse
- Présentation de l'API Vidéo Amazon Rekognition
- Appeler les opérations de Vidéo Amazon Rekognition
- Configuration de Vidéo Amazon Rekognition
- Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK)
- Analyse d'une vidéo à l'aide du AWS Command Line Interface
- Référence : Notification des résultats d'une analyse vidéo
- Résolution des problèmes liés à Vidéo Amazon Rekognition

Types d'analyse

Vous pouvez utiliser Vidéo Amazon Rekognition pour analyser des vidéos et détecter les informations suivantes :

- Segments vidéo
- Etiquettes
- Contenu pour adulte suggestif et explicite
- Text
- Célébrités
- Visages
- Personnes

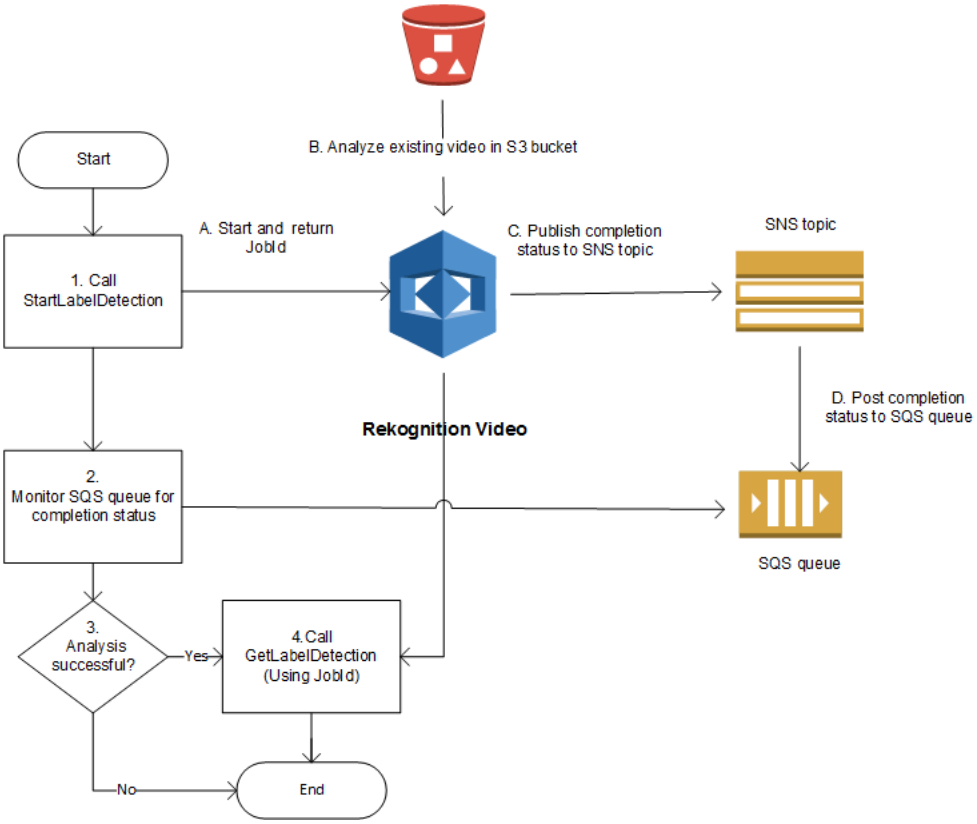
Pour de plus amples informations, veuillez consulter Fonctionnement d'Amazon Rekognition.

Présentation de l'API Vidéo Amazon Rekognition

Vidéo Amazon Rekognition traite une vidéo stockée dans un compartiment Amazon S3. Le schéma de conception est constitué d'un ensemble asynchrone d'opérations. Vous lancez l'analyse vidéo en appelant une opération `Start` telle que StartLabelDetection. L'état d'achèvement de la demande est publié dans une rubrique Amazon Simple Notification Service (Amazon SNS). Pour connaître l'état d'achèvement à partir de la rubrique Amazon SNS, vous pouvez utiliser une file d'attente ou une fonction Amazon Simple Queue Service (Amazon SQS). AWS Lambda Après avoir obtenu le statut

d'achèvement, vous appelez une opération `Get` telle que GetLabelDetection pour obtenir le résultat de la demande.

Le schéma suivant illustre le processus de détection d'étiquettes dans une vidéo stockée dans un compartiment Amazon S3. Dans ce schéma, une file d'attente Amazon SNS obtient le statut d'achèvement à partir de la rubrique SNS. Vous pouvez également utiliser une AWS Lambda fonction.



Le processus est le même pour les autres opérations Vidéo Amazon Rekognition. Le tableau suivant répertorie les opérations `Start` et `Get` pour chacune des opérations Amazon Rekognition hors stockage.

Détection	Opération Start	Opération Get
Segments vidéo	StartSegmentDetection	GetSegmentDetection
Étiquettes	StartLabelDetection	GetLabelDetection
Contenu pour adulte suggestif ou explicite	StartContentModeration	GetContentModeration

Détection	Opération Start	Opération Get
Texte	StartTextDetection	GetTextDetection
Célébrités	StartCelebrityRecognition	GetCelebrityRecognition
Visages	StartFaceDetection	GetFaceDetection
Personnes	StartPersonTracking	GetPersonTracking

Pour les opérations `Get` autres que `GetCelebrityRecognition`, Vidéo Amazon Rekognition renvoie des informations de suivi lorsque des entités sont détectées dans la vidéo en entrée.

Pour de plus amples informations sur l'utilisation de Vidéo Amazon Rekognition, veuillez consulter Appeler les opérations de Vidéo Amazon Rekognition. Pour obtenir un exemple d'analyse vidéo avec Amazon SQS, consultez Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK). Pour AWS CLI des exemples, voirAnalyse d'une vidéo à l'aide du AWS Command Line Interface.

Formats et stockage de vidéos

Les opérations Amazon Rekognition peuvent analyser des vidéos stockées dans des compartiments Amazon S3. Pour obtenir la liste de toutes les limites liées aux opérations d'analyse vidéo, veuillez consulter la rubrique Consignes et quotas.

La vidéo doit être encodée à l'aide du H.264 codec. Les formats de fichier pris en charge sont MPEG-4 et MOV.

Un codec est un logiciel ou un matériel qui compresse les données pour une livraison plus rapide, et décompresse les données reçues dans leur format d'origine. Le H.264 codec est couramment utilisé pour enregistrer, compresser et distribuer du contenu vidéo. Un format de fichier vidéo peut contenir un ou plusieurs codecs. Si votre fichier MOV ou MPEG-4 format vidéo ne fonctionne pas avec Amazon Rekognition Video, vérifiez que le codec utilisé pour encoder la vidéo est bien le cas. H.264

Toute API Vidéo Amazon Rekognition qui analyse les données audio ne prend en charge que les codecs audio AAC.

La taille maximale d'un fichier pour une vidéo stockée est de 10 Go.

Recherche de personnes

Vous pouvez utiliser des métadonnées faciales, stockées dans une collection, pour rechercher des personnes dans une vidéo. Par exemple, vous pouvez rechercher une ou plusieurs personnes en particulier dans une vidéo archivée. Vous stockez des métadonnées faciales à partir d'images source dans une collection en utilisant l'opération IndexFaces. Vous pouvez ensuite utiliser StartFaceSearch pour lancer de manière asynchrone une recherche sur les visages de la collection. Vous utilisez GetFaceSearch pour obtenir les résultats de la recherche. Pour de plus amples informations, veuillez consulter Recherche de visages dans des vidéos stockées. La recherche de personnes est un exemple d'opération Amazon Rekognition basée sur le stockage. Pour de plus amples informations, veuillez consulter Storage-based Opérations d'API.

Vous pouvez aussi rechercher des personnes dans une vidéo en streaming. Pour de plus amples informations, veuillez consulter Utilisation des vidéos streaming.

Appeler les opérations de Vidéo Amazon Rekognition

Vidéo Amazon Rekognition est une API asynchrone que vous pouvez utiliser pour analyser des vidéos stockées dans un compartiment Amazon Simple Storage Service (Amazon Simple Storage Service (Amazon S3). Vous commencez l'analyse d'une vidéo en appelant une opération Amazon Rekognition `Start` Video, telle que. StartPersonTracking Vidéo Amazon Rekognition publie le résultat de la demande d'analyse dans une rubrique Amazon Simple Notification Service (Amazon SNS). Vous pouvez utiliser une file d'attente Amazon Simple Queue Service (Amazon SQS) ou AWS Lambda une fonction pour obtenir l'état d'avancement de la demande d'analyse vidéo à partir de la rubrique Amazon SNS. Enfin, vous obtenez les résultats de la demande d'analyse vidéo en appelant une opération `Get` Amazon Rekognition, telle que. GetPersonTracking

Les informations contenues dans les sections suivantes utilisent des opérations de détection d'étiquette pour montrer comment Vidéo Amazon Rekognition détecte des étiquettes (objets, événements, concepts et activités) dans une vidéo stockée dans un compartiment Amazon S3. La même approche fonctionne pour les autres opérations Amazon Rekognition Video, par exemple, et. StartFaceDetectionStartPersonTracking L'exemple Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK) présente comment analyser une vidéo en utilisant une file d'attente Amazon SQS pour obtenir le statut d'achèvement à partir de la rubrique Amazon SNS. Il est également utilisé comme base pour d'autres exemples Vidéo Amazon Rekognition tels que Tracé du parcours de personnes. Pour AWS CLI des exemples, voirAnalyse d'une vidéo à l'aide du AWS Command Line Interface.

Rubriques

- Démarrage d'une analyse vidéo
- Obtention du statut d'achèvement d'une demande d'analyse Vidéo Amazon Rekognition
- Obtenir les résultats de l'analyse de Vidéo Amazon Rekognition

Démarrage d'une analyse vidéo

Vous lancez une demande de détection d'étiquette Amazon Rekognition Video en appelant. StartLabelDetection L'exemple suivant est une demande JSON transmise par `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"
}
```

Le paramètre d'entrée `Video` contient le nom du fichier vidéo et le compartiment Amazon S3 dont ce fichier peut être extrait. `NotificationChannel` contient l'Amazon Resource Name (ARN) de la rubrique Amazon SNS à laquelle Vidéo Amazon Rekognition adresse une notification lorsque la demande d'analyse vidéo est terminée. La rubrique Amazon SNS doit être située dans la même région AWS que le point de terminaison Vidéo Amazon Rekognition que vous appelez. `NotificationChannel` contient également l'ARN pour un rôle qui permet à Vidéo Amazon Rekognition de publier dans la rubrique Amazon SNS. Vous attribuez des autorisations de publication Amazon Rekognition dans vos rubriques Amazon SNS en créant une fonction du service IAM. Pour de plus amples informations, veuillez consulter Configuration de Vidéo Amazon Rekognition.

Vous pouvez également spécifier un paramètre d'entrée facultatif, `JobTag`, qui vous permet d'identifier la tâche dans le statut d'achèvement publié dans la rubrique Amazon SNS.

Afin d'éviter toute duplication accidentelle des tâches d'analyse, vous pouvez, si vous le souhaitez, fournir un jeton idempotent, `ClientRequestToken`. Si vous indiquez une valeur pour `ClientRequestToken`, l'opération `Start` renvoie le même `JobId` pour plusieurs appels identiques à l'opération de démarrage, par exemple `StartLabelDetection`. Un jeton `ClientRequestToken` a une durée de vie de 7 jours. Au delà de 7 jours, vous pouvez le réutiliser. Si vous réutilisez le jeton pendant sa durée de vie, ce qui suit se produit :

- Si vous réutilisez le jeton avec la même opération `Start` et les mêmes paramètres d'entrée, le même `JobId` est renvoyé. La tâche n'est pas exécutée à nouveau et Vidéo Amazon Rekognition n'envoie pas de statut d'achèvement à la rubrique Amazon SNS enregistrée.
- Si vous réutilisez le jeton avec la même opération `Start` et une modification mineure du paramètre d'entrée, vous obtenez une exception `IdempotentParameterMismatchException` (code de statut HTTP : 400).
- Vous ne devez pas réutiliser un jeton avec des opérations `Start` différentes, car vous obtiendrez des résultats imprévisibles d'Amazon Rekognition.

La réponse à l'opération `StartLabelDetection` est un identifiant de tâche (`JobId`). Utilisez `JobId` pour suivre les demandes et obtenir les résultats d'analyse une fois que Vidéo Amazon Rekognition a publié le statut d'achèvement dans la rubrique Amazon SNS. Par exemple :

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

Si vous lancez un trop grand nombre de tâches simultanément, les appels à `StartLabelDetection` génèrent une `LimitExceededException` (code de statut HTTP : 400) jusqu'à ce que le nombre de tâches exécutées simultanément soit inférieur à la limite de service Amazon Rekognition.

Si vous constatez que les exceptions `LimitExceededException` sont générées lors des pics d'activité, vous devez envisager d'utiliser une file d'attente Amazon SQS afin de gérer les demandes entrantes. Contactez le AWS support si vous constatez que votre nombre moyen de demandes simultanées ne peut pas être géré par une file d'attente Amazon SQS et que vous recevez `LimitExceededException` toujours des exceptions.

Obtention du statut d'achèvement d'une demande d'analyse Vidéo Amazon Rekognition

Vidéo Amazon Rekognition envoie une notification d'achèvement d'analyse à la rubrique Amazon SNS enregistrée. La notification comprend l'identifiant de la tâche et le statut d'achèvement de l'opération dans une chaîne JSON. Une demande d'analyse vidéo réussie a un statut `SUCCEEDED`. Par exemple, le résultat suivant montre la réussite du traitement d'une tâche de détection d'étiquette.

```
{
    "JobId": "270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1nnnnnnnnnnnn",
    "Status": "SUCCEEDED",
    "API": "StartLabelDetection",
    "JobTag": "DetectingLabels",
    "Timestamp": 1510865364756,
    "Video": {
        "S3ObjectName": "video.mp4",
        "S3Bucket": "amzn-s3-demo-bucket"
    }
}
```

Pour de plus amples informations, veuillez consulter Référence : Notification des résultats d'une analyse vidéo.

Pour obtenir les informations de statut publiées dans la rubrique Amazon SNS par Vidéo Amazon Rekognition, utilisez l'une des options suivantes :

- AWS Lambda : vous pouvez abonner une fonction AWS Lambda que vous écrivez à une rubrique Amazon SNS. La fonction est appelée quand Amazon Rekognition informe la rubrique Amazon SNS que la demande est terminée. Utilisez une fonction Lambda si vous souhaitez que le code côté serveur traite les résultats d'une demande d'analyse vidéo. Par exemple, vous pouvez utiliser du code côté serveur pour annoter la vidéo ou créer un rapport sur le contenu de la vidéo avant de renvoyer les informations vers une application cliente. Nous recommandons également un traitement côté serveur pour les vidéos volumineuses, car l'API Amazon Rekognition risque de renvoyer d'importants volumes de données.
- Amazon Simple Queue Service : vous pouvez abonner une file d'attente Amazon SQS à une rubrique Amazon SNS. Vous pouvez ensuite interroger la file d'attente Amazon SQS pour extraire le statut d'achèvement publié par Amazon Rekognition lorsqu'une demande d'analyse vidéo se termine. Pour de plus amples informations, veuillez consulter Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK). Utilisez une file d'attente Amazon SQS

si vous souhaitez appeler des opérations Vidéo Amazon Rekognition uniquement à partir d'une application cliente.

> **Important**
>
> Nous vous déconseillons d'obtenir le statut d'achèvement d'une demande en appelant de manière répétée l'opération `Get` Vidéo Amazon Rekognition. La raison en est que Vidéo Amazon Rekognition limite l'opération `Get` si de trop nombreuses demandes sont lancées. Si vous traitez plusieurs vidéos simultanément, il est plus simple et plus efficace de surveiller la création d'une notification d'achèvement dans une file d'attente SQS que d'interroger Vidéo Amazon Rekognition pour obtenir le statut de chaque vidéo individuellement.

Obtenir les résultats de l'analyse de Vidéo Amazon Rekognition

Pour obtenir les résultats d'une demande d'analyse vidéo, assurez-vous tout d'abord que le statut d'achèvement extrait de la rubrique est Amazon SNS est `SUCCEEDED`. Ensuite, appelez `GetLabelDetection`, qui transmet la valeur `JobId` renvoyée par `StartLabelDetection`. Le format JSON de la demande est similaire à l'exemple suivant :

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

JobId est l'identifiant de l'opération d'analyse vidéo. Dans la mesure où l'analyse vidéo peut générer de grandes quantités de données, utilisez `MaxResults` pour spécifier le nombre maximum de résultats pouvant être renvoyés dans une opération Get. La valeur par défaut du paramètre `MaxResults` est 1000. Si vous spécifiez une valeur supérieure à 1 000, seuls 1 000 résultats sont renvoyés au maximum. Si l'opération ne renvoie pas l'ensemble des résultats, un jeton de pagination pour la page suivante est renvoyé dans la réponse de l'opération. Si vous obtenez un jeton de pagination transmis par une précédente demande Get, utilisez-le avec `NextToken` pour obtenir la page suivante des résultats.

> **Note**
>
> Amazon Rekognition conserve les résultats d'une opération d'analyse vidéo pendant 7 jours. Passé ce délai, vous ne pourrez pas récupérer les résultats de l'analyse.

Le format JSON d'une réponse d'opération `GetLabelDetection` est similaire à l'exemple suivant :

```
{
    "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
    }
}
```

Les opérations `GetLabelDetection` et `GetContentModeration` vous permettent de trier les résultats de l'analyse par horodatage ou par nom d'étiquette. Vous pouvez également agréger les résultats par horodatage ou par segments vidéo.

Vous pouvez trier les résultats en fonction de l'heure de détection (millisecondes à partir du début de la vidéo) ou de l'entité détectée (objet, visage, célébrité, étiquette de modération ou personne), par ordre alphabétique. Pour trier en fonction de l'heure, définissez la valeur du paramètre d'entrée `SortBy` sur `TIMESTAMP`. Si `SortBy` n'est pas spécifié, le tri est, par défaut, effectué en fonction de l'heure. L'exemple précédent est trié en fonction de l'heure. Pour trier en fonction de l'entité, utilisez le paramètre d'entrée `SortBy` avec la valeur appropriée pour l'opération que vous exécutez. Par exemple, pour trier en fonction de l'étiquette détectée dans un appel à `GetLabelDetection`, utilisez la valeur `NAME`.

Pour agréger les résultats par horodatage, définissez la valeur du paramètre `AggregateBy` sur `TIMESTAMPS`. Pour agréger par segment vidéo, définissez la valeur de `AggregateBy` à`SEGMENTS`. Le mode d'agrégation `SEGMENTS` agrègera les étiquettes au fil du temps, tandis que `TIMESTAMPS` donne l'horodatage auquel une étiquette a été détectée, en utilisant un échantillonnage à 2 images par seconde et une sortie par image (Remarque : ce taux d'échantillonnage actuel est sujet à modification, aucune hypothèse ne doit être faite quant au taux d'échantillonnage actuel). Si aucune valeur n'est spécifiée, la méthode d'agrégation par défaut est `TIMESTAMPS`.

Configuration de Vidéo Amazon Rekognition

Pour utiliser l'API Vidéo Amazon Rekognition avec les vidéos stockées, vous devez configurer l'utilisateur et une fonction du service IAM pour l'accès à vos rubriques Amazon SNS. Vous devez également abonner une file d'attente Amazon SQS à vos rubriques Amazon SNS.

> **Note**
>
> Si vous utilisez ces instructions pour configurer l'exemple Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK), vous n'avez pas besoin d'effectuer les étapes 3, 4, 5 et 6. L'exemple inclut le code permettant de créer et de configurer la rubrique Amazon SNS et la file d'attente Amazon SQS.

Les exemples de cette section créent une nouvelle rubrique Amazon SNS à l'aide des instructions permettant à Vidéo Amazon Rekognition d'accéder à plusieurs rubriques. Si vous souhaitez utiliser une rubrique Amazon SNS existante, utilisez Octroi de l'accès à une rubrique Amazon SNS existante pour l'étape 3.

Pour configurer Vidéo Amazon Rekognition

1. Créez un AWS compte pour accéder à Amazon Rekognition Video. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.
2. Installez et configurez le AWS SDK requis. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.
3. Pour exécuter les exemples de code présentés dans ce guide du développeur, assurez-vous que l'utilisateur que vous avez choisi dispose d'un accès programmatique. Pour plus d'informations, consultez Octroi d'un accès par programmation.

   Votre utilisateur doit également disposer au moins des autorisations suivantes :

   - AmazonSQSFullAccess
   - AmazonRekognitionFullAccess
   - AmazonS3FullAccess
   - AmazonSNSFullAccess

   Si vous utilisez IAM Identity Center pour vous authentifier, ajoutez les autorisations à l'ensemble d'autorisations pour votre rôle, sinon ajoutez les autorisations à votre rôle IAM.
4. Créez une rubrique Amazon SNS à l'aide de la console Amazon SNS. Ajoutez le nom du sujet avec AmazonRekognition. Notez l'Amazon Resource Name (ARN) de la rubrique. La rubrique doit se trouver dans la même région que le point de terminaison AWS que vous utilisez.
5. Créez une file d'attente Amazon SQS standard à l'aide de la console Amazon SQS. Notez l'ARN de la file d'attente.
6. Abonnez la file d'attente à la rubrique que vous avez créée à l'étape 3.
7. Autorisez la rubrique Amazon SNS à envoyer des messages à la file d'attente Amazon SQS.
8. Créez une fonction du service IAM pour permettre à Vidéo Amazon Rekognition d'accéder à vos rubriques Amazon SNS. Notez l'Amazon Resource Name (ARN) de la fonction du service. Pour de plus amples informations, veuillez consulter Accorder l'accès à plusieurs rubriques Amazon SNS.
9. Pour garantir la sécurité de votre compte, vous devez limiter l'accès de Rekognition aux seules ressources que vous utilisez. Pour cela, attachez une politique de confiance à votre fonction du service IAM. Pour plus d'informations sur la procédure à utiliser, consultez Cross-service prévention confuse des adjoints.
10. Ajoutez la stratégie en ligne suivante à l'utilisateur que vous avez créé à l'étape 1 :

Attribuez le nom de votre choix à la stratégie en ligne.

11. Si vous utilisez une AWS Key Management Service clé gérée par le client pour chiffrer les vidéos de votre compartiment Amazon S3, ajoutez des autorisations à la clé qui permettent au rôle de service que vous avez créé à l'étape 7 de déchiffrer les vidéos. Au minimum, la fonction du service nécessite une autorisation pour des actions `kms:GenerateDataKey` et `kms:Decrypt`. Par exemple :

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

    Pour plus d'informations, consultez la section Mon compartiment Amazon S3 est crypté par défaut à l'aide d'une clé AWS KMS personnalisée. Comment puis-je autoriser les utilisateurs à effectuer des téléchargements depuis et vers le bucket ? et protection des données à l'aide du Server-Side chiffrement avec des clés KMS stockées dans AWS Key Management Service (SSE-KMS).

12. Vous pouvez désormais exécuter les exemples pour Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK) et Analyse d'une vidéo à l'aide du AWS Command Line Interface.

Accorder l'accès à plusieurs rubriques Amazon SNS

Vous utilisez une fonction du service IAM pour permettre à Vidéo Amazon Rekognition d'accéder aux rubriques Amazon SNS que vous créez. IAM fournit le cas d'utilisation Rekognition pour la création d'une fonction du service Vidéo Amazon Rekognition.

Vous pouvez autoriser Amazon Rekognition Video à accéder à plusieurs rubriques Amazon SNS en utilisant la politique d'autorisation et en faisant précéder `AmazonRekognitionServiceRole` les noms des rubriques par, par exemple,. AmazonRekognition`AmazonRekognitionMyTopicName`

Pour permettre à Vidéo Amazon Rekognition d'accéder à plusieurs rubriques Amazon SNS

1. Créez une fonction du service IAM. Utilisez les informations suivantes pour créer la fonction du service IAM :
   1. Choisissez Rekognition pour le nom du service.
   2. Choisissez Rekognition pour le cas d'utilisation de la fonction de service. Vous devriez voir la politique AmazonRekognitionServiceRoled'autorisation répertoriée. AmazonRekognitionServiceRolepermet à Amazon Rekognition Video d'accéder aux rubriques Amazon SNS préfixées par. AmazonRekognition
   3. Attribuez le nom de votre choix à la fonction du service.
2. Notez l'ARN de la fonction du service. Vous en aurez besoin pour démarrer les opérations d'analyse vidéo.

Octroi de l'accès à une rubrique Amazon SNS existante

Vous pouvez créer une stratégie d'autorisations qui donne à Vidéo Amazon Rekognition l'accès à une rubrique Amazon SNS existante.

Pour donner à Vidéo Amazon Rekognition l'accès à une rubrique Amazon SNS existante

1. Créez une nouvelle stratégie d'autorisations avec l'éditeur de stratégie IAM JSON, et utilisez la stratégie suivante. Remplacez `topicarn` par l'Amazon Resource Name (ARN) de la rubrique Amazon SNS souhaitée.
2. Créez une fonction du service IAM, ou mettez à jour une fonction du service IAM existant. Utilisez les informations suivantes pour créer la fonction du service IAM :
   1. Choisissez Rekognition pour le nom du service.
   2. Choisissez Rekognition pour le cas d'utilisation de la fonction du service.
   3. Attachez la stratégie d'autorisations que vous avez créée à l'étape 1.
3. Notez l'ARN de la fonction du service. Vous en aurez besoin pour démarrer les opérations d'analyse vidéo.

Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK)

Cette procédure vous montre comment détecter des étiquettes dans une vidéo en utilisant les opérations de détection d'étiquette Vidéo Amazon Rekognition, une vidéo stockée dans un compartiment Amazon S3 et une rubrique Amazon SNS. La procédure montre également comment utiliser une file d'attente Amazon SQS pour obtenir le statut d'achèvement à partir de la rubrique Amazon SNS. Pour de plus amples informations, veuillez consulter Appeler les opérations de Vidéo Amazon Rekognition. Vous n'êtes pas limité à l'utilisation d'une file d'attente Amazon SQS. Par exemple, vous pouvez utiliser une AWS Lambda fonction pour obtenir le statut d'achèvement. Pour en savoir plus, consultez Invocation des fonctions Lambda en utilisant des notifications Amazon SNS.

L'exemple de code de cette procédure montre comment effectuer les opérations suivantes :

1. Créer la rubrique Amazon SNS.
2. Créer la file d'attente Amazon SQS.
3. Accorder à Vidéo Amazon Rekognition l'autorisation de publier le statut d'achèvement d'une opération d'analyse vidéo dans la rubrique Amazon SNS.
4. Abonner la file d'attente Amazon SNS à la rubrique Amazon SNS.
5. Lancer la demande d'analyse vidéo en appelant StartLabelDetection.
6. Obtenir le statut d'achèvement à partir de la file d'attente Amazon SQS. L'exemple suit l'identifiant de tâche (`JobId`) renvoyé dans `StartLabelDetection` et obtient uniquement les résultats pour les identifiants de tâche correspondants qui sont lus à partir du statut d'achèvement. Ceci doit être pris en considération si d'autres applications utilisent la même file d'attente et la même rubrique. Par souci de simplicité, l'exemple supprime les tâches qui ne correspondent pas. Envisagez de les ajouter à une file d'attente de lettres mortes Amazon SQS pour un examen plus approfondi.
7. Obtenir et afficher les résultats de l'analyse vidéo en appelant GetLabelDetection.

Conditions préalables

L'exemple de code pour cette procédure est fourni en Java et Python. Le AWS SDK approprié doit être installé. Pour de plus amples informations, veuillez consulter Premiers pas avec Amazon Rekognition. Le compte AWS que vous utilisez doit avoir les autorisations d'accès à l'API Amazon Rekognition. Pour plus d'informations, consultez Actions définies par Amazon Rekognition.

Pour détecter des étiquettes dans une vidéo

1. Configurez l'accès des utilisateurs à Vidéo Amazon Rekognition et configurez l'accès de Vidéo Amazon Rekognition à Amazon SNS. Pour de plus amples informations, veuillez consulter Configuration de Vidéo Amazon Rekognition. Vous n'avez pas besoin d'effectuer les étapes 3, 4, 5 et 6, car l'exemple de code crée et configure la rubrique Amazon SNS et la file d'attente Amazon SQS.
2. Chargez un fichier MOV ou un fichier vidéo MPEG-4 formaté dans un compartiment Amazon S3. À des fins de test, téléchargez une vidéo de 30 secondes au maximum.

   Pour en savoir plus, consultez Chargement d'objets dans Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

3. Utilisez les exemples de code suivants pour détecter des étiquettes dans une vidéo.

   Java

   Dans la fonction `main` :

   - Remplacez `roleArn` par l'ARN de la fonction du service IAM que vous avez créé au cours de l'étape 7 de Pour configurer Vidéo Amazon Rekognition.
   - Remplacez les valeurs de `amzn-s3-demo-bucket` et `video` par le nom du compartiment et le nom du fichier vidéo que vous avez spécifiés à l'étape 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

Dans la fonction `main` :

- Remplacez `roleArn` par l'ARN de la fonction du service IAM que vous avez créé au cours de l'étape 7 de Pour configurer Vidéo Amazon Rekognition.
- Remplacez les valeurs de `amzn-s3-demo-bucket` et `video` par le nom du compartiment et le nom du fichier vidéo que vous avez spécifiés à l'étape 2.
- Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

- Vous pouvez également inclure des critères de filtrage dans le paramètre des paramètres. Par exemple, vous pouvez utiliser un `LabelsInclusionFilter` ou un `LabelsExclusionFilter` à côté d'une liste de valeurs souhaitées. Dans le code ci-dessous, vous pouvez supprimer les commentaires de la section `Features` et `Settings` et fournir vos propres valeurs afin de limiter les résultats renvoyés aux seules étiquettes qui vous intéressent.
- Dans l'appel à `GetLabelDetection`, vous pouvez fournir des valeurs pour les arguments `SortBy` et `AggregateBy`. Pour trier en fonction de l'heure, définissez la valeur du paramètre d'entrée `SortBy` sur `TIMESTAMP`. Pour trier en fonction de l'entité, utilisez le paramètre d'entrée `SortBy` avec la valeur appropriée pour l'opération que vous exécutez. Pour agréger les résultats par horodatage, définissez la valeur du paramètre `AggregateBy` sur `TIMESTAMPS`. Pour agréger par segment vidéo, utilisez `SEGMENTS`.

Dans l'exemple de code, placez la politique IAM suivante entre guillemets sur la ligne. `policy = """""".format(sqsQueueArn, self.snsTopicArn)`

```
## Copyright 2018 Amazon.com, Inc. or its 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
        # Insert the policy from the documentation from
        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

Dans l'exemple de code suivant :

- Remplacez la valeur de `REGION` par le nom de la région d'exploitation de votre compte.
- Remplacez la valeur de `amzn-s3-demo-bucket` par le nom du compartiment Amazon S3 qui contient votre fichier vidéo.
- Remplacez la valeur de `videoName` par le nom du fichier vidéo dans votre compartiment Amazon S3.
- Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.
- Remplacez `roleArn` par l'ARN de la fonction du service IAM que vous avez créé au cours de l'étape 7 de Pour configurer Vidéo Amazon Rekognition.

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

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

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

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

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

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

```
      Version: "2012-10-17",&TCX5-2025-waiver;
      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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
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. Créez et exécutez le code. L'opération peut prendre un certain temps pour s'exécuter. Lorsqu'elle est terminée, une liste des étiquettes détectées dans la vidéo s'affiche. Pour de plus amples informations, veuillez consulter Détection des étiquettes dans une vidéo.

Analyse d'une vidéo à l'aide du AWS Command Line Interface

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour appeler les opérations Amazon Rekognition Video. Le modèle de conception est identique à celui de l'utilisation de l'API Amazon Rekognition Video avec le AWS SDK pour Java ou d'autres SDK AWS. Pour de plus amples informations, veuillez consulter Présentation de l'API Vidéo Amazon Rekognition. Les procédures suivantes montrent comment utiliser le AWS CLI pour détecter des étiquettes dans une vidéo.

Vous commencez à détecter des étiquettes dans une vidéo en appelant `start-label-detection`. Lorsque Amazon Rekognition a terminé l'analyse de la vidéo, le statut d'achèvement est envoyé à la rubrique Amazon SNS spécifiée dans le paramètre `--notification-channel` de `start-label-detection`. Vous pouvez obtenir le statut d'achèvement en abonnant une file d'attente Amazon Simple Queue Service (Amazon SQS) à la rubrique Amazon SNS. Vous pouvez ensuite interroger receive-message pour obtenir le statut d'achèvement à partir de la file d'attente Amazon SQS.

Lorsque vous appelez`StartLabelDetection`, vous pouvez filtrer vos résultats en fournissant des arguments de filtration aux `LabelsInclusionFilter` and/or `LabelsExclusionFilter` arguments. Pour plus d'informations, consultez Détection des étiquettes dans une vidéo.

La notification de statut d'achèvement est une structure JSON dans la réponse `receive-message`. Vous devez extraire le code JSON de la réponse. Pour plus d'informations sur le code JSON du statut d'achèvement, consultez Référence : Notification des résultats d'une analyse vidéo. Si la valeur du champ `Status` du code JSON du statut d'achèvement est `SUCCEEDED`, vous pouvez obtenir les résultats de la demande d'analyse vidéo en appelant `get-label-detection`. Lorsque vous appelez `GetLabelDetection`, vous pouvez trier et agréger les résultats renvoyés à l'aide `SortBy` et `AggregateBy`.

Les procédures suivantes n'incluent pas le code requis pour interroger la file d'attente Amazon SQS. Elles n'incluent également pas le code requis pour analyser le code JSON renvoyé à partir de la file d'attente Amazon SQS. Pour obtenir un exemple en Java, consultez Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK).

Conditions préalables

Pour exécuter cette procédure, vous devez avoir AWS CLI installé le. Pour de plus amples informations, veuillez consulter Premiers pas avec Amazon Rekognition. Le compte AWS que vous utilisez doit avoir les autorisations d'accès à l'API Amazon Rekognition. Pour plus d'informations, Actions définies par Amazon Rekognition.

Pour configurer Vidéo Amazon Rekognition et télécharger une vidéo

1. Configurez l'accès des utilisateurs à Vidéo Amazon Rekognition et configurez l'accès de Vidéo Amazon Rekognition à Amazon SNS. Pour de plus amples informations, veuillez consulter Configuration de Vidéo Amazon Rekognition.
2. Téléchargez un fichier MOV ou un fichier vidéo MPEG-4 formaté dans votre compartiment S3. Pendant le développement et les tests, nous suggérons d'utiliser des vidéos courtes de moins de 30 secondes.

   Pour en savoir plus, consultez Chargement d'objets dans Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

Pour détecter des étiquettes dans une vidéo

1. Exécutez la AWS CLI commande suivante pour commencer à détecter les étiquettes dans une vidéo.

   ```
   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"]}}
   ```

   Mettez à jour les valeurs suivantes :

   - Remplacez `amzn-s3-demo-bucket` et `videofile` par le nom du compartiment Amazon S3 et le nom du fichier que vous avez spécifiés à l'étape 2.
   - Remplacez `us-east-1` par la région AWS que vous utilisez.
   - Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

- Remplacez `TopicARN` par l'ARN de la rubrique Amazon SNS que vous avez créée à l'étape 3 de Configuration de Vidéo Amazon Rekognition.
- Remplacez `RoleARN` par l'ARN de la fonction du service que vous avez créé à l'étape 7 de Configuration de Vidéo Amazon Rekognition.
- Si nécessaire, vous pouvez spécifier le `endpoint-url`. La CLI AWS doit déterminer automatiquement l'URL du point de terminaison approprié en fonction de la région fournie. Toutefois, si vous utilisez un point de terminaison de votre VPC privé, il peut être nécessaire de spécifier le `endpoint-url`. La ressource AWS Service Endpoints répertorie la syntaxe permettant de spécifier les URL des points de terminaison ainsi que les noms et codes de chaque région.
- Vous pouvez également inclure des critères de filtrage dans le réglage des paramètres. Par exemple, vous pouvez utiliser un `LabelsInclusionFilter` ou un `LabelsExclusionFilter` à côté d'une liste de valeurs souhaitées.

Si vous accédez à la CLI sur un périphérique Windows, utilisez des guillemets doubles au lieu de guillemets simples et évitez les guillemets doubles internes par une barre oblique inverse (c'est-à-dire \) pour corriger les erreurs d'analyse que vous pourriez rencontrer. Pour un exemple, voir ci-dessous :

```
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. Notez la valeur de `JobId` dans la réponse. La réponse est semblable à l'exemple JSON suivant.

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

3. Écrivez le code pour interroger la file d'attente Amazon SQS sur le statut d'achèvement JSON (à l'aide de receive-message).
4. Écrivez le code pour extraire le champ `Status` du code JSON de statut d'achèvement.

5. Si la valeur de `Status` est`SUCCEEDED`, exécutez la AWS CLI commande suivante pour afficher les résultats de détection des étiquettes.

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

Mettez à jour les valeurs suivantes :

- Remplacez `JobId` par l'identifiant de tâche que vous avez noté à l'étape 2.
- Remplacez `Endpoint` et `us-east-1` par le point de terminaison et la région AWS que vous utilisez.

Les résultats sont semblables à l'exemple JSON suivant :

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

Référence : Notification des résultats d'une analyse vidéo

Amazon Rekognition publie les résultats d'une demande d'analyse Vidéo Amazon Rekognition, y compris l'état d'achèvement, dans une rubrique Amazon Simple Notification Service (Amazon SNS). Pour recevoir la notification d'une rubrique Amazon SNS, utilisez une file d'attente ou une AWS Lambda fonction Amazon Simple Queue Service. Pour de plus amples informations, veuillez consulter the section called "Appeler les opérations de Vidéo Amazon Rekognition". Pour obtenir un exemple, consultez Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK).

La charge utile est au format JSON suivant :

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

Nom	Description
JobId	Identifiant de tâche. Correspond à l'identifiant de tâche renvoyé par une opération `Start`, par exemple StartPersonTracking.
Statut	Statut de la tâche. Les valeurs valides sont SUCCEEDED, FAILED ou ERROR.
API	Opération Vidéo Amazon Rekognition utilisée pour analyser la vidéo d'entrée.

Nom	Description
JobTag	Identifiant de la tâche. Vous spécifiez `JobTag` dans un appel à une opération Start, par exemple StartLabelDetection.
Horodatage	L'horodatage Unix du moment où la tâche s'est terminée.
Vidéo	Détails sur la vidéo qui a été traitée. Comprend le nom du fichier et le compartiment Amazon S3 dans lequel le fichier est stocké.

L'exemple suivant montre une notification réussie envoyée à une rubrique 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"
  }
}
```

Résolution des problèmes liés à Vidéo Amazon Rekognition

La section suivante fournit des informations de dépannage relatives à l'utilisation de Vidéo Amazon Rekognition et des vidéos stockées.

Je ne reçois jamais le statut d'achèvement qui est envoyé à la rubrique Amazon SNS

Vidéo Amazon Rekognition publie les informations sur le statut dans une rubrique Amazon SNS à la fin de l'analyse d'une vidéo. En général, vous obtenez le message relatif au statut d'achèvement en vous abonnant à la rubrique dotée d'une fonction de file d'attente Amazon SQS, ou d'une fonction Lambda. Pour tenter de résoudre votre problème, abonnez-vous à la rubrique Amazon SNS par e-

mail afin de recevoir les messages qui sont envoyés à votre rubrique Amazon SNS dans votre boîte de réception. Pour plus d'informations, consultez Abonnement à une rubrique Amazon SNS.

Si vous ne recevez pas le message dans votre application, prenez en considération les éléments suivants :

- Vérifiez que l'analyse est terminée. Vérifiez la valeur `JobStatus` dans la réponse de l'opération Get (`GetLabelDetection`, par exemple). Si la valeur est `IN_PROGRESS`, l'analyse n'est pas terminée, et le statut d'achèvement n'a pas encore été publié dans la rubrique Amazon SNS.
- Vérifiez que vous disposez d'une fonction du service IAM autorisant Vidéo Amazon Rekognition à publier dans vos rubriques Amazon SNS. Pour de plus amples informations, veuillez consulter Configuration de Vidéo Amazon Rekognition.
- Vérifiez que la fonction du service IAM que vous utilisez peut publier sur la rubrique Amazon SNS à l'aide des informations d'identification du rôle, et que les autorisations de votre fonction du service sont correctement limitées aux ressources que vous utilisez. Pour ce faire, procédez comme suit :
  - Récupérez l'Amazon Resource Name (ARN) d'utilisateur :

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

  - Ajoutez une relation d'approbation au rôle d'utilisateur ARN. Pour plus d'informations, consultez Modification d'un rôle. L'exemple de politique de confiance suivant spécifie les informations d'identification du rôle de l'utilisateur et limite les autorisations de la fonction du service aux seules ressources que vous utilisez (pour plus d'informations sur la limitation sécurisée de l'étendue des autorisations d'une fonction du service, voir Cross-service prévention confuse des adjoints) :
  - Prenez le rôle : `aws sts assume-role --role-arn arn:Role ARN --role-session-name SessionName --profile RekognitionUser`
  - Publiez dans une rubrique Amazon SNS :`aws sns publish --topic-arn arn:Topic ARN --message "Hello World!" --region us-east-1 --profile RekognitionUser`

  Si la commande AWS CLI fonctionne, vous recevez le message (dans votre boîte de réception, si vous vous êtes abonné au sujet par e-mail). Si vous ne recevez pas le message :

  - Vérifiez que vous avez correctement configuré Vidéo Amazon Rekognition. Pour de plus amples informations, veuillez consulter Configuration de Vidéo Amazon Rekognition.
  - Consultez les autres conseils de dépannage pour cette question.

- Vérifiez que vous utilisez la rubrique Amazon SNS adéquate :
  - Si vous utilisez une fonction du service IAM pour permettre à Vidéo Amazon Rekognition d'accéder à une seule rubrique Amazon SNS, vérifiez que vous avez accordé les autorisations pour la rubrique Amazon SNS adéquate. Pour de plus amples informations, veuillez consulter Octroi de l'accès à une rubrique Amazon SNS existante.
  - Si vous utilisez un rôle de service IAM pour donner à Amazon Rekognition Video l'accès à plusieurs rubriques SNS, vérifiez que vous utilisez la bonne rubrique et que le nom de la rubrique est précédé d'un. AmazonRekognition Pour de plus amples informations, veuillez consulter Accorder l'accès à plusieurs rubriques Amazon SNS.
  - Si vous utilisez une AWS Lambda fonction, vérifiez que votre fonction Lambda est abonnée à la rubrique Amazon SNS appropriée. Pour de plus amples informations, consultez Diffusion en éventail vers fonctions Lambda.
- Si vous abonnez une file d'attente Amazon SQS à votre rubrique Amazon SNS, vérifiez que votre rubrique Amazon SNS est autorisée à envoyer des messages à la file d'attente Amazon SQS. Pour en savoir plus, consultez Donner l'autorisation d'envoi de messages à la rubrique Amazon SNS pour envoyer des messages à la file d'attente Amazon SNS.

J'ai besoin d'une aide supplémentaire pour résoudre des problèmes liés à la rubrique Amazon SNS

Vous pouvez utiliser AWS X-Ray Amazon SNS pour suivre et analyser les messages qui transitent par votre application. Pour plus d'informations, consultez Amazon SNS et. AWS X-Ray

Pour obtenir de l'aide supplémentaire, vous pouvez publier votre question sur le forum Amazon Rekognition ou envisager de vous inscrire au support technique AWS.

Utilisation des vidéos streaming

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vous pouvez utiliser Vidéo Amazon Rekognition pour détecter et reconnaître des visages dans des vidéos diffusées en streaming. Vidéo Amazon Rekognition utilise Amazon Kinesis Video Streams pour recevoir et traiter un flux vidéo. Vous créez un processeur de flux avec des paramètres indiquant ce que le processeur de flux doit détecter dans le flux vidéo. Rekognition envoie les résultats de détection d'étiquettes issus d'événements vidéo diffusés en continu sous forme de notifications Amazon SNS et Amazon S3. Rekognition génère les résultats de recherche de visages dans un flux de données Kinesis.

Les processeurs de flux de recherche de visages utilisent `FaceSearchSettings` pour rechercher des visages dans une collection. Pour plus d'informations sur la mise en œuvre de processeurs de flux de recherche de visages pour analyser les visages dans le streaming vidéo, consultez the section called "Recherche de visages dans une collection en vidéo streaming".

Les processeurs de flux de détection d'étiquettes utilisent `ConnectedHomeSettings` pour rechercher des personnes, des packages et des animaux de compagnie dans des événements vidéo en streaming. Pour plus d'informations sur la mise en œuvre des processeurs de flux de détection d'étiquettes, consultez the section called "Détection d'étiquettes dans des événements vidéo en streaming".

Présentation des opérations du processeur de flux Vidéo Amazon Rekognition

Pour démarrer l'analyse d'une vidéo en streaming, lancez un processeur de flux Vidéo Amazon Rekognition et une vidéo en streaming dans Vidéo Amazon Rekognition. Un processeur de flux Vidéo Amazon Rekognition vous permet de démarrer, d'arrêter et de gérer des processeurs de flux. Vous créez un processeur de flux en appelant CreateStreamProcessor. Les paramètres de demande pour la création d'un processeur de flux de recherche de visage incluent Amazon Resource Names (ARN) pour le flux vidéo Kinesis, le flux de données Kinesis et l'identifiant de la collection utilisée pour identifier les visages dans la vidéo en streaming. Les paramètres de demande pour la création d'un processeur de flux de surveillance de la sécurité incluent Amazon Resource Names (ARN) pour le flux vidéo Kinesis et le sujet Amazon SNS, les types d'objets que vous souhaitez détecter dans le flux vidéo, et les informations relatives à un compartiment Amazon S3 pour les résultats de sortie. Ils comprennent également le nom que vous définissez pour le processeur de flux.

Vous commencez le traitement d'une vidéo en appelant l'opération StartStreamProcessor. Pour obtenir des informations sur l'état d'un processeur de flux, appelez DescribeStreamProcessor. Les autres opérations que vous pouvez effectuer consistent TagResourceà étiqueter un processeur de flux et DeleteStreamProcessorà supprimer un processeur de flux. Si vous utilisez un processeur

de flux de recherche faciale, vous pouvez également l'utiliser StopStreamProcessorpour arrêter un processeur de flux. Pour obtenir une liste des processeurs de flux de votre compte, appelez ListStreamProcessors.

Lorsque le processeur de flux commence à s'exécuter, vous diffusez la vidéo dans Vidéo Amazon Rekognition via le flux vidéo Kinesis que vous avez spécifié dans `CreateStreamProcessor`. Vous pouvez utiliser l'opération du PutMediaSDK Kinesis Video Streams pour diffuser de la vidéo dans le flux vidéo Kinesis. Pour un exemple, voir Exemple PutMedia d'API.

Pour plus d'informations sur la manière dont votre application peut utiliser les résultats d'analyse Vidéo Amazon Rekognition à partir d'un processeur de flux de recherche faciale, consultez Lecture des résultats d'analyse de vidéo en streaming.

Ajouter des balises à un nouveau processeur de flux

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vous pouvez identifier, organiser, rechercher et filtrer les processeurs de flux Amazon Rekognition à l'aide de balises. Chaque balise est une étiquette composée d'une clé définie par l'utilisateur et d'une valeur.

Vous pouvez ajouter des balises à un processeur de flux lorsque vous le créez à l'aide de l'opération `CreateStreamProcessor`. Spécifiez une ou plusieurs balises dans le paramètre d'entrée du tableau `Tags`. L'exemple suivant est un exemple JSON de requête `CreateStreamProcessor` avec balises.

```
{
       "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"
        }
}
```

Ajouter des balises à un processeur de flux existant

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vous pouvez identifier, organiser, rechercher et filtrer les processeurs de flux Amazon Rekognition à l'aide de balises. Chaque balise est une étiquette composée d'une clé définie par l'utilisateur et d'une valeur.

Pour ajouter une ou plusieurs balises à un processeur de flux existant, utilisez l'opération `TagResource`. Spécifiez l'Amazon Resource Name (ARN) (`ResourceArn`) du processeur de flux et les balises (`Tags`) que vous voulez ajouter. L'exemple suivant vous montre comment ajouter deux balises.

```
aws rekognition tag-resource --resource-arn resource-arn \
```

```
                --tags '{"key1":"value1","key2":"value2"}'
```

> **Note**
>
> Si vous ne connaissez pas le nom de ressource Amazon du processeur de flux, vous pouvez utiliser l'opération `DescribeStreamProcessor`.

Répertorier les balises dans un processeur de flux

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vous pouvez identifier, organiser, rechercher et filtrer les processeurs de flux Amazon Rekognition à l'aide de balises. Chaque balise est une étiquette composée d'une clé définie par l'utilisateur et d'une valeur.

Pour répertorier les balises associées à un processeur de flux, utilisez l'opération `ListTagsForResource` et spécifiez l'ARN du processeur de flux (`ResourceArn`). La réponse est une carte des clés de balise et des valeurs attachées au processeur de flux spécifié.

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

La sortie affiche une liste de balises attachées au processeur de flux :

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

```
    }
}
```

Supprimer des balises d'un processeur de flux

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vous pouvez identifier, organiser, rechercher et filtrer les processeurs de flux Amazon Rekognition à l'aide de balises. Chaque balise est une étiquette composée d'une clé définie par l'utilisateur et d'une valeur.

Pour supprimer une ou plusieurs balises d'un processeur de flux, utilisez l'opération `UntagResource`. Spécifiez l'ARN du modèle (`ResourceArn`) et les clés de balise (`Tag-Keys`) que vous souhaitez supprimer.

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

Vous pouvez aussi spécifier les clés de balise dans ce format :

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

Gestion des erreurs

Cette section décrit les erreurs d'exécution et la façon de les traiter. Elle décrit également les messages d'erreur et les codes spécifiques d'Amazon Rekognition.

Rubriques

- Composants erreur

- Codes et messages d'erreur
- Gestion des erreurs dans votre application

Composants erreur

Lorsque votre programme envoie une demande, Amazon Rekognition tente de la traiter. Si la demande aboutit, Amazon Rekognition renvoie un code d'état HTTP de succès (`200 OK`), ainsi que les résultats de l'opération demandée.

Si la demande échoue, Amazon Rekognition retourne une erreur. Chaque erreur possède trois composants :

- Un code d'état HTTP (`400`, par exemple).
- Un nom d'exception (`InvalidS3ObjectException`, par exemple).
- Un message d'erreur (`Unable to get object metadata from S3. Check object key, region and/or access permissions.`, par exemple).

Les AWS SDK prennent soin de répercuter les erreurs dans votre application, afin que vous puissiez prendre les mesures appropriées. Par exemple, dans un programme Java, vous pouvez écrire une logique `try-catch` de façon à gérer un `ResourceNotFoundException`.

Si vous n'utilisez pas un kit AWS SDK, vous devez analyser le contenu de la réponse de bas niveau à partir d'Amazon Rekognition. Voici un exemple de réponse :

```
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."}
```

Codes et messages d'erreur

Voici une liste des exceptions retournées par Amazon Rekognition et regroupées par code d'état HTTP. Si OK to retry? a pour réponse Yes, vous pouvez soumettre à nouveau la même demande. Si

OK to retry? a pour réponse No, vous devez résoudre le problème côté client avant de soumettre une nouvelle demande.

Code d’état HTTP 400

Un code de statut HTTP `400` indique un problème avec votre demande. Parmi les exemples de problème, on peut citer l’échec d’authentification, des paramètres obligatoires manquants ou le dépassement de débit provisionné d’une opération. Vous devez corriger le problème dans votre application avant de soumettre à nouveau la demande.

AccessDeniedException

Message : Une erreur s'est produite (AccessDeniedException) lors de l'appel de l'opération : L'utilisateur : n'est pas autorisé à effectuer : sur la ressource :. <Operation><User ARN><Operation><Resource ARN>

Vous n’êtes pas autorisé à effectuer l’action. Utilisez l’Amazon Resource Name (ARN) d’un utilisateur ou d’un rôle IAM autorisé pour effectuer l’opération.

OK pour réessayer ? Non

GroupFacesInProgressException

Message : Impossible de planifier la GroupFaces tâche. There is an existing group faces job for this collection.

Réessayer l’opération lorsque la tâche existante est terminée.

OK pour réessayer ? Non

IdempotentParameterMismatchException

Message : Le ClientRequestToken : que vous avez fourni est déjà utilisé. <Token>

Un paramètre ClientRequestToken d'entrée a été réutilisé avec une opération, mais au moins l'un des autres paramètres d'entrée est différent de l'appel précédent à l'opération.

OK pour réessayer ? Non

ImageTooLargeException

Message : Image size is too large.

La taille de l'image d'entrée dépasse la limite autorisée. Si vous appelez DetectProtectiveEquipment, la taille ou la résolution de l'image dépasse la limite autorisée. Pour de plus amples informations, veuillez consulter Directives et quotas dans Amazon Rekognition.

OK pour réessayer ? Non

InvalidImageFormatException

Message : Request has invalid image format.

Le format d'image fourni n'est pas pris en charge. Utilisez un format d'image pris en charge (.JPEG et .PNG). Pour de plus amples informations, veuillez consulter Directives et quotas dans Amazon Rekognition.

OK pour réessayer ? Non

InvalidPaginationTokenException

Messages

- Jeton non valide
- Jeton de pagination non valide

Le jeton de pagination inclus dans la demande n'est pas valide. Le jeton peut avoir expiré.

OK pour réessayer ? Non

InvalidParameterException

Message : Request has invalid parameters.

Un paramètre d'entrée a enfreint une contrainte. Validez vos paramètres avant d'appeler à nouveau l'opération d'API.

OK pour réessayer ? Non

InvalidS3ObjectException

Messages :

- Request has invalid S3 object.
- Unable to get object metadata from S3. Vérifiez la clé de l'objet et les autorisations and/or d'accès à la région.

Amazon Rekognition n’est pas en mesure d’accéder à l’objet S3 spécifié dans la demande. Pour plus d’informations, consultez les informations sur la gestion des accès à AWS S3. Pour plus d’informations sur le dépannage, consultez Dépannage Amazon S3.

OK pour réessayer ? Non

LimitExceededException

Messages :

- Stream processor limit exceeded for account, limit - <Current Limit>.
- <Number of Open Jobs> open Jobs for User <User ARN> Maximum limit: <Maximum Limit>

La limite de service Amazon Rekognition a été dépassée. Par exemple, si vous lancez un trop grand nombre de tâches Vidéo Amazon Rekognition simultanément, les appels pour démarrer les opérations, tels que `StartLabelDetection` déclenchent une exception `LimitExceededException` (code de statut HTTP : 400) jusqu’à ce que le nombre de tâches exécutées simultanément soit inférieur à la limite de service Amazon Rekognition.

OK pour réessayer ? Non

ProvisionedThroughputExceededException

Messages :

- Provisioned Rate exceeded.
- S3 download limit exceeded.

Le nombre de demandes dépasse votre limite de débit. Pour de plus amples informations, veuillez consulter limites de service Amazon Rekognition.

Pour demander une augmentation de la limite, suivez les instructions à l’adressethe section called “Créez un dossier pour modifier les quotas TPS”.

OK pour réessayer ? Oui

ResourceAlreadyExistsException

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

Une collection avec l'ID spécifié existe déjà.

OK pour réessayer ? Non

ResourceInUseException

Messages :

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

Réessayez lorsque la ressource est disponible.

OK pour réessayer ? Non

ResourceNotFoundException

Message : Plusieurs messages en fonction de l'appel d'API.

La ressource spécifiée n'existe pas.

OK pour réessayer ? Non

ThrottlingException

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

Votre taux de demande d'augmentation est trop rapide. Diminuez le débit de vos demandes et augmentez-les progressivement. Nous vous recommandons de faire un retour en arrière exponentiel et de réessayer. Par défaut, les AWS SDK utilisent une logique de nouvelle tentative automatique et un retard exponentiel. Pour plus d'informations, consultez Nouvelles tentatives après erreur et interruptions exponentiellesdans AWS et Backoff exponentiel et Jitter.

OK pour réessayer ? Oui

VideoTooLargeException

Message : Video size in bytes: <Video Size> is more than the maximum limit of: <Max Size> bytes.

La taille ou la durée de fichier du support fourni est trop volumineuse. Pour de plus amples informations, veuillez consulter Directives et quotas dans Amazon Rekognition.

OK pour réessayer ? Non

Code d'état HTTP 5xx

Un code d'état HTTP `5xx` indique un problème qui doit être résolu par AWS. Il s'agit peut être d'une erreur temporaire. Si c'est le cas, vous pouvez relancer la demande jusqu'à ce qu'elle aboutisse. Sinon, accédez au AWS Service Health Dashboard pour voir si le service présente des problèmes de fonctionnement.

InternalServerError (HTTP 500)

Message : Internal server error

Amazon Rekognition a rencontré un problème de service. Renouvelez votre appel. Faites un retour en arrière exponentiel et réessayez. Par défaut, les AWS SDK utilisent une logique de nouvelle tentative et une interruption exponentielle automatiques. Pour plus d'informations, consultez Nouvelles tentatives après erreur et interruptions exponentiellesdans AWS et Backoff exponentiel et Jitter.

OK pour réessayer ? Oui

ThrottlingException (HTTP 500)

Message : Service Unavailable

Amazon Rekognition est temporairement dans l'impossibilité de traiter la demande. Renouvelez votre appel. Nous vous recommandons de faire un retour en arrière exponentiel et de réessayer. Par défaut, les AWS SDK utilisent une logique de nouvelle tentative automatique et un retard exponentiel. Pour plus d'informations, consultez Nouvelles tentatives après erreur et interruptions exponentiellesdans AWS et Backoff exponentiel et Jitter.

OK pour réessayer ? Oui

Gestion des erreurs dans votre application

Pour que votre application fonctionne correctement, vous devez ajouter une logique destinée à capturer les erreurs et à y répondre. Les approches classiques incluent l'utilisation de blocs `try-catch` ou d'instructions `if-then`.

Les AWS SDK effectuent leurs propres tentatives et contrôles d'erreur. Si vous rencontrez une erreur tout en utilisant l'un des AWS SDK, le code d'erreur et la description peuvent vous aider à la résoudre.

Vous devez également voir un `Request ID` dans la réponse. L'`Request ID` peut être utile si vous devez utiliser AWS Support pour diagnostiquer un problème.

L'extrait de code Java suivant tente de détecter des objets dans une image et exécute une gestion rudimentaire des erreurs. (Dans ce cas, il informe l'utilisateur que la demande a échoué.)

```
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());
}
```

Dans cet extrait de code, la construction `try-catch` gère deux types différents d'exceptions :

- `AmazonRekognitionException` : cette exception se produit si la demande du client a été correctement transmise à Amazon Rekognition, mais qu'Amazon Rekognition n'a pas pu la traiter et a renvoyé une réponse d'erreur à la place.
- `AmazonClientException` : cette exception se produit si le client n'a pas pu obtenir de réponse d'un service, ou si le client n'a pas pu analyser la réponse d'un service.

Utilisation d'Amazon Rekognition en tant que service autorisé FedRAMP

Le programme de conformité AWS FedRAMP inclut Amazon Rekognition en tant que service. FedRAMP-authorized Si vous êtes un client fédéral ou commercial, vous pouvez utiliser le service pour traiter et stocker des charges de travail sensibles dans les régions de l' AWS est et de l'ouest des États-Unis, avec des données pouvant atteindre un niveau d'impact modéré. Vous pouvez utiliser le service pour des charges de travail sensibles dans la limite d'autorisation de la région AWS GovCloud (États-Unis), avec des données allant jusqu'au niveau d'impact élevé. Pour de plus amples informations sur la conformité FedRAMP, veuillez consulter ConformitéAWS FedRAMP.

Pour être conforme à FedRAMP, vous pouvez utiliser un point de terminaison FIPS (Federal Information Processing Standard). Cela vous donne accès aux modules cryptographiques validés FIPS 140-2 lorsque vous travaillez avec des informations sensibles. Pour plus d'informations sur les points de terminaison FIPS, consultez Vue d'ensemble FIPS 140-2.

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) ou l'un des kits SDK AWS pour spécifier le point de terminaison utilisé par Amazon Rekognition.

Pour connaître les points de terminaison pouvant être utilisés avec Amazon Rekognition, consultez Régions et points de terminaison Amazon Rekognition.

Voici des exemples tirés de la rubrique Dresser une liste des collections dans le Guide du développeur Amazon Rekognition. Ils sont modifiés pour spécifier la région et le point de terminaison FIPS via lequel Amazon Rekognition est accessible.

Java

Pour Java, utilisez la méthode `withEndpointConfiguration` lorsque vous construisez le client Amazon Rekognition. Cet exemple montre les collections dont vous disposez qui utilisent le point de terminaison FIPS dans la région USA Est (N.Virginia) :

```
//Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-
rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;

import java.util.List;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.ListCollectionsRequest;
import com.amazonaws.services.rekognition.model.ListCollectionsResult;

public class ListCollections {

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


      AmazonRekognition amazonRekognition =
 AmazonRekognitionClientBuilder.standard()
         .withEndpointConfiguration(new
 AwsClientBuilder.EndpointConfiguration("https://rekognition-fips.us-
east-1.amazonaws.com","us-east-1"))
         .build();


      System.out.println("Listing collections");
      int limit = 10;
      ListCollectionsResult listCollectionsResult = null;
      String paginationToken = null;
      do {
         if (listCollectionsResult != null) {
            paginationToken = listCollectionsResult.getNextToken();
         }
         ListCollectionsRequest listCollectionsRequest = new
 ListCollectionsRequest()
                 .withMaxResults(limit)
                 .withNextToken(paginationToken);

 listCollectionsResult=amazonRekognition.listCollections(listCollectionsRequest);

         List < String > collectionIds = listCollectionsResult.getCollectionIds();
         for (String resultId: collectionIds) {
```

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

   }
}
```

AWS CLI

Pour le AWS CLI, utilisez l'`--endpoint-url`argument pour spécifier le point de terminaison via lequel Amazon Rekognition est accessible. Cet exemple montre vos collections qui utilisent le point de terminaison FIPS dans la région USA Est (Ohio) :

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

Python

Pour Python, utilisez l'argument `endpoint_url` dans la fonction boto3.client. Définissez le point de terminaison que vous souhaitez spécifier. Cet exemple montre vos collections qui utilisent le point de terminaison FIPS dans la région USA Ouest (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()
```

Bonnes pratiques pour les capteurs, images d’entrée et vidéos

Cette section contient des informations sur les bonnes pratiques pour l’utilisation d’Amazon Rekognition. Ces bonnes pratiques vous aideront à optimiser les performances des opérations que vous invoquez. Si vous avez du mal à obtenir les résultats attendus d'une opération, assurez-vous de suivre les meilleures pratiques documentées ici.

Pour plus d'informations sur la latence des opérations d'imagerie, consultez les rubriques suivantes :

- the section called “Latence de fonctionnement d’Image Amazon Rekognition”

Les opérations de comparaison faciale et de recherche de visages vous obligent à suivre les meilleures pratiques spécifiques pour trouver des visages sur une image. Ces exigences ont également été documentées dans le lien suivant :

- the section called “Recommandations pour les images d’entrée de comparaison faciale”
- the section called “Recommandations pour rechercher des visages dans une collection”

Les sections suivantes expliquent comment configurer votre caméra pour chaque type de média qu'Amazon Rekognition est capable d'analyser :

- the section called “Recommandations pour la configuration de la caméra (image et vidéo)”
- the section called “Recommandations pour la configuration de la caméra (vidéos stockées et en streaming)”
- the section called “Recommandations pour la configuration de la caméra (vidéo en streaming)”

Les opérations Face Liveness ont également leurs propres meilleures pratiques qui doivent être suivies pour tirer le meilleur parti de l'outil de vérification de la qualité de vie :

- the section called “Recommandations pour l’utilisation de Face Liveness”

Latence de fonctionnement d'Image Amazon Rekognition

Pour garantir la latence la plus faible possible pour les opérations d'Image Amazon Rekognition, prenez en considération les éléments suivants :

- La région du compartiment Amazon S3 contenant vos images doit correspondre à la région que vous utilisez pour les opérations d'API d'Image Amazon Rekognition.
- Il est plus rapide d'appeler une opération d'Image Amazon Rekognition avec des octets d'image que de charger l'image dans un compartiment Amazon S3, puis de référencer l'image chargée dans une opération d'Image Amazon Rekognition. Envisagez cette approche si vous chargez des images dans Image Amazon Rekognition en vue d'un traitement pratiquement en temps réel. Tel est le cas, par exemple, des images chargées à partir d'une caméra IP ou des images chargées via un portail web.
- Si l'image se trouve déjà dans un compartiment Amazon S3, il est probablement plus rapide de la référencer dans une opération d'Image Amazon Rekognition que de transmettre des octets d'image à l'opération.

Recommandations pour les images d'entrée de comparaison faciale

Les modèles utilisés pour les opérations de comparaison faciale sont conçus pour fonctionner dans une grande variété de poses, d'expressions faciales, de tranches d'âge, de rotations, de conditions d'éclairage et de tailles. Nous vous recommandons de suivre les directives suivantes lorsque vous choisissez des photos de référence pour CompareFaces ou pour ajouter des visages à une collection à l'aide d'IndexFaces.

Recommandations générales pour la saisie d'images pour les opérations faciales

- Utilisez des images lumineuses et nettes. Évitez autant que possible d'utiliser des images qui peuvent être floues en raison du mouvement de la personne et de la caméra. DetectFaces peut être utilisé pour déterminer la luminosité et la netteté d'un visage.
- À des fins de détection du regard, il est recommandé de télécharger l'image originale à sa taille et à sa qualité d'origine.

- Utilisez une image dont le visage se trouve dans la plage d'angles recommandée. Le tangage doit être inférieur à 30 degrés lorsque la caméra est orientée vers le bas, et inférieur à 45 degrés lorsqu'elle est orientée vers le haut. Le lacet doit être inférieur à 45 degrés dans les deux directions. Il n'existe aucune restriction sur le roulis.
- Utilisez l'image d'un visage dont les deux yeux sont ouverts et visibles.
- Utilisez une image de visage qui n'est pas dissimulé ou étroitement rogné. L'image doit contenir l'intégralité de la tête et des épaules de la personne. Elle ne doit pas être rognée en suivant le cadre de délimitation du visage.
- Évitez les accessoires qui cachent le visage, comme les bandeaux et les masques.
- Utilisez une image dont le visage occupe une grande partie de l'image. La mise en correspondance des images où le visage occupe une plus grande partie de l'image est plus précise.
- Assurez-vous que les images sont suffisamment grandes en termes de résolution. Amazon Rekognition peut reconnaître des visages aussi petits que 50 x 50 pixels dans des résolutions d'image allant jusqu'à 1920 x 1080. Higher-resolution les images nécessitent une taille minimale de visage plus grande. Les visages dont la taille est supérieure à la taille minimale génèrent un ensemble de résultats de comparaison faciale plus précis.
- Utilisez des images en couleur.
- Utilisez des images de visage éclairé de façon uniforme, par opposition aux variations d'éclairage telles que les ombres.
- Utilisez des images dont le contraste est suffisant avec l'arrière-plan. Un arrière-plan monochrome avec un contraste élevé fonctionne bien.
- Pour les applications nécessitant une précision élevée, utilisez des images de visage dont les expressions faciales sont neutres avec la bouche fermée et un petit sourire voire aucun.

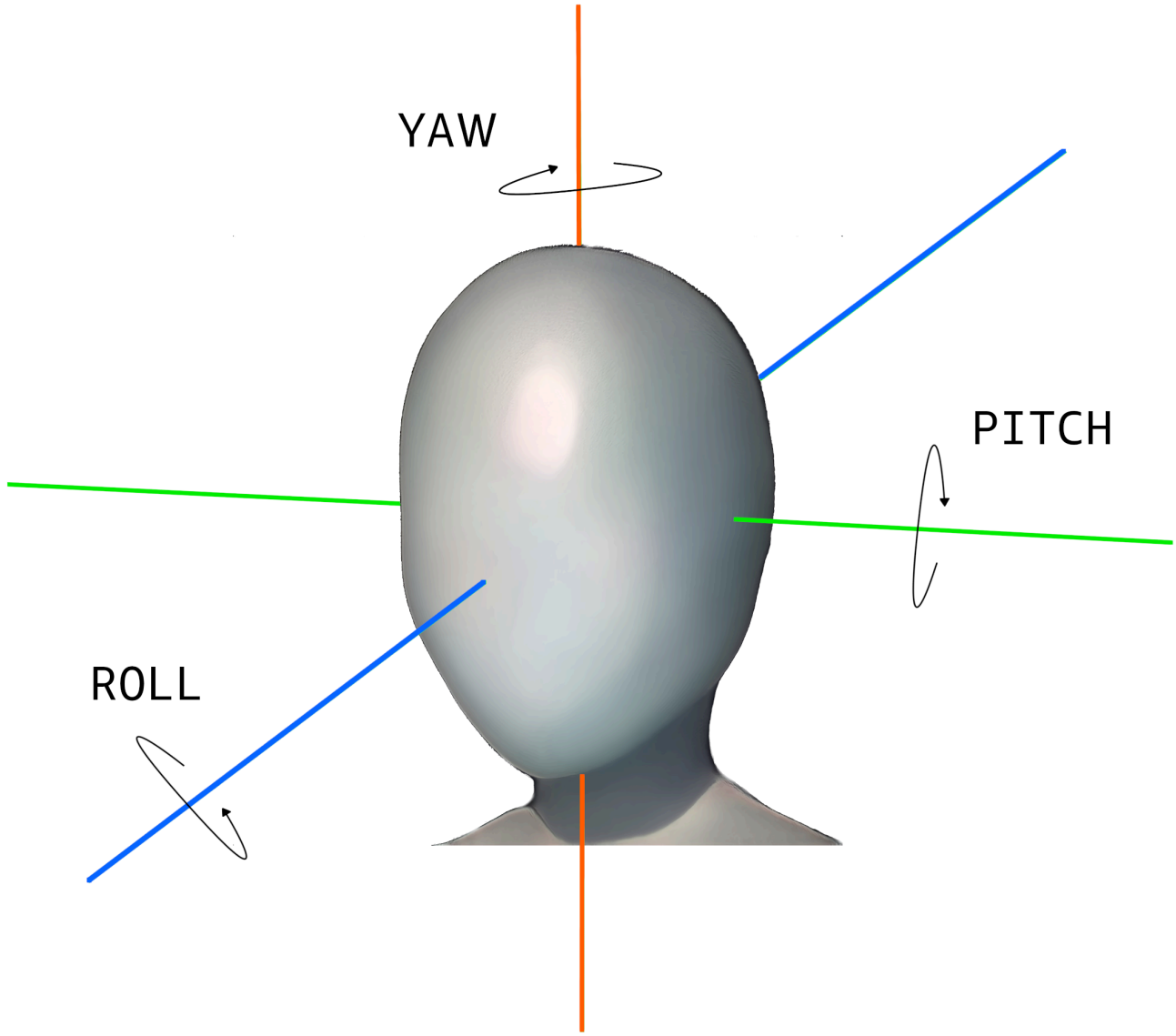
Recommandations pour rechercher des visages dans une collection

- Lorsque vous recherchez des visages dans une collection, assurez-vous que les images de visages récentes sont indexées.
- Lorsque vous créez une collection à l'aide d'`IndexFaces`, utilisez plusieurs images de visage d'une personne avec différents tangages et lacets (dans la plage d'angles recommandée). Nous recommandons d'indexer au moins 5 images de la personne : visage droit, visage tourné vers la gauche avec un lacet de 45 degrés ou moins, visage tourné vers la droite avec un lacet

de 45 degrés ou moins, visage incliné vers le bas avec un tangage de 30 degrés ou moins, et visage incliné vers le haut avec un tangage de 45 degrés ou moins. Si vous souhaitez suivre les instances de visage qui appartiennent à une même personne, pensez à utiliser l'attribut d'ID d'image externe si l'image en cours d'indexation ne contient qu'un seul visage. Par exemple, vous pouvez suivre 5 images de John Doe dans la collection portant les ID d'image externe comme `John_Doe_1.jpg, … John_Doe_5.jpg`.

Recommandations pour la configuration de la caméra (image et vidéo)

Les recommandations suivantes viennent compléter les Recommandations pour les images d'entrée de comparaison faciale.



- Résolution d'image — Il n'y a aucune exigence minimale en matière de résolution d'image, à condition que la résolution du visage soit de 50 x 50 pixels pour les images d'une résolution totale maximale de 1920 x 1080. Higher-resolution les images nécessitent une taille minimale de visage plus grande.

> **Note**
>
> La recommandation précédente se base sur la résolution native de la caméra. La génération d'une image haute résolution à partir d'une image faible résolution ne génère pas les résultats nécessaires à la recherche de visages (en raison des artefacts générés par le sur-échantillonnage de l'image).

- Angle de la caméra : il existe trois mesures pour l'angle de la caméra : tangage, roulis et lacet.
  - Tangage : nous recommandons de configurer un tangage inférieur à 30 degrés lorsque la caméra est orientée vers le bas, et inférieur à 45 degrés lorsque la caméra est orientée vers le haut.
  - Roulis : il n'existe pas d'exigence minimale pour ce paramètre. Amazon Rekognition peut gérer n'importe quelle valeur de roulis.
  - Lacet : nous recommandons de configurer un lacet inférieur à 45 degrés dans les deux directions.

  L'angle du visage le long de n'importe quel axe qui est capturé par la caméra est une combinaison de l'angle de la caméra face à la scène et de l'angle de la tête de la personne dans la scène. Par exemple, si la caméra est orientée vers le bas à 30 degrés et que la personne baisse la tête de 30 degrés, le tangage réel du visage tel qu'il est vu par la caméra est de 60 degrés. Dans ce cas, Amazon Rekognition n'est pas en mesure de reconnaître le visage. Nous vous recommandons de configurer les caméras de telle manière que les angles de la caméra supposent que les personnes regardent généralement la caméra avec un tangage global (combinaison du visage et de la caméra) inférieur ou égal à 30 degrés.
- Zoom de la caméra : la résolution minimale recommandée du visage de 50 x 50 pixels doit influencer ce paramètre. Nous vous recommandons d'utiliser le paramètre de zoom d'une caméra afin que la résolution des visages ne soit pas inférieure à 50 x 50 pixels.
- Hauteur de la caméra : le tangage recommandé de la caméra doit influencer ce paramètre.

Recommandations pour la configuration de la caméra (vidéos stockées et en streaming)

Les recommandations suivantes viennent compléter les Recommandations pour la configuration de la caméra (image et vidéo).

- Le codec doit être codé en h.264.
- La fréquence d'images recommandée est de 30 images par seconde. (Elle ne doit pas être inférieure à 5 images par seconde).
- La vitesse de transmission recommandée de l'encodeur est de 3 Mbit/s. (Elle ne doit pas être inférieure à 1,5 Mbit/s).
- Fréquence/résolution d'images : si la vitesse de transmission de l'encodeur est une contrainte, nous vous recommandons de préférer une résolution d'image supérieure à une fréquence d'images supérieure afin d'obtenir de meilleurs résultats de recherche faciale. Cela garantit qu'Amazon Rekognition obtienne la meilleure qualité de trame dans le débit binaire alloué. Toutefois, cela présente un inconvénient. En raison de la faible fréquence d'images, la caméra manque un mouvement rapide dans une scène. Il est important de comprendre le compromis entre ces deux paramètres pour une configuration donnée. Par exemple, si la vitesse de transmission possible maximale est de 1,5 Mbit/s, une caméra peut capturer 1080p à 5 images par seconde ou 720p à 15 images par seconde. Le choix entre ces deux paramètres dépend de l'application, tant que la résolution de visage recommandée de 50 x 50 pixels est respectée.

Recommandations pour la configuration de la caméra (vidéo en streaming)

La recommandation suivante vient compléter les Recommandations pour la configuration de la caméra (vidéos stockées et en streaming).

La bande passante Internet constitue une autre contrainte liée aux applications de streaming. Pour les vidéos en direct, Amazon Rekognition accepte uniquement Amazon Kinesis Video Streams en entrée. Vous devez comprendre la dépendance qui existe entre la vitesse de transmission de l'encodeur et la bande passante réseau disponible. La bande passante disponible doit au minimum prendre en charge la même vitesse de transmission que celle utilisée par la caméra pour coder le flux en direct. Les images capturées par la caméra sont ainsi relayées par l'intermédiaire d'Amazon Kinesis Video Streams. Si la bande passante disponible est inférieure à la vitesse de transmission de l'encodeur, Amazon Kinesis Video Streams diminue les bits en fonction de la bande passante réseau. Cela se traduit par une faible qualité vidéo.

Une configuration de streaming typique consiste à connecter plusieurs caméras à un hub réseau qui relaie les flux. Dans ce cas, la bande passante doit prendre en compte la somme cumulative des flux provenant de l'ensemble des caméras connectées au hub. Par exemple, si le hub est connecté

à 5 caméras qui codent à 1,5 Mbit/s, la bande passante réseau disponible doit être de 7,5 Mbit/s minimum. Pour éviter toute perte de paquets, vous devez envisager de maintenir la bande passante réseau à une valeur supérieure à 7,5 Mbit/s pour tenir compte des instabilités dues aux pertes de connexion entre une caméra et le hub. La valeur réelle dépend de la fiabilité du réseau interne.

Recommandations pour l'utilisation de Face Liveness

Les bonnes pratiques suivantes sont recommandées pour l'utilisation de Rekognition Face Liveness :

- Les utilisateurs doivent effectuer le test Face Liveness dans des environnements qui ne sont ni trop sombres ni trop lumineux et où l'éclairage est assez uniforme.
- Les utilisateurs doivent augmenter la luminosité de leur écran d'affichage à son niveau maximum lorsqu'ils effectuent des vérifications sur les navigateurs Web. Les SDK Mobile Native ajustent automatiquement la luminosité de l'écran.
- Choisissez un seuil de confiance qui reflète la nature de votre cas d'utilisation. Pour les cas d'utilisation présentant des problèmes de sécurité plus importants, utilisez un seuil élevé.
- Effectuez régulièrement des contrôles humains sur les images d'audit pour vous assurer que les attaques frauduleuses sont limitées au seuil de confiance que vous avez défini.
- Proposez une méthode alternative de vérification de la vivacité faciale à vos utilisateurs s'ils sont photosensibles ou s'ils ne souhaitent pas vérifier leur vivacité faciale à l'aide de Rekognition.
- N'envoyez pas et n'affichez pas le score du test de vivacité sur l'application utilisateur. Envoyez uniquement un signal de réussite ou d'échec.
- N'autorisez que cinq échecs au test de vivacité en trois minutes à partir d'un seul appareil. Après cinq échecs, durée de temporisation pour l'utilisateur de 30 à 60 minutes. Si le schéma est observé 3 à 5 fois de manière répétée, empêchez l'appareil utilisateur de passer des appels supplémentaires.
- Intégrez l'écran de préparation dans votre flux de travail afin que les utilisateurs puissent passer plus facilement les tests Face Liveness.
- Il vous incombe de fournir des avis de confidentialité juridiquement adéquats à vos utilisateurs finaux et d'obtenir le consentement nécessaire de leur part pour le traitement, le stockage, l'utilisation et le transfert du contenu par Face Liveness.

Détection d’objets et de concepts

Cette section fournit des informations de détection d’étiquettes dans des images et des vidéos avec Image Amazon Rekognition et Vidéo Amazon Rekognition.

Une étiquette ou une balise constituent un objet ou un concept (y compris des scènes et des actions) détecté dans une image ou une vidéo, en fonction de son contenu. Par exemple, une photo de personnes sur une plage tropicale peut contenir les étiquettes « Palmier » (objet), « Plage » (scène), « Courir » (action) et « Extérieur » (concept).

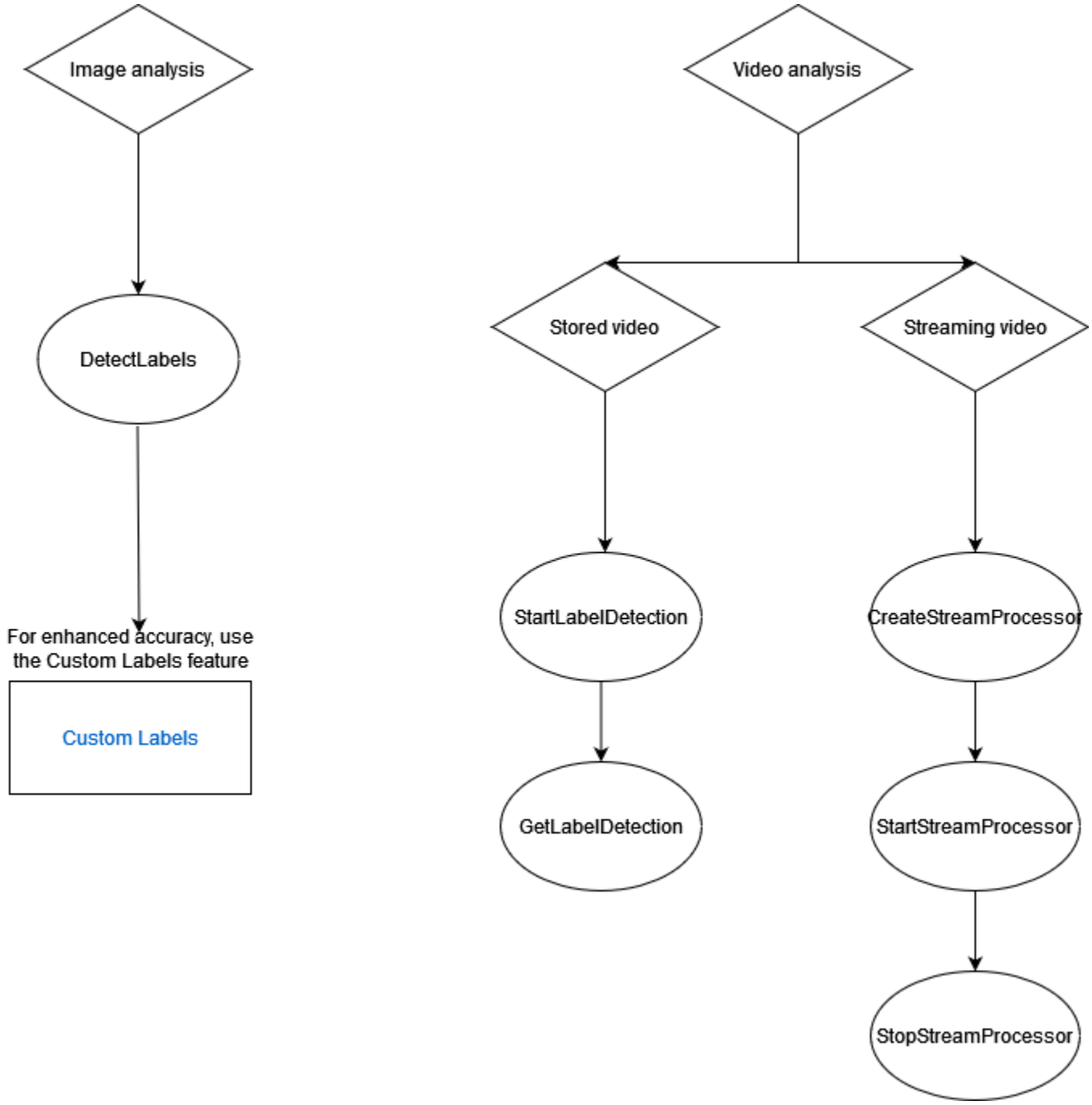
Étiquettes prises en charge par les opérations de détection d'étiquettes Rekognition

- Pour télécharger la dernière liste des étiquettes et des encadrements d’objets compatibles avec Amazon Rekognition, cliquez ici.
- Pour télécharger la liste précédente des étiquettes et des encadrés d’objets, cliquez ici.

> **Note**
>
> Amazon Rekognition fait des prédictions binaires liées au genre (homme, femme, fille, etc.) sur la base de l’apparence physique d’une personne sur une image donnée. Ce type de prédiction n’est pas conçu pour catégoriser l’identité de genre d’une personne, et vous ne devez pas utiliser Amazon Rekognition pour effectuer une telle détermination. Par exemple, un acteur masculin portant une perruque aux cheveux longs et des boucles d’oreilles pour un rôle peut être prédit comme une femme.
> L’utilisation d’Amazon Rekognition pour établir des prédictions binaires entre les sexes convient parfaitement aux cas d’utilisation dans lesquels les statistiques agrégées de répartition par sexe doivent être analysées sans identifier d’utilisateurs spécifiques. Par exemple, le pourcentage d’utilisateurs qui sont des femmes par rapport aux hommes sur une plateforme de médias sociaux.
> Nous vous déconseillons d’utiliser des prédictions binaires de sexe pour prendre des décisions ayant un impact sur les droits, la confidentialité ou l’accès aux services d’une personne.

Amazon Rekognition renvoie des étiquettes en anglais. Vous pouvez utiliser Amazon Translate pour traduire des étiquettes en anglais dans d’autres langues.

Le schéma suivant montre l'ordre des opérations d'appel, en fonction de vos objectifs d'utilisation des opérations Amazon Rekognition Image ou Amazon Rekognition Video :



Étiqueter les objets de réponse

Cadres de délimitation

Image Amazon Rekognition et Vidéo Amazon Rekognition peuvent renvoyer le cadre de délimitation pour des étiquettes d'objets courants tels que des voitures, des meubles, des appareils ou des animaux de compagnie. Les informations relatives au cadre de délimitation ne sont pas renvoyées pour les étiquettes d'objets moins courants. Vous pouvez utiliser des cadres de délimitation pour trouver les emplacements exacts des objets dans une image et le nombre d'instances d'objets détectés, ou pour mesurer la taille d'un objet à l'aide des dimensions de ces cadres.

Par exemple, dans l'image suivante, Image Amazon Rekognition peut détecter la présence d'une personne, d'un skateboard, de voitures garées et d'autres informations. Image Amazon Rekognition renvoie également le cadre de délimitation pour une personne détectée, ainsi que d'autres objets détectés, tels que des voitures et des roues.

Score de fiabilité

Vidéo Amazon Rekognition et Image Amazon Rekognition fournissent un score en pourcentage indiquant le degré de confiance d'Amazon Rekognition dans l'exactitude de chaque étiquette détectée.

Parents

Image Amazon Rekognition et Vidéo Amazon Rekognition utilisent une taxonomie hiérarchique des étiquettes d'ancêtres pour classer les étiquettes. Par exemple, une personne qui traverse une route peut être détectée en tant que Pedestrian (Piéton). L'étiquette parent de Pedestrian (Piéton) est Person (Personne). Les deux étiquettes sont renvoyées dans la réponse. Toutes les étiquettes d'ancêtre sont renvoyées et une étiquette donnée contient une liste de ses étiquettes parent et des autres étiquettes d'ancêtre. Par exemple, des étiquettes de grand-parent et d'arrière grand-parent, si elles existent. Vous pouvez utiliser des étiquettes parent pour créer des groupes d'étiquettes liées et autoriser l'interrogation des étiquettes similaires dans une ou plusieurs images. Par exemple, une requête pour tous les véhicules (Vehicles) peut renvoyer une voiture depuis une image et une moto depuis une autre image.

Catégories

Image Amazon Rekognition et Vidéo Amazon Rekognition renvoient des informations sur les catégories d'étiquettes. Les étiquettes font partie de catégories qui regroupent des étiquettes individuelles selon des fonctions et des contextes communs, tels que « Véhicules et automobiles » et « Produits alimentaires et boissons ». Une catégorie d'étiquette peut être une sous-catégorie d'une catégorie parent.

les alias ;

Outre le renvoi des étiquettes, Image Amazon Rekognition et Vidéo Amazon Rekognition renvoient tous les alias associés à l'étiquette. Les alias sont des étiquettes ayant la même signification ou des étiquettes visuellement interchangeables avec l'étiquette principale renvoyée. Par exemple, « téléphone cellulaire » est un alias de « téléphone portable ».

Dans les versions précédentes, Image Amazon Rekognition renvoyait des alias tels que « téléphone cellulaire » dans la même liste de noms d'étiquette principale qui contenait « téléphone portable ». Image Amazon Rekognition renvoie désormais « Téléphone cellulaire » dans un champ appelé « alias » et « Téléphone portable » dans la liste des noms d'étiquette principaux. Si votre application

repose sur les structures renvoyées par une version précédente de Rekognition, vous devrez peut-être transformer la réponse actuelle renvoyée par les opérations de détection d'étiquettes d'images ou de vidéos dans la structure de réponse précédente, dans laquelle toutes les étiquettes et alias sont renvoyés en tant qu'étiquettes principales.

Si vous devez transformer la réponse actuelle de l' DetectLabels API (pour la détection d'étiquettes dans les images) dans la structure de réponse précédente, consultez l'exemple de code dansTransformation de la DetectLabels réponse.

Si vous devez transformer la réponse actuelle de l' GetLabelDetection API (pour la détection d'étiquettes dans les vidéos stockées) dans la structure de réponse précédente, consultez l'exemple de code dansTransformer la GetLabelDetection réponse.

Propriétés de l'image

Image Amazon Rekognition renvoie des informations sur la qualité de l'image (netteté, luminosité et contraste) pour l'ensemble de l'image. La netteté et la luminosité sont également renvoyées pour le premier plan et l'arrière-plan de l'image. Les propriétés de l'image peuvent également être utilisées pour détecter les couleurs dominantes de l'ensemble de l'image, du premier plan, de l'arrière-plan et des objets dotés de cadres de délimitation.

Voici un exemple des ImageProperties données contenues dans la réponse d'une DetectLabels opération pour l'image suivante :

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	

Propriétés de l'image n'est pas disponible pour Vidéo Amazon Rekognition.

Version de modèle

Image Amazon Rekognition et Vidéo Amazon Rekognition renvoient la version du modèle de détection d'étiquette utilisé pour détecter les étiquettes dans une image ou une vidéo stockée.

Filtres d'inclusion et d'exclusion

Vous pouvez filtrer les résultats renvoyés par les opérations de détection d'étiquettes Image Amazon Rekognition et Vidéo Amazon Rekognition. Filtrez les résultats en fournissant des critères de filtrage pour les étiquettes et les catégories. Les filtres d'étiquette peuvent être inclusifs ou exclusifs.

Voir Détection des étiquettes dans une image pour plus d'informations concernant la filtration des résultats obtenus avec `DetectLabels`.

Voir Détection des étiquettes dans une vidéo pour plus d'informations concernant le filtrage des résultats obtenus par `GetLabelDetection`.

Tri et agrégation des résultats

Les résultats obtenus à partir de certaines opérations Vidéo Amazon Rekognition peuvent être triés et agrégés en fonction des horodatages et des segments vidéo. Lorsque vous récupérez les résultats d'une tâche de détection d'étiquettes ou de modération de contenu, avec `GetLabelDetection` ou `GetContentModeration` respectivement, vous pouvez utiliser les arguments `SortBy` et `AggregateBy` pour spécifier la manière dont vous souhaitez que vos résultats soient renvoyés. Vous pouvez utiliser `SortBy` avec `TIMESTAMP` ou `NAME` (noms d'étiquettes) et utiliser `TIMESTAMPS` ou `SEGMENTS` avec l' AggregateByargument.

Détection des étiquettes dans une image

Vous pouvez utiliser cette DetectLabelsopération pour détecter des étiquettes (objets et concepts) dans une image et récupérer des informations sur les propriétés de l'image. Les propriétés de l'image incluent des attributs tels que la couleur du premier plan et de l'arrière-plan, ainsi que la netteté, la luminosité et le contraste de l'image. Vous pouvez récupérer uniquement les étiquettes d'une image, uniquement les propriétés de l'image, ou les deux. Pour obtenir un exemple, consultez Analyse d'images stockées dans un compartiment Amazon S3.

Les exemples suivants utilisent différents AWS SDK et le AWS CLI to call`DetectLabels`. Pour en savoir plus sur la réponse d'opération `DetectLabels`, consultez DetectLabels réponse.

Pour détecter des étiquettes dans une image

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur avec `AmazonRekognitionFullAccess` et autorisations `AmazonS3ReadOnlyAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Chargez une image qui contient un ou plusieurs objets - par exemple des arbres, des maisons et un bateau - dans votre compartiment S3. L'image doit être au format .jpg ou .png.

   Pour en savoir plus, consultez Chargement d'objets dans Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

3. Utilisez les exemples suivants pour appeler l'opération `DetectLabels`.

Java

Cet exemple affiche la liste des étiquettes qui ont été détectées dans l'image d'entrée. Remplacez les valeurs de `bucket` et `photo` par le nom du compartiment Amazon S3 et le nom de l'image utilisés à l'étape 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

Cet exemple affiche la sortie JSON de l'opération `detect-labels` de l'interface de ligne de commande (CLI). Remplacez les valeurs de `bucket` et `photo` par le nom du compartiment Amazon S3 et le nom de l'image utilisés à l'étape 2. Remplacez la valeur de `profile-name` par le nom de votre profil de développeur.

```
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
```

Si vous accédez à la CLI sur un périphérique Windows, utilisez des guillemets doubles au lieu de guillemets simples et évitez les guillemets doubles internes par une barre oblique inverse (c'est-à-dire \) pour corriger les erreurs d'analyse que vous pourriez rencontrer. Par exemple, consultez ce qui suit :

```
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

Cet exemple affiche les étiquettes qui ont été détectées dans l'image d'entrée. Dans la fonction `main`, remplacez les valeurs de `bucket` et `photo` par les noms du compartiment Amazon S3 et de l'image que vous avez utilisés à l'étape 2. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

Cet exemple affiche la liste des étiquettes qui ont été détectées dans l'image d'entrée. Remplacez les valeurs de `bucket` et `photo` par le nom du compartiment Amazon S3 et le nom de l'image utilisés à l'étape 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

Cet exemple affiche la liste des étiquettes qui ont été détectées dans l'image d'entrée. Remplacez les valeurs de `bucket` et `photo` par le nom du compartiment Amazon S3 et le nom de l'image utilisés à l'étape 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

Cet exemple affiche la liste des étiquettes qui ont été détectées dans l'image d'entrée. Remplacez les valeurs de `bucket` et `photo` par le nom du compartiment Amazon S3 et le nom de l'image utilisés à l'étape 2. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

Si vous utilisez des TypeScript définitions, vous devrez peut-être utiliser à la `import AWS from 'aws-sdk'` place de`const AWS = require('aws-sdk')`, pour exécuter le

programme avec Node.js. Vous pouvez consulter l'AWS SDK pour Javascript pour plus de détails. Selon la façon dont vous avez configuré vos configurations, vous devrez peut-être également spécifier votre région avec `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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
//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 demande d'opération

La valeur d'entrée de `DetectLabel` est une image. Dans cet exemple d'entrée JSON, l'image source est chargée à partir d'un compartiment Amazon S3. `MaxLabels` correspond au nombre maximal d'étiquettes à renvoyer dans la réponse. `MinConfidence` est le niveau de fiabilité minimum que doit avoir Image Amazon Rekognition en ce qui concerne la précision de l'étiquette détectée pour que celle-ci soit renvoyée dans la réponse.

Fonctionnalités vous permet de spécifier une ou plusieurs caractéristiques de l'image que vous souhaitez renvoyer, en vous permettant de sélectionner `GENERAL_LABELS` et `IMAGE_PROPERTIES`. L'inclusion `GENERAL_LABELS` renverra les étiquettes détectées dans l'image d'entrée, tandis que l'inclusion `IMAGE_PROPERTIES` vous permettra d'accéder à la couleur et à la qualité de l'image.

Les paramètres vous permettent de filtrer les articles renvoyés à la fois pour les fonctionnalités `GENERAL_LABELS` et `IMAGE_PROPERTIES`. Pour les étiquettes, vous pouvez utiliser des filtres inclusifs et exclusifs. Vous pouvez également filtrer par étiquette spécifique, par étiquette individuelle ou par catégorie d'étiquette :

- LabelInclusionFilters - Vous permet de spécifier les libellés que vous souhaitez inclure dans la réponse.

- LabelExclusionFilters - Vous permet de spécifier les étiquettes que vous souhaitez exclure de la réponse.
- LabelCategoryInclusionFilters - Vous permet de spécifier les catégories d'étiquettes que vous souhaitez inclure dans la réponse.
- LabelCategoryExclusionFilters - Vous permet de spécifier les catégories d'étiquettes que vous souhaitez exclure de la réponse.

Vous pouvez également combiner des filtres inclusifs et exclusifs en fonction de vos besoins, en excluant certaines étiquettes ou catégories et en incluant d'autres.

`IMAGE_PROPERTIES` font référence aux couleurs dominantes et aux attributs de qualité d'une image tels que la netteté, la luminosité et le contraste. Lors de la détection `IMAGE_PROPERTIES`, vous pouvez spécifier le nombre maximum de couleurs dominantes à renvoyer (10 par défaut) à l'aide du paramètre `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 réponse

La réponse de `DetectLabels` est un tableau qui recense les étiquettes détectées dans l'image et qui indique le niveau de fiabilité de la détection.

Voici un exemple de réponse de `DetectLabels`. L'exemple de réponse ci-dessous contient divers attributs renvoyés pour GENERAL_LABELS, notamment :

- Nom : nom de l'étiquette détectée. Dans cet exemple, l'opération a détecté un objet portant l'étiquette Téléphone portable.
- À chaque étiquette est associé un niveau de fiabilité. Dans cet exemple, le niveau de confiance pour l'étiquette était de 99,36 %.
- Parents : étiquettes d'ancêtres associées à une étiquette détectée. Dans cet exemple, l'étiquette Téléphone portable possède une étiquette parent nommée Téléphone.
- Alias : informations sur les alias possibles pour l'étiquette. Dans cet exemple, l'étiquette Téléphone portable comporte un alias possible de téléphone cellulaire.
- Catégories : catégorie d'étiquette à laquelle appartient l'étiquette détectée. Dans cet exemple, il s'agit de la technologie et de l'informatique.

La réponse pour les étiquettes d'objets courants comprend des informations de cadre de délimitation pour l'emplacement de l'étiquette sur l'image d'entrée. Par exemple, l'étiquette Personne comporte un tableau d'instances contenant deux cadres de délimitation. Il s'agit des emplacements de deux personnes détectées dans l'image.

La réponse inclut également des attributs concernant IMAGE_PROPERTIES. Les attributs présentés par la fonctionnalité IMAGE_PROPERTIES sont les suivants :

- Qualité : informations sur la netteté, la luminosité et le contraste de l'image d'entrée, notées entre 0 et 100. La qualité est indiquée pour l'ensemble de l'image ainsi que pour l'arrière-plan et le premier plan de l'image, si disponible. Toutefois, le contraste n'est indiqué que pour l'ensemble de l'image, tandis que la netteté et la luminosité sont également indiquées pour l'arrière-plan et le premier plan.
- Couleur dominante : tableau des couleurs dominantes de l'image. Chaque couleur dominante est décrite par un nom de couleur simplifié, une palette de couleurs CSS, des valeurs RGB et un code hexadécimal.
- Premier plan : informations sur les couleurs dominantes, la netteté et la luminosité du premier plan de l'image d'entrée.

- Arrière-plan : informations sur les couleurs dominantes, la netteté et la luminosité de l'arrière-plan de l'image d'entrée.

Lorsque GENERAL_LABELS et IMAGE_PROPERTIES sont utilisés conjointement comme paramètres d'entrée, Image Amazon Rekognition renvoie également les couleurs dominantes des objets avec des cadres de délimitation.

Le champ `LabelModelVersion` contient le numéro de version du modèle de détection utilisé par `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"
}
```

Transformation de la DetectLabels réponse

Lorsque vous utilisez l' DetectLabels API, vous pouvez avoir besoin de la structure de réponse pour imiter l'ancienne structure de réponse de l'API, où les étiquettes principales et les alias figuraient dans la même liste.

Voici un exemple de la réponse API actuelle de DetectLabels:

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

```
                {
                "Name": "Cell Phone"
                }
            ]
        }
 ]
```

L'exemple suivant montre la réponse précédente de l'DetectLabelsAPI :

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

Si nécessaire, vous pouvez transformer la réponse actuelle pour qu'elle suive le format de l'ancienne réponse. Vous pouvez utiliser l'exemple de code suivant pour transformer la dernière réponse d'API en structure de réponse d'API précédente :

Python

L'exemple de code suivant montre comment transformer la réponse actuelle à partir de l' DetectLabels API. Dans l'exemple de code ci-dessous, vous pouvez remplacer la valeur de

EXAMPLE_INFERENCE_OUTPUT par le résultat d'une DetectLabels opération que vous avez exécutée.

```
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)
```

Voici un exemple de la réponse transformée :

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

Détection des étiquettes dans une vidéo

Vidéo Amazon Rekognition peut détecter les étiquettes (objets et concepts), ainsi que l’heure à laquelle une étiquette est détectée, dans une vidéo. Pour obtenir un exemple de code SDK, consultez

Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK). Pour un AWS CLI exemple, voirAnalyse d'une vidéo à l'aide du AWS Command Line Interface.

La détection d'étiquettes Vidéo Amazon Rekognition est une opération asynchrone. Pour démarrer la détection d'étiquettes dans une vidéo, appelez StartLabelDetection.

Vidéo Amazon Rekognition publie l'état d'achèvement de l'opération d'analyse vidéo dans une rubrique Amazon Simple Notification Service. Si l'analyse vidéo réussit, appelez GetLabelDetection pour obtenir les étiquettes détectées. Pour en savoir plus sur l'appel des opérations d'API d'analyse dans les vidéos, consultez Appeler les opérations de Vidéo Amazon Rekognition.

StartLabelDetection Demande

L'exemple suivant est un exemple JSON de requête `StartLabelDetection`. Vous fournissez l'opération `StartLabelDetection` avec une vidéo stockée dans un compartiment Amazon S3. Dans l'exemple de demande JSON, le compartiment Amazon S3 et le nom de la vidéo sont spécifiés, ainsi que `MinConfidence`, `Features`, `Settings`, et `NotificationChannel`.

`MinConfidence` est le niveau de fiabilité minimum que doit avoir Vidéo Amazon Rekognition en ce qui concerne la précision de l'étiquette détectée ou une instance de cadre de délimitation (en cas de détection) pour que celle-ci soit renvoyée dans la réponse.

Avec `Features`, vous pouvez spécifier que vous souhaitez que GENERAL_LABELS soit renvoyé dans le cadre de la réponse.

Avec `Settings`, vous pouvez filtrer les articles retournés pour GENERAL_LABELS. Pour les étiquettes, vous pouvez utiliser des filtres inclusifs et exclusifs. Vous pouvez également filtrer par étiquette spécifique, par étiquette individuelle ou par catégorie d'étiquette :

- `LabelInclusionFilters` : utilisé pour spécifier les étiquettes que vous souhaitez inclure dans la réponse.
- `LabelExclusionFilters` : utilisé pour spécifier les étiquettes que vous souhaitez exclure de la réponse.
- `LabelCategoryInclusionFilters` : utilisé pour spécifier les catégories d'étiquettes que vous souhaitez inclure dans la réponse.
- `LabelCategoryExclusionFilters` : utilisé pour spécifier les catégories d'étiquettes que vous souhaitez exclure de la réponse.

Vous pouvez également combiner des filtres inclusifs et exclusifs en fonction de vos besoins, en excluant certaines étiquettes ou catégories et en incluant d'autres.

`NotificationChannel` est l'ARN de la rubrique Amazon SNS sur laquelle vous souhaitez que Vidéo Amazon Rekognition publie l'état d'achèvement de l'opération de détection d'étiquettes. Si vous utilisez la politique d'autorisation `AmazonRekognitionServiceRole`, le nom de la rubrique Amazon SNS doit commencer par Rekognition.

Voici un exemple de demande `StartLabelDetection` au format JSON, y compris des filtres :

```
{
    "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 Réponse à l'opération

`GetLabelDetection` renvoie un tableau (`Labels`) comprenant des informations sur les étiquettes détectées dans la vidéo. Le tableau peut être trié en fonction de l'heure ou de l'étiquette détectée lors de la spécification du paramètre `SortBy`. Vous pouvez également sélectionner la manière dont les éléments de réponse sont agrégés à l'aide du paramètre `AggregateBy`.

L'exemple suivant constitue la réponse JSON de `GetLabelDetection`. Dans la réponse, notez les points suivants :

- Ordre de tri : le tableau d'étiquettes renvoyé est trié en fonction de l'heure. Pour trier par étiquette, spécifiez `NAME` dans le paramètre d'entrée `SortBy` pour `GetLabelDetection`. Si l'étiquette apparaît plusieurs fois dans la vidéo, il existera plusieurs instances de l'élément (LabelDetection). L'ordre de tri par défaut est `TIMESTAMP`, tandis que l'ordre de tri secondaire est `NAME`.
- Informations de l'étiquette :l'élément du tableau `LabelDetection` contient un objet (étiquette) qui contient quant à lui le nom de l'étiquette et le niveau de confiance d'Amazon Rekognition dans l'exactitude de l'étiquette détectée. Un objet `Label` comprend également une taxonomie hiérarchique des étiquettes et des informations sur le cadre de délimitation pour les étiquettes communes. `Timestamp` est le temps, en millisecondes, qui s'est écoulé entre le début de la vidéo et le moment où l'étiquette a été détectée.

  Les informations relatives aux catégories ou alias associés à une étiquette sont également renvoyées. Pour les résultats agrégés par vidéo `SEGMENTS`, les structures `StartTimestampMillis`, `EndTimestampMillis` et `DurationMillis` sont renvoyées, qui définissent respectivement l'heure de début, l'heure de fin et la durée d'un segment.
- Agrégation : spécifie la manière dont les résultats sont agrégés lorsqu'ils sont renvoyés. La valeur par défaut est d'agréger par `TIMESTAMPS`. Vous pouvez également choisir d'agréger par `SEGMENTS`, ce qui permet d'agréger les résultats sur une période donnée. En cas d'agrégation par `SEGMENTS`, les informations sur les instances détectées avec des cadres de délimitation ne sont pas renvoyées. Seules les étiquettes détectées lors des segments sont renvoyées.
- Informations sur la pagination :l'exemple montre une page d'informations de détection d'étiquette. Vous pouvez spécifier le nombre d'objets `LabelDetection` à renvoyer dans le paramètre d'entrée `MaxResults` de `GetLabelDetection`. Si le nombre de résultats est supérieur à `MaxResults`, `GetLabelDetection` renvoie un jeton (`NextToken`) utilisé pour obtenir la page de résultats suivante. Pour de plus amples informations, veuillez consulter Obtenir les résultats de l'analyse de Vidéo Amazon Rekognition.
- Informations sur la vidéo – La réponse comprend des informations sur le format de la vidéo (`VideoMetadata`) dans chaque page d'informations renvoyée par `GetLabelDetection`.

Voici un exemple de GetLabelDetection réponse au format JSON avec agrégation par 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
    }
}
```

Voici un exemple de GetLabelDetection réponse au format JSON avec agrégation par SEGMENTS :

```
{
    "JobStatus": "SUCCEEDED",
    "LabelModelVersion": "3.0",
    "Labels": [
        {
            "StartTimestampMillis": 225,
            "EndTimestampMillis": 3578,
            "DurationMillis": 3353,
            "Label": {
                "Name": "Car",
                "Categories": [
                  {
                    "Name": "Vehicles and Automotive"
                  }
                ],
                "Aliases": [
                  {
                    "Name": "Automobile"
                  }
                ],
                "Parents": [
                  {
                    "Name": "Vehicle"
                  }
                ],
                "Confidence": 99.9364013671875 // Maximum confidence score for Segment
 mode
            }
        },
        {
            "StartTimestampMillis": 7578,
```

```
            "EndTimestampMillis": 12371,
            "DurationMillis": 4793,
            "Label": {
                "Name": "Kangaroo",
                "Categories": [
                  {
                    "Name": "Animals and Pets"
                  }
                ],
                "Aliases": [
                  {
                    "Name": "Wallaby"
                  }
                ],
                "Parents": [
                  {
                    "Name": "Mammal"
                  }
                ],
                "Confidence": 99.9364013671875
            }
        },
        {
            "StartTimestampMillis": 22225,
            "EndTimestampMillis": 22578,
            "DurationMillis": 2353,
            "Label": {
                "Name": "Bicycle",
                "Categories": [
                  {
                    "Name": "Hobbies and Interests"
                  }
                ],
                "Aliases": [
                  {
                    "Name": "Bike"
                  }
                ],
                "Parents": [
                  {
                    "Name": "Vehicle"
                  }
                ],
                "Confidence": 99.9364013671875
```

```
            }
        }
    ],
    "VideoMetadata": {
        "ColorRange": "FULL",
        "DurationMillis": 5000,
        "Format": "MP4",
        "FrameWidth": 1280,
        "FrameHeight": 720,
        "FrameRate": 24
    }
}
```

Transformer la GetLabelDetection réponse

Lorsque vous récupérez des résultats avec l'opération d' GetLabelDetection API, vous aurez peut-être besoin de la structure de réponse pour imiter l'ancienne structure de réponse d'API, où les étiquettes principales et les alias figuraient dans la même liste.

L'exemple de réponse JSON trouvé dans la section précédente affiche la forme actuelle de la réponse API de GetLabelDetection.

L'exemple suivant montre la réponse précédente de l' 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
    }
}
```

Si nécessaire, vous pouvez transformer la réponse actuelle pour qu'elle suive le format de l'ancienne réponse. Vous pouvez utiliser l'exemple de code suivant pour transformer la dernière réponse d'API en structure de réponse d'API précédente :

```
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
```

Détection d’étiquettes dans des événements vidéo en streaming

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vous pouvez utiliser Vidéo Amazon Rekognition pour détecter les étiquettes dans les vidéos en streaming. Pour ce faire, vous créez un processeur de flux (CreateStreamProcessor) pour démarrer et gérer l'analyse de la vidéo en streaming.

Vidéo Amazon Rekognition utilise Amazon Kinesis Video Streams pour recevoir et traiter un flux vidéo. Lorsque vous créez le processeur de flux, vous choisissez ce que le processeur de flux doit détecter. Vous pouvez choisir des personnes, des forfaits et des animaux de compagnie, ou des personnes et des forfaits. Les résultats d'analyse sont envoyés dans votre compartiment Amazon S3 et dans les notifications Amazon SNS. Notez que Vidéo Amazon Rekognition détecte la présence d'une personne dans la vidéo, mais ne détecte pas s'il s'agit d'une personne en particulier. Pour rechercher un visage dans une collection dans une vidéo en streaming, voir the section called "Recherche de visages dans une collection en vidéo streaming".

Pour utiliser Vidéo Amazon Rekognition avec du streaming vidéo, votre application a besoin des éléments suivants :

- Un flux vidéo Kinesis pour envoyer des vidéos en streaming à Vidéo Amazon Rekognition. Pour plus d'informations, consultez le Guide du développeur Amazon Kinesis Video Streams.
- Un processeur de flux Vidéo Amazon Rekognition pour gérer l'analyse de la vidéo en streaming. Pour de plus amples informations, veuillez consulter Présentation des opérations du processeur de flux Vidéo Amazon Rekognition.
- Un compartiment Amazon S3. Vidéo Amazon Rekognition publie le résultat de la session dans le compartiment S3. La sortie inclut le cadre d'image dans lequel une personne ou un objet d'intérêt a été détecté pour la première fois. Vous devez être propriétaire du compartiment S3.
- Rubrique Amazon SNS dans laquelle Vidéo Amazon Rekognition publie des alertes intelligentes et un résumé de fin de session.

Rubriques

- Configuration de vos ressources Vidéo Amazon Rekognition et Amazon Kinesis
- Opérations de détection des étiquettes d'appel pour les événements vidéo en streaming

Configuration de vos ressources Vidéo Amazon Rekognition et Amazon Kinesis

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Les procédures suivantes décrivent les étapes à suivre pour approvisionner le flux vidéo Kinesis et les autres ressources utilisées pour détecter les étiquettes dans une vidéo en streaming.

Conditions préalables

Pour exécuter cette procédure, vous AWS SDK pour Java devez être installé. Pour de plus amples informations, veuillez consulter Premiers pas avec Amazon Rekognition. Le compte AWS que vous utilisez nécessite des autorisations d'accès à l'API Amazon Rekognition. Pour de plus amples informations, veuillez consulter Actions définies par Amazon Rekognition dans le Guide de l'utilisateur IAM.

Pour détecter les étiquettes dans un flux vidéo (AWS SDK)

1. Créez un compartiment Amazon S3. Notez le nom du compartiment et les préfixes de la clé que vous souhaitez utiliser. Vous aurez besoin de ces informations ultérieurement.
2. Créer une rubrique Amazon SNS. Vous pouvez l'utiliser pour recevoir des notifications lorsqu'un objet d'intérêt est détecté pour la première fois dans le flux vidéo. Notez l'Amazon Resource Name ARN) de la rubrique. Pour plus d'informations, consultez Création d'une règle Amazon SNS dans le Guide du développeur Amazon SNS.
3. Abonner un point de terminaison à une rubrique Amazon SNS. Pour plus d'informations, consultez Abonnement à une rubrique Amazon SNS dans le Guide du développeur Amazon SNS.
4. Créez un flux vidéo Kinesis et notez l'Amazon Resource Name (ARN) du flux.
5. Si ce n'est pas déjà fait, créez une fonction de service IAM pour permettre à Vidéo Amazon Rekognition d'accéder à vos flux vidéo Kinesis, à votre compartiment S3 et à votre rubrique

Amazon SNS. Pour de plus amples informations, veuillez consulter Donner accès aux processeurs de flux de détection d'étiquettes.

Vous pouvez ensuite créer le processeur de flux de détection d'étiquettes et démarrer le processeur de flux en utilisant le nom de processeur de flux que vous avez choisi.

> **Note**
>
> Démarrez le processeur de flux uniquement après avoir vérifié que vous pouvez intégrer du contenu multimédia dans le flux vidéo Kinesis.

Orientation et configuration de la caméra

Vidéo Amazon Rekognition Streaming Video Events est compatible avec toutes les caméras compatibles avec Kinesis Video Streams. Pour de meilleurs résultats, nous recommandons de placer la caméra entre 0 et 45 degrés par rapport au sol. La caméra doit être dans sa position verticale canonique. Par exemple, s'il y a une personne dans le cadre, celle-ci doit être orientée verticalement et la tête de la personne doit être plus haute dans le cadre que les pieds.

Donner accès aux processeurs de flux de détection d'étiquettes

Vous utilisez un rôle de service Gestion des identités et des accès AWS (IAM) pour donner à Amazon Rekognition Video un accès en lecture aux flux vidéo Kinesis. Pour ce faire, utilisez les rôles IAM pour permettre à Vidéo Amazon Rekognition d'accéder à votre compartiment Amazon S3 et à une rubrique Amazon SNS.

Vous pouvez créer une politique d'autorisation qui autorise Vidéo Amazon Rekognition à accéder à une rubrique Amazon SNS, à un compartiment Amazon S3 et à un flux vidéo Kinesis existants. Pour une procédure étape par étape utilisant le AWS CLI, voirthe section called "AWS CLI commandes pour configurer un rôle IAM de détection d'étiquettes".

Pour permettre à Vidéo Amazon Rekognition d'accéder aux ressources de détection des étiquettes

1. Créez une nouvelle stratégie d'autorisations avec l'éditeur de stratégie JSON IAM;, et utilisez la stratégie suivante. Remplacez `kvs-stream-name` par le nom du flux vidéo Kinesis, remplacez `topicarn` par l'Amazon Ressource Name (ARN) de la rubrique Amazon SNS que vous souhaitez utiliser, et remplacez `bucket-name` par le nom du compartiment Amazon S3.

2. Créez une fonction du service IAM;, ou mettez à jour une fonction du service IAM existant. Utilisez les informations suivantes pour créer la fonction du service IAM :
   1. Choisissez Rekognition pour le nom du service.
   2. Choisissez Rekognition pour le cas d'utilisation de la fonction de service.
   3. Attachez la stratégie d'autorisations que vous avez créée à l'étape 1.
3. Notez l'ARN de la fonction de service. Vous en avez besoin pour créer le processeur de flux avant d'effectuer des opérations d'analyse vidéo.
4. (Facultatif) Si vous utilisez votre propre AWS KMS clé pour chiffrer les données envoyées à votre compartiment S3, vous devez ajouter l'instruction suivante avec le rôle IAM. (Il s'agit du rôle IAM que vous avez créé pour la stratégie de clé, qui correspond à la clé gérée par le client que vous souhaitez utiliser.)

```
            {
                        "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 commandes pour configurer un rôle IAM de détection d'étiquettes

Si ce n'est pas déjà fait, configurez et configurez le AWS CLI avec vos informations d'identification.

Entrez les commandes suivantes dans le AWS CLI pour configurer un rôle IAM doté des autorisations nécessaires à la détection des étiquettes.

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

3. Créez un fichier de politique de relation de confiance (par exemple, assume-role-rekognition.json) avec le contenu suivant.

   JSON

```
{
  "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. Si le nom de la rubrique SNS pour laquelle vous souhaitez recevoir des notifications ne commence pas par le préfixe « AmazonRekognition », ajoutez la politique suivante :

   `aws iam attach-role-policy --role-name $IAM_ROLE_NAME --policy-arn "arn:aws:iam::aws:policy/AmazonSNSFullAccess" --region $AWS_REGION`
7. Si vous utilisez votre propre clé AWS KMS pour chiffrer les données envoyées à votre compartiment Amazon S3, mettez à jour la stratégie de clé gérée par le client que vous souhaitez utiliser.

   a. Créez un fichier kms_key_policy.json contenant le contenu suivant :

```
{
"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`. Remplacez KMS_KEY_ID par l'ID de clé KMS que vous avez créé.

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

Opérations de détection des étiquettes d'appel pour les événements vidéo en streaming

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vidéo Amazon Rekognition peut détecter des personnes ou des objets pertinents dans une vidéo en streaming et vous avertir lorsqu'ils sont détectés. Lorsque vous créez un processeur de flux de détection d'étiquettes, choisissez les étiquettes que vous souhaitez que Vidéo Amazon Rekognition détecte. Il peut s'agir de personnes, de colis et d'animaux domestiques, ou de personnes, de colis et d'animaux domestiques. Choisissez uniquement les étiquettes spécifiques que vous souhaitez détecter. Ainsi, les seules étiquettes pertinentes créent des notifications. Vous pouvez configurer des options pour déterminer quand stocker les informations vidéo, puis effectuer un traitement supplémentaire en fonction des étiquettes détectées dans l'image.

Une fois que vous avez configuré vos ressources, le processus de détection des libellés dans une vidéo en streaming est le suivant :

1. Créez le processeur de flux.
2. Démarrage du processeur de flux
3. Si un objet d'intérêt est détecté, vous recevez une notification Amazon SNS pour la première occurrence de chaque objet d'intérêt.
4. Le processeur de flux s'arrête lorsque le temps spécifié dans `MaxDurationInSeconds` est écoulé.
5. Vous recevez une notification Amazon SNS finale avec un résumé de l'événement.
6. Vidéo Amazon Rekognition publie un résumé détaillé de la session dans votre compartiment S3.

Rubriques

- Création du processeur de flux de détection d'étiquettes Vidéo Amazon Rekognition
- Démarrage du processeur de flux de détection d'étiquettes Vidéo Amazon Rekognition
- Analyse des résultats de détection des étiquettes

Création du processeur de flux de détection d'étiquettes Vidéo Amazon Rekognition

Avant de pouvoir analyser une vidéo en streaming, vous devez créer un processeur de streaming Amazon Rekognition Video (). CreateStreamProcessor

Si vous souhaitez créer un processeur de flux pour détecter les étiquettes qui vous intéressent et les personnes, fournissez en entrée un flux vidéo Kinesis (`Input`), des informations sur le compartiment Amazon S3 (`Output`) et un ARN de rubrique Amazon SNS (`StreamProcessorNotificationChannel`). Vous pouvez également fournir un ID de clé KMS pour chiffrer les données envoyées à votre compartiment S3. Vous spécifiez ce que vous souhaitez détecter dans `Settings`, comme les personnes, les colis et les personnes, ou les animaux domestiques, les personnes et les colis. Vous pouvez également spécifier l'endroit du cadre que vous souhaitez qu'Amazon Rekognition surveille avec `RegionsOfInterest`. L'exemple suivant est un exemple JSON de requête `CreateStreamProcessor`.

```
{
```

```
  "DataSharingPreference": { "OptIn":TRUE
  },
  "Input": {
    "KinesisVideoStream": {
      "Arn": "arn:aws:kinesisvideo:us-east-1:nnnnnnnnnnnn:stream/muh_video_stream/
nnnnnnnnnnnnn"
    }
  },
  "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
        }
      ]
    }
  ]
}
```

Notez que vous pouvez modifier la valeur `MinConfidence` lorsque vous spécifiez le `ConnectedHomeSettings` pour le processeur de flux. `MinConfidence` est une valeur numérique comprise entre 0 et 100 qui indique dans quelle mesure l'algorithme est certain de ses prédictions. Par exemple, une notification `person` avec une valeur de confiance de 90 signifie que l'algorithme est absolument certain que la personne est présente dans la vidéo. Une valeur de confiance de 10 indique qu'il pourrait y avoir une personne. Vous pouvez définir `MinConfidence` à la valeur de votre choix comprise entre 0 et 100 en fonction de la fréquence à laquelle vous souhaitez être averti. Par exemple, si vous souhaitez être averti uniquement lorsque Rekognition est absolument certaine de la présence d'un package dans l'image vidéo, vous pouvez définir `MinConfidence` à la valeur 90.

Par défaut, `MinConfidence` est défini à 50. Si vous souhaitez optimiser l'algorithme pour une plus grande précision, vous pouvez définir `MinConfidence` à une valeur supérieure à 50. Vous recevez alors moins de notifications, mais chaque notification est plus fiable. Si vous souhaitez optimiser l'algorithme pour augmenter le taux de rappel, vous pouvez régler `MinConfidence` à une valeur inférieure à 50 pour recevoir davantage de notifications.

Démarrage du processeur de flux de détection d'étiquettes Vidéo Amazon Rekognition

Pour démarrer l'analyse de vidéo en streaming, appelez StartStreamProcessor avec le nom de processeur de flux que vous avez spécifié dans `CreateStreamProcessor`. Lorsque vous exécutez l'opération `StartStreamProcessor` sur un processeur de flux de détection d'étiquettes, vous saisissez des informations de démarrage et d'arrêt pour déterminer le temps de traitement.

Lorsque vous démarrez le processeur de flux, l'état du processeur de flux de détection d'étiquettes change de la manière suivante :

1. Lorsque vous appelez `StartStreamProcessor`, l'état du processeur de flux de détection d'étiquettes passe de `STOPPED` ou `FAILED` vers `STARTING`.
2. Lorsque le processeur de flux de détection d'étiquettes fonctionne, il reste actif dans `STARTING`.
3. Lorsque le processeur de flux de détection d'étiquettes a terminé de fonctionner, l'état devient soit `STOPPED` ou `FAILED`.

Le `StartSelector` spécifie le point de départ du flux Kinesis pour démarrer le traitement. Vous pouvez utiliser l'horodatage du producteur KVS ou le numéro du fragment KVS. Pour de plus amples informations, veuillez consulter Fragment.

> **Note**
>
> Si vous utilisez l'horodatage KVS Producer, vous devez saisir l'heure en millisecondes.

Le `StopSelector` spécifie quand arrêter le traitement du flux. Vous pouvez spécifier la durée maximale de traitement de la vidéo. La durée par défaut est 10 secondes maximum. Notez que le temps de traitement réel peut être légèrement supérieur à la durée maximale, en fonction de la taille des fragments KVS individuels. Si la durée maximale est atteinte ou dépassée à la fin d'un fragment, le temps de traitement s'arrête.

L'exemple suivant est un exemple JSON de requête `StartStreamProcessor`.

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

Si le processeur de flux démarre avec succès, une réponse HTTP 200 est renvoyée. Un corps JSON vide est inclus.

Analyse des résultats de détection des étiquettes

Vidéo Amazon Rekognition publie les notifications provenant d'un processeur de flux de détection d'étiquettes de trois manières : des notifications Amazon SNS pour les événements de détection d'objets, une notification Amazon SNS pour un résumé de fin de session et un rapport détaillé sur le compartiment Amazon S3.

- Notifications Amazon SNS pour les événements de détection d'objets.

  Si des étiquettes sont détectées dans le flux vidéo, vous recevez des notifications Amazon SNS pour les événements de détection d'objets. Amazon Rekognition publie une notification la première fois qu'un objet d'intérêt ou une personne est détecté dans le flux vidéo. Les notifications incluent des informations telles que le type d'étiquette détecté, le niveau de confiance et un lien vers l'image du héros. Elles incluent également une image recadrée de la personne ou de l'objet détecté et un horodatage de détection. La notification de test a le format suivant :

```
{"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
    }
}
```

- Récapitulatif de fin de session Amazon SNS.

  Vous recevez également une notification Amazon SNS à la fin de la session de traitement du flux. Cette notification répertorie les métadonnées de la session. Cela inclut des détails tels que la durée du flux traité. La notification de test a le format suivant :

```
{"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
    }
```

```
}
```

- Rapport sur le compartiment Amazon S3.

  Vidéo Amazon Rekognition publie les résultats d'inférence détaillés d'une opération d'analyse vidéo vers le compartiment Amazon S3 fournis dans l'opération `CreateStreamProcessor`. Ces résultats incluent des cadres d'image dans lesquels un objet d'intérêt ou une personne a été détecté pour la première fois.

  Les cadres sont disponibles dans S3 dans le chemin suivant : ObjectKeyPrefix/StreamProcessorName/SessionId/*`service_determined_unique_path`*. Dans ce chemin, LabelKeyPrefixse trouvent un argument facultatif fourni par le client, StreamProcessorNamele nom de la ressource du processeur de flux et SessionIdun identifiant unique pour la session de traitement de flux. Remplacez-les en fonction de votre situation.

Détection des étiquettes personnalisées

Les étiquettes personnalisées Amazon Rekognition peuvent identifier les objets et les scènes dans des images spécifiques aux besoins de votre entreprise, telles que les logos ou les pièces de machines d'ingénierie. Pour de plus amples informations, veuillez consulter What Is Amazon Rekognition Custom Labels? dans le manuel Guide du développeur d'étiquettes personnalisées Amazon Rekognition.

Détection et analyse des visages

Amazon Rekognition met à votre disposition des API que vous pouvez utiliser pour détecter et analyser les visages sur les images et les vidéos. Cette section fournit une vue d'ensemble des opérations autres que le stockage pour l'analyse faciale. Ces opérations incluent des fonctionnalités telles que la détection des repères faciaux, l'analyse des émotions et la comparaison des visages.

Amazon Rekognition peut identifier les repères faciaux (par exemple, la position des yeux), détecter les émotions (par exemple, le bonheur ou la tristesse) et d'autres attributs (par exemple, la présence de lunettes, l'occlusion du visage). Lorsqu'un visage est détecté, le système analyse les attributs faciaux et renvoie un score de confiance pour chaque attribut.

Cette section contient des exemples d'opérations relatives aux images et aux vidéos.

Pour plus d'informations sur l'utilisation des opérations d'imagerie de Rekognition, consultez. Travail avec les images

Pour plus d'informations sur l'utilisation des opérations vidéo de Rekognition, consultez. Utilisation des opérations d'analyse vidéo stockées

Notez que ces opérations ne sont pas des opérations de stockage. Vous pouvez utiliser les opérations de stockage et les collections de visages pour enregistrer les métadonnées faciales des visages détectés sur une image. Par la suite, vous pouvez rechercher les visages stockés sur les images et dans les vidéos. Cela permet notamment de rechercher une personne spécifique dans une vidéo de Pour de plus amples informations, veuillez consulter Recherche de visages dans une collection.

Pour plus d'informations, consultez la section Visages des FAQ Amazon Rekognition.

> **Note**
>
> Les modèles de détection de visages utilisés par Amazon Rekognition Image et Amazon Rekognition Video ne prennent pas en charge la détection de visages chez des personnages ou des entités non humaines. cartoon/animated Si vous souhaitez détecter des personnages de dessins animés dans des images ou des vidéos, nous vous recommandons d'utiliser Étiquettes personnalisées Amazon Rekognition. Pour de plus amples informations, veuillez consulter le guide du développeur Étiquettes personnalisées Amazon Rekognition.

Rubriques

- Vue d'ensemble de la détection des visages et de la comparaison faciale
- Directives pour les attributs faciaux
- Détection de visages sur une image
- Comparaison de visages dans les images
- Détection de visages dans une vidéo stockée

Vue d'ensemble de la détection des visages et de la comparaison faciale

Amazon Rekognition permet aux utilisateurs d'accéder à deux applications d'apprentissage automatique principales pour les images contenant des visages : la détection des visages et la comparaison des visages. Ils intègrent des fonctionnalités cruciales telles que l'analyse faciale et la vérification d'identité, ce qui les rend essentiels pour diverses applications, de la sécurité à l'organisation de photos personnelles.

Détection des visages

Un système de détection de visage répond à la question suivante : « Y a-t-il un visage sur cette photo ? » Les principaux aspects de la détection des visages sont les suivants :

- Emplacement et orientation : détermine la présence, l'emplacement, l'échelle et l'orientation des visages dans les images ou les images vidéo.
- Attributs du visage : détecte les visages quels que soient leurs attributs tels que le sexe, l'âge ou la pilosité du visage.
- Informations supplémentaires : fournit des détails sur l'occlusion du visage et la direction du regard.

Comparaison des visages

Un système de comparaison de visages se concentre sur la question suivante : « Le visage d'une image correspond-il à un visage d'une autre image ? » Les fonctionnalités du système de comparaison des visages incluent :

- Prédictions de correspondance des visages : compare un visage dans une image à un visage dans une base de données fournie pour prédire les correspondances.
- Gestion des attributs du visage : gère les attributs pour comparer les visages indépendamment de leur expression, de leur pilosité et de leur âge.

Scores de confiance et détections manquées

Les systèmes de détection des visages et de comparaison des visages utilisent des scores de confiance. Un score de confiance indique la probabilité de prédictions, telles que la présence d'un visage ou une correspondance entre des visages. Des scores plus élevés indiquent une plus grande probabilité. Par exemple, un niveau de confiance de 90 % indique une probabilité de détection ou de correspondance correcte supérieure à 60 %.

Si un système de détection de visages ne détecte pas correctement un visage ou fournit une prédiction peu fiable pour un visage réel, il s'agit d'un detection/false négatif oublié. Si le système prédit de manière incorrecte la présence d'un visage avec un niveau de confiance élevé, il s'agit d'un faux alarm/false positif.

De même, un système de comparaison faciale peut ne pas faire correspondre deux visages appartenant à la même personne (detection/false négatif oublié) ou peut prédire à tort que deux visages de personnes différentes sont la même personne (faux alarm/false positif).

Conception de l'application et définition des seuils

- Vous pouvez définir un seuil qui indique le niveau de confiance minimal requis pour renvoyer un résultat. Le choix de seuils de confiance appropriés est essentiel pour la conception des applications et la prise de décision en fonction des résultats du système.
- Le niveau de confiance que vous avez choisi doit refléter votre cas d'utilisation. Quelques exemples de cas d'utilisation et de seuils de confiance :
- • Applications photo : un seuil inférieur (par exemple, 80 %) peut suffire pour identifier les membres de la famille sur les photos.
  - High-Stakes Scénarios : dans les cas d'utilisation où le risque de détection manquée ou de fausse alarme est plus élevé, comme dans le cas des applications de sécurité, le système doit utiliser un niveau de confiance plus élevé. Dans de tels cas, un seuil plus élevé (par exemple, 99 %) est recommandé pour des correspondances faciales précises.

Pour plus d'informations sur la définition et la compréhension des seuils de confiance, consultezRecherche de visages dans une collection.

Directives pour les attributs faciaux

Voici des informations détaillées sur la manière dont Amazon Rekognition traite et renvoie les attributs faciaux.

- FaceDetail Objet : pour chaque visage détecté, un FaceDetail objet est renvoyé. Il FaceDetail contient des données sur les repères faciaux, la qualité, la pose, etc.
- Prédictions d'attributs : des attributs tels que l'émotion, le sexe, l'âge, etc. sont prédits. Un niveau de confiance est attribué à chaque prédiction, et les prédictions sont renvoyées avec le score de confiance correspondant. Un seuil de confiance de 99 % est recommandé pour les cas d'utilisation sensibles. Pour l'estimation de l'âge, le point médian de la fourchette d'âge prévue constitue la meilleure approximation.

Notez que les prédictions relatives au genre et aux émotions sont basées sur l'apparence physique et ne doivent pas être utilisées pour déterminer l'identité sexuelle ou l'état émotionnel réels. Une prédiction binaire basée sur le genre (male/female) est basée sur l'apparence physique d'un visage dans une image donnée. Cela n'indique pas l'identité sexuelle d'une personne, et vous ne devez pas utiliser Rekognition pour prendre une telle décision. Nous vous déconseillons d’utiliser des prédictions binaires de sexe pour prendre des décisions ayant un impact sur les droits, la confidentialité ou l’accès aux services d’une personne. De même, la prédiction d'une émotion n'indique pas l'état émotionnel interne réel d'une personne, et vous ne devriez pas utiliser la Rekognition pour prendre

une telle décision. Une personne prétendant avoir un visage heureux sur une photo peut avoir l'air heureuse, mais elle n'est peut-être pas heureuse.

Applications et cas d'utilisation

Voici quelques applications pratiques et cas d'utilisation de ces attributs :

- Applications : Des attributs tels que le sourire, la pose et la netteté peuvent être utilisés pour sélectionner des photos de profil ou estimer des données démographiques de manière anonyme.
- Cas d'utilisation courants : les applications de réseaux sociaux et l'estimation démographique lors d'événements ou de magasins de détail en sont des exemples typiques.

Pour des informations plus détaillées sur chaque attribut, consultez FaceDetail.

Détection de visages sur une image

Amazon Rekognition Image DetectFacespermet de rechercher les principales caractéristiques du visage, telles que les yeux, le nez et la bouche, afin de détecter les visages sur une image d'entrée. Image Amazon Rekognition détecte les 100 plus grands visages d'une image.

Vous pouvez fournir l'image d'entrée sous la forme d'un tableau d'octets d'image (octets d'image encodés en base64) ou spécifier un objet Amazon S3. Dans cette procédure, vous chargez une image (JPEG ou PNG) dans votre compartiment S3 et spécifiez le nom de clé de l'objet.

Pour détecter des visages sur une image

1. Si vous ne l’avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur avec `AmazonRekognitionFullAccess` et autorisations `AmazonS3ReadOnlyAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Chargez une image (qui contient un ou plusieurs visages) dans votre compartiment S3.

   Pour en savoir plus, consultez Chargement d’objets dans Amazon S3 dans le Guide de l’utilisateur Amazon Simple Storage Service.

3. Utilisez les exemples suivants pour appeler `DetectFaces`.

Java

Cet exemple affiche la fourchette d'âges estimée pour les visages détectés et répertorie la version JSON de tous les attributs faciaux détectés. Remplacez la valeur de `photo` par le nom du fichier image. Remplacez la valeur de `amzn-s3-demo-bucket` par le compartiment Amazon S3; dans lequel l'image est stockée.

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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
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

Cet exemple affiche le résultat JSON de l'`detect-faces` AWS CLI opération. Remplacez `file` par le nom d'un fichier image. Remplacez `amzn-s3-demo-bucket` par le nom du compartiment Amazon S3; qui contient le fichier image.

```
aws rekognition detect-faces --image "{"S3Object":{"Bucket":"amzn-s3-demo-
bucket,"Name":"image-name"}}"\
                              --attributes "ALL" --profile profile-name --region
 region-name
```

Si vous accédez à la CLI sur un périphérique Windows, utilisez des guillemets doubles au lieu de guillemets simples et évitez les guillemets doubles internes par une barre oblique inverse (c'est-à-dire \) pour corriger les erreurs d'analyse que vous pourriez rencontrer. Par exemple, consultez ce qui suit :

```
aws rekognition detect-faces --image "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --attributes "ALL"
--profile profile-name --region region-name
```

Python

Cet exemple affiche la fourchette d'âge estimée et les autres attributs pour les visages détectés, et répertorie la version JSON de tous les attributs faciaux détectés. Remplacez la valeur de `photo` par le nom du fichier image. Remplacez la valeur de `amzn-s3-demo-bucket` par le compartiment Amazon S3; dans lequel l'image est stockée. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
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

Cet exemple affiche la fourchette d'âges estimée pour les visages détectés et répertorie la version JSON de tous les attributs faciaux détectés. Remplacez la valeur de `photo` par le nom du fichier image. Remplacez la valeur de `amzn-s3-demo-bucket` par le compartiment Amazon S3; dans lequel l'image est stockée.

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

Cet exemple affiche la tranche d'âge estimée pour les visages détectés et répertorie différents attributs faciaux. Remplacez la valeur de `photo` par le nom du fichier image. Remplacez la valeur de `amzn-s3-demo-bucket` par le compartiment Amazon S3; dans lequel l'image est stockée.

```
  # 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

Cet exemple affiche la tranche d'âge estimée pour les visages détectés et répertorie différents attributs faciaux. Remplacez la valeur de `photo` par le nom du fichier image. Remplacez la valeur de `amzn-s3-demo-bucket` par le compartiment Amazon S3; dans lequel l'image est stockée.

Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

Si vous utilisez des TypeScript définitions, vous devrez peut-être utiliser à la `import AWS from 'aws-sdk'` place de`const AWS = require('aws-sdk')`, pour exécuter le programme avec Node.js. Vous pouvez consulter l'AWS SDK pour Javascript pour plus de détails. Selon la façon dont vous avez configuré vos configurations, vous devrez peut-être également spécifier votre région avec `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 demande d'opération

La valeur d'entrée de `DetectFaces` est une image. Dans cet exemple, l'image est chargée à partir d'un compartiment Amazon S3. Le paramètre `Attributes` indique que tous les attributs faciaux doivent être renvoyés. Pour de plus amples informations, veuillez consulter Travail avec les images.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "input.jpg"
        }
    },
    "Attributes": [
        "ALL"
    ]
}
```

DetectFaces réponse à l'opération

`DetectFaces` renvoie les informations suivantes pour chaque visage détecté :

- Cadre de délimitation :coordonnées du cadre de délimitation entourant le visage.
- Fiabilité : niveau de certitude que le cadre de délimitation contient un visage.
- Repères faciaux : tableau des repères faciaux. Pour chaque repère (par exemple, l'œil gauche, l'œil droit et la bouche), la réponse indique les coordonnées x et y.
- Attributs faciaux : ensemble d'attributs faciaux, indiquant par exemple si le visage est occlus, renvoyé sous forme d'objet `FaceDetail`. L'ensemble comprend : barbe AgeRange, émotions, lunettes EyeDirection, genre EyesOpen FaceOccluded, moustache MouthOpen, sourire et lunettes de soleil. Pour chacun de ces attributs, la réponse fournit une valeur. La valeur peut être de différents types, tels qu'un type booléen (si une personne porte des lunettes de soleil), une chaîne

(qu'il s'agisse d'un homme ou d'une femme) ou une valeur de degré angulaire (pour la direction pitch/yaw du regard). De plus, pour la plupart des attributs, la réponse fournit également un niveau de fiabilité de la valeur détectée pour chaque attribut. Notez que si les EyeDirection attributs FaceOccluded et sont pris en charge lors de l'utilisation`DetectFaces`, ils ne le sont pas lors de l'analyse de vidéos avec `StartFaceDetection` et`GetFaceDetection`.

- Qualité : décrit la luminosité et la netteté du visage. Pour plus d'informations sur une détection optimale des visages, consultez Recommandations pour les images d'entrée de comparaison faciale.
- Pose : décrit la rotation du visage à l'intérieur de l'image.

La demande peut décrire un ensemble d'attributs faciaux que vous souhaitez recevoir. Un sous-ensemble `DEFAULT` d'attributs faciaux - `BoundingBox`, `Confidence`, `Pose`, `Quality`, et `Landmarks` - sera toujours renvoyé. Vous pouvez demander le retour d'attributs faciaux spécifiques (en plus de la liste par défaut) en utilisant `["DEFAULT", "FACE_OCCLUDED", "EYE_DIRECTION"]` ou en utilisant un seul attribut, par exemple `["FACE_OCCLUDED"]`. Vous pouvez demander tous les attributs faciaux en utilisant `["ALL"]`. Le fait de demander davantage d'attributs peut augmenter le temps de réponse.

Voici un exemple de réponse à un appel d'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
  }
}
```

Notez ce qui suit :

- Les données `Pose` décrivent la rotation du visage détecté. Vous pouvez utiliser la combinaison des données `BoundingBox` et `Pose` pour tracer le cadre de délimitation autour des visages que votre application affiche.
- `Quality` décrit la luminosité et la netteté du visage. Vous pouvez juger utile de comparer les visages de différentes images pour trouver le meilleur.
- La réponse précédente montre l'ensemble des `landmarks` faciaux que le service peut détecter, ainsi que l'ensemble des attributs du visage et des émotions. Pour obtenir toutes ces informations dans la réponse, définissez le paramètre `attributes` avec la valeur `ALL`. Par défaut, l'API `DetectFaces` renvoie uniquement les cinq attributs faciaux suivants : `BoundingBox`, `Confidence`, `Pose`, `Quality` et `landmarks`. Les repères par défaut renvoyés sont les suivants : `eyeLeft`, `eyeRight`, `nose`, `mouthLeft` et `mouthRight`.

Comparaison de visages dans les images

Avec Rekognition, vous pouvez comparer des visages entre deux images à l'aide de cette opération. CompareFaces Cette fonctionnalité est utile pour les applications telles que la vérification d'identité ou la correspondance de photos.

CompareFaces compare un visage de l'image source avec chaque visage de l'image cible. Les images sont transmises CompareFaces sous la forme suivante :

- Représentation d'une image codée en base64.
- Objets Amazon S3.

Détection des visages et comparaison des visages

La comparaison des visages est différente de la détection des visages. La détection des visages (qui utilise DetectFaces) identifie uniquement la présence et l'emplacement des visages dans une image ou une vidéo. En revanche, la comparaison de visages consiste à comparer un visage détecté dans une image source à des visages dans une image cible afin de trouver des correspondances.

Seuils de similarité

Utilisez le `similarityThreshold` paramètre pour définir le niveau de confiance minimal pour les correspondances à inclure dans la réponse. Par défaut, seuls les visages présentant un score de similarité supérieur ou égal à 80 % sont renvoyés dans la réponse.

> **Note**
>
> `CompareFaces`utilise des algorithmes d'apprentissage automatique, qui sont probabilistes. Un faux négatif est une prédiction incorrecte selon laquelle un visage de l'image cible présente un faible score de confiance en matière de similarité par rapport au visage de l'image source. Pour réduire le risque de faux négatifs, nous vous recommandons de comparer l'image cible à plusieurs images sources. Si vous prévoyez d'utiliser `CompareFaces` pour prendre une décision qui a une incidence sur les droits, la confidentialité ou l'accès aux services d'une personne, nous vous recommandons de transmettre le résultat à un humain pour examen et validation supplémentaire avant de passer à l'action.

Les exemples de code suivants montrent comment utiliser les CompareFaces opérations pour différents AWS SDK. Dans cet AWS CLI exemple, vous chargez deux images JPEG dans votre compartiment Amazon S3 et spécifiez le nom de la clé de l'objet. Dans les autres exemples, vous chargez deux fichiers à partir du système de fichiers local et les entrez sous forme de tableaux d'octets d'image.

Pour comparer des visages

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur doté d'autorisations `AmazonRekognitionFullAccess` et `AmazonS3ReadOnlyAccess` (à titre d'AWS CLI exemple uniquement). Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Utilisez l'exemple de code suivant pour appeler l'opération `CompareFaces`.

   Java

   Cet exemple affiche des informations sur des visages correspondants dans les images source et cible, qui sont chargées à partir du système de fichiers local.

   Remplacez la valeur de `sourceImage` et de `targetImage` par le chemin et le nom de fichier des images source et cible.

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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
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

Cet exemple affiche le résultat JSON de l'`compare-faces` AWS CLI opération.

Remplacez `amzn-s3-demo-bucket` par le nom du compartiment Amazon S3 qui contient les images source et cible. Remplacez la valeur de `source.jpg` et de `target.jpg` par le nom de fichier de l'image source et le nom de fichier de l'image cible.

```
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
```

Si vous accédez à la CLI sur un périphérique Windows, utilisez des guillemets doubles au lieu de guillemets simples et évitez les guillemets doubles internes par une barre oblique inverse (c'est-à-dire \) pour corriger les erreurs d'analyse que vous pourriez rencontrer. Par exemple, consultez ce qui suit :

```
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

Cet exemple affiche des informations sur des visages correspondants dans les images source et cible, qui sont chargées à partir du système de fichiers local.

Remplacez la valeur de `source_file` et de `target_file` par le chemin et le nom de fichier des images source et cible. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

Cet exemple affiche des informations sur des visages correspondants dans les images source et cible, qui sont chargées à partir du système de fichiers local.

Remplacez la valeur de `sourceImage` et de `targetImage` par le chemin et le nom de fichier des images source et cible.

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

Cet exemple affiche des informations sur des visages correspondants dans les images source et cible, qui sont chargées à partir du système de fichiers local.

Remplacez la valeur de `photo_source` et de `photo_target` par le chemin et le nom de fichier des images source et cible.

```
 # 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

Cet exemple affiche des informations sur des visages correspondants dans les images source et cible, qui sont chargées à partir du système de fichiers local.

Remplacez la valeur de `photo_source` et de `photo_target` par le chemin et le nom de fichier des images source et cible. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
// 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 demande d'opération

La valeur d'entrée de `CompareFaces` est une image. Dans cet exemple, les images source et cible sont chargées à partir du système de fichiers local. Le paramètre d'entrée `SimilarityThreshold` spécifie le niveau de fiabilité minimum auquel les visages comparés doivent correspondre pour être inclus dans la réponse. Pour de plus amples informations, veuillez consulter Travail avec les images.

```
{
    "SourceImage": {
        "Bytes": "/9j/4AAQSk2Q==..."
    },
    "TargetImage": {
        "Bytes": "/9j/4O1Q==..."
    },
    "SimilarityThreshold": 70
}
```

CompareFaces réponse à l'opération

La réponse inclut :

- Un tableau de correspondances de visages : une liste de visages correspondants avec des scores de similarité et des métadonnées pour chaque visage correspondant. Si plusieurs faces correspondent, `faceMatches`

le tableau inclut toutes les correspondances faciales.

- Détails de la correspondance faciale : chaque visage correspondant fournit également un cadre de délimitation, une valeur de confiance, des emplacements repères et un score de similarité.
- Liste des visages non correspondants : la réponse inclut également les visages de l'image cible qui ne correspondent pas au visage de l'image source. Comprend une boîte de délimitation pour chaque visage inégalé.
- Informations sur le visage source : inclut des informations sur le visage provenant de l'image source utilisée pour la comparaison, notamment le cadre de sélection et la valeur de confiance.

L'exemple montre qu'une correspondance faciale a été trouvée dans l'image cible. Pour cette correspondance, un cadre de délimitation et une valeur de fiabilité (niveau de fiabilité de Amazon Rekognition en ce qui concerne la présence d'un visage dans le cadre de délimitation) sont fournis. Le score de similarité de 99,99 indique à quel point les visages sont similaires. L'exemple montre également un visage trouvé par Amazon Rekognition dans l'image cible et qui ne correspond pas au visage analysé dans l'image source.


```
{
    "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
    }
}
```


Détection de visages dans une vidéo stockée

Vidéo Amazon Rekognition peut détecter des visages dans les vidéos stockées dans un compartiment Amazon S3 et fournir des informations telles que :

- Le ou les moments auxquels les visages sont détectés dans une vidéo.
- L'emplacement des visages dans le cadre vidéo au moment où ils ont été détectés.
- Des repères faciaux tels que la position de l'œil gauche.
- Attributs supplémentaires, comme expliqué sur la page the section called “Directives pour les attributs faciaux”.

La détection des visages par Vidéo Amazon Rekognition dans les vidéos stockées est une opération asynchrone. Pour démarrer la détection des visages dans les vidéos, appelez StartFaceDetection. Vidéo Amazon Rekognition publie l’état d’achèvement de l’analyse vidéo dans une rubrique Amazon Simple Notification Service (Amazon SNS). Si l'analyse vidéo aboutit, vous pouvez appeler GetFaceDetection pour obtenir les résultats de cette opération. Pour plus d’informations sur le démarrage de l’analyse vidéo et l’obtention des résultats, consultez Appeler les opérations de Vidéo Amazon Rekognition.

Cette procédure s’appuie sur le code figurant dans Analyse d’une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK), qui utilise une file d’attente Amazon Simple Queue Service (Amazon SQS) pour obtenir le statut d’achèvement d’une demande d’analyse vidéo.

Pour détecter des visages dans une vidéo stockée dans un compartiment Amazon S3 (SDK)

1. Effectuez une Analyse d’une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK).
2. Ajoutez le code suivant à la classe `VideoDetect` que vous avez créée à l’étape 1.

   AWS CLI

   - Dans l'exemple de code suivant, remplacez `amzn-s3-demo-bucket` et `video-name` par le nom du compartiment Amazon S3 et le nom du fichier vidéo que vous avez spécifiés à l’étape 2.
   - Remplacez `region-name` par la région AWS que vous utilisez. Remplacez la valeur de `profile_name` par le nom de votre profil de développeur.
   - Remplacez `TopicARN` par l’ARN de la rubrique Amazon SNS que vous avez créée à l’étape 3 de Configuration de Vidéo Amazon Rekognition.
   - Remplacez `RoleARN` par l’ARN de la fonction de service IAM que vous avez créé à l’étape 7 de Configuration de Vidéo Amazon Rekognition.

```
aws rekognition start-face-detection --video "{"S3Object":{"Bucket":"amzn-s3-
demo-bucket","Name":"Video-Name"}}" --notification-channel
"{"SNSTopicArn":"Topic-ARN","RoleArn":"Role-ARN"}" --region region-name --
profile profile-name
```

Si vous accédez à la CLI sur un périphérique Windows, utilisez des guillemets doubles au lieu de guillemets simples et évitez les guillemets doubles internes par une barre oblique inverse (c'est-à-dire \) pour corriger les erreurs d'analyse que vous pourriez rencontrer. Par exemple, consultez ce qui suit :

```
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
```

Après avoir exécuté l'opération `StartFaceDetection` et obtenu le numéro d'identification de la tâche, exécutez l'opération `GetFaceDetection` suivante et fournissez le numéro d'identification de la tâche :

```
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);


}
```

Dans la fonction `main`, remplacez les lignes :

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

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

avec :

```
        StartFaceDetection(amzn-s3-demo-bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetFaceDetectionResults();
```

Java V2

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
//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
```

Dans la fonction `main`, remplacez les lignes :

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

avec :

```
    analyzer.StartFaceDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetFaceDetectionResults()
```

> **Note**
>
> Si vous avez déjà exécuté un exemple de vidéo différent de Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK), la nom de la fonction à remplacer est différent.

3. Exécutez le code. Des informations sur les visages détectés dans la vidéo s'affichent à l'écran.

GetFaceDetection réponse à l'opération

`GetFaceDetection` renvoie un tableau (`Faces`) qui contient des informations sur les visages détectés dans la vidéo. Un élément de liste, FaceDetection, existe pour chaque occurrence de détection d'un visage dans la vidéo. Les éléments du tableau renvoyés sont classés par ordre chronologique, en millisecondes depuis le début de la vidéo.

Voici un exemple de réponse JSON partielle renvoyée par `GetFaceDetection`. Dans la réponse, notez les points suivants :

- Cadre de délimitation :coordonnées du cadre de délimitation entourant le visage.
- Fiabilité : niveau de certitude que le cadre de délimitation contient un visage.
- Repères faciaux : tableau des repères faciaux. Pour chaque repère (par exemple, l'œil gauche, l'œil droit et la bouche), la réponse indique les coordonnées `x` et `y`.
- Attributs faciaux — Un ensemble d'attributs faciaux, qui comprend : la barbe AgeRange, les émotions, les lunettes, le sexe EyesOpen, la moustache MouthOpen, le sourire et les lunettes de soleil. La valeur peut être de différents types : de type booléen (si la personne porte des lunettes de soleil ou non) ou une chaîne (si la personne est un homme ou une femme). De plus, pour la plupart des attributs, la réponse fournit également un niveau de fiabilité de la valeur détectée pour chaque attribut. Notez que si les EyeDirection attributs FaceOccluded et sont pris en charge lors de l'utilisation`DetectFaces`, ils ne le sont pas lors de l'analyse de vidéos avec `StartFaceDetection` et`GetFaceDetection`.
- Horodatage : heure à laquelle le visage a été détecté dans la vidéo.
- Informations sur la pagination : l'exemple montre une page d'informations sur la détection des visages. Vous pouvez spécifier le nombre d'éléments de personne à renvoyer dans le paramètre d'entrée `MaxResults` pour `GetFaceDetection`. Si le nombre de résultats est supérieur à `MaxResults`, `GetFaceDetection` renvoie un jeton (`NextToken`) qui sert à obtenir la page de

résultats suivante. Pour de plus amples informations, veuillez consulter Obtenir les résultats de l'analyse de Vidéo Amazon Rekognition.

- Informations sur la vidéo : la réponse comprend des informations sur le format vidéo (`VideoMetadata`) dans chaque page d'informations renvoyée par `GetFaceDetection`.
- Qualité : décrit la luminosité et la netteté du visage.
- Pose : décrit la rotation du visage.


```
{
    "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
    }
}
```

Recherche de visages dans une collection

Amazon Rekognition vous permet d'utiliser un visage en entrée pour rechercher des correspondances dans une collection de visages enregistrés. Vous commencez par stocker des informations sur les visages détectés dans des conteneurs côté serveur appelés « collections ». Les collections stockent à la fois les visages individuels et les utilisateurs (plusieurs visages d'une même personne). Les visages individuels sont stockés sous forme de vecteurs de visage, une représentation mathématique du visage (et non une image réelle du visage). Différentes images d'une même personne peuvent être utilisées pour créer et stocker plusieurs vecteurs de visage dans la même collection. Vous pouvez ensuite agréger plusieurs vecteurs de visage de la même personne pour créer un vecteur utilisateur. Les vecteurs utilisateur peuvent offrir une plus grande précision de recherche faciale grâce à des représentations plus robustes, contenant différents degrés d'éclairage, de netteté, de pose, d'apparence, etc.

Une fois que vous avez créé une collection, vous pouvez utiliser un visage en entrée pour rechercher des vecteurs utilisateur ou des vecteurs de visage correspondants dans une collection. La recherche par rapport aux vecteurs utilisateur peut améliorer considérablement la précision comparée à la recherche par rapport à des vecteurs faciaux individuels. Vous pouvez utiliser des visages détectés dans des images, des vidéos stockées et des vidéos en streaming pour rechercher des vecteurs de visage stockés. Vous pouvez utiliser les visages détectés dans les images pour effectuer une recherche par rapport aux vecteurs utilisateur enregistrés.

Pour stocker les informations faciales, vous devez procéder comme suit :

1. Créer une collection - Pour stocker des informations faciales, vous devez d'abord créer (CreateCollection) une collection faciale dans l'une des AWS régions de votre compte. Vous spécifiez cette collection de visages lorsque vous appelez l'opération `IndexFaces`.
2. Index des visages : l'IndexFacesopération détecte le ou les visages dans une image, extrait et stocke le ou les vecteurs de visage dans la collection. Cette opération vous permet de détecter les visages dans une image et de conserver les informations sur les caractéristiques faciales détectées dans une collection. Voici un exemple d'opération d'API basée sur le stockage, car le service conserve sur le serveur les informations relatives au vecteur de visage.

Pour créer un utilisateur et associer plusieurs vecteurs de visage à un utilisateur, vous devez effectuer les opérations suivantes :

1. Créer un utilisateur - Vous devez d'abord créer un utilisateur avec CreateUser. Vous pouvez améliorer la précision de la correspondance faciale en agrégeant plusieurs vecteurs de visage d'une même personne dans un vecteur utilisateur. Vous pouvez associer jusqu'à 100 vecteurs de visage à un vecteur utilisateur.
2. Associer des visages : après avoir créé l'utilisateur, vous pouvez ajouter des vecteurs de visage existants à cet utilisateur par le biais de l'AssociateFacesopération. Les vecteurs de visage doivent résider dans la même collection qu'un vecteur utilisateur pour être associés à ce vecteur utilisateur.

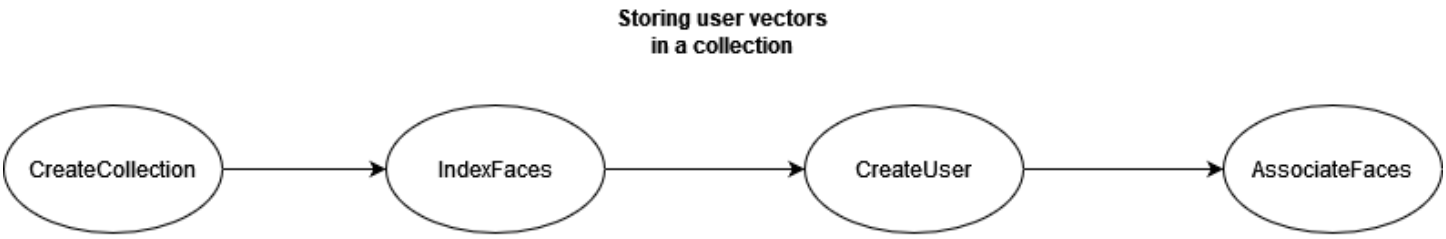
Après avoir créé une collection et enregistré les vecteurs de visages et d'utilisateurs, vous pouvez utiliser les opérations suivantes pour rechercher des correspondances de visages :

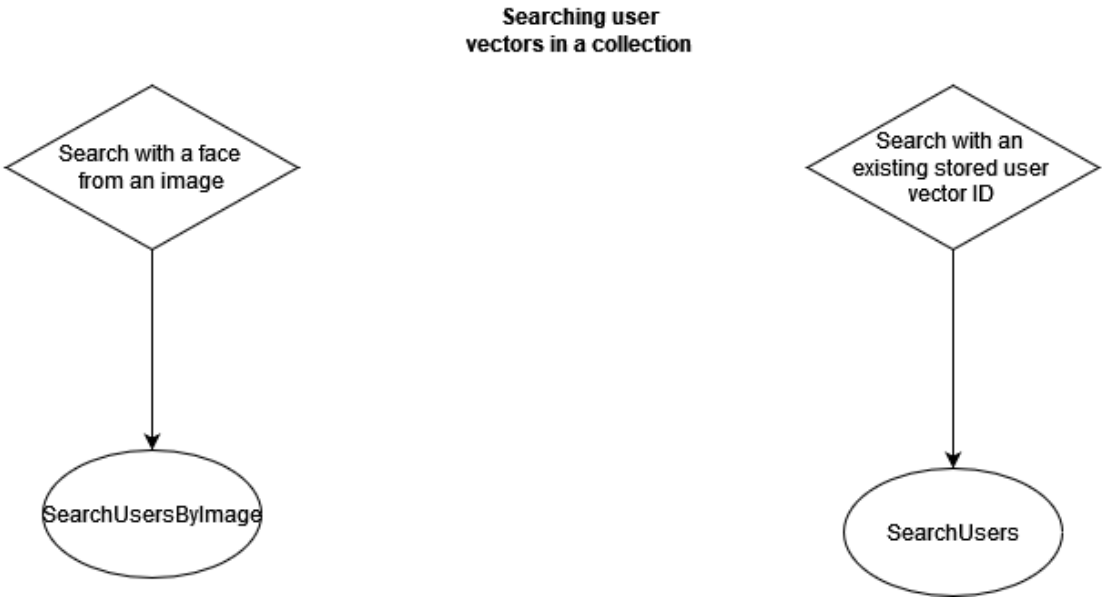
- SearchFacesByImage- Pour effectuer une recherche dans des visages individuels enregistrés à l'aide d'un visage issu d'une image.
- SearchFaces- Pour effectuer une recherche parmi des visages individuels enregistrés à l'aide d'un identifiant facial fourni.
- SearchUsers- Pour effectuer une recherche auprès d'utilisateurs enregistrés à l'aide d'un identifiant facial ou d'un identifiant utilisateur fourni.
- SearchUsersByImage- Pour effectuer une recherche parmi les utilisateurs enregistrés avec un visage à partir d'une image.
- StartFaceSearch- Pour rechercher des visages dans une vidéo enregistrée.
- CreateStreamProcessor- Pour rechercher des visages dans une vidéo en streaming.

> **Note**
>
> Les collections stockent des vecteurs de visages, qui sont des représentations mathématiques de visages. Les collections ne stockent pas d'images de visages.

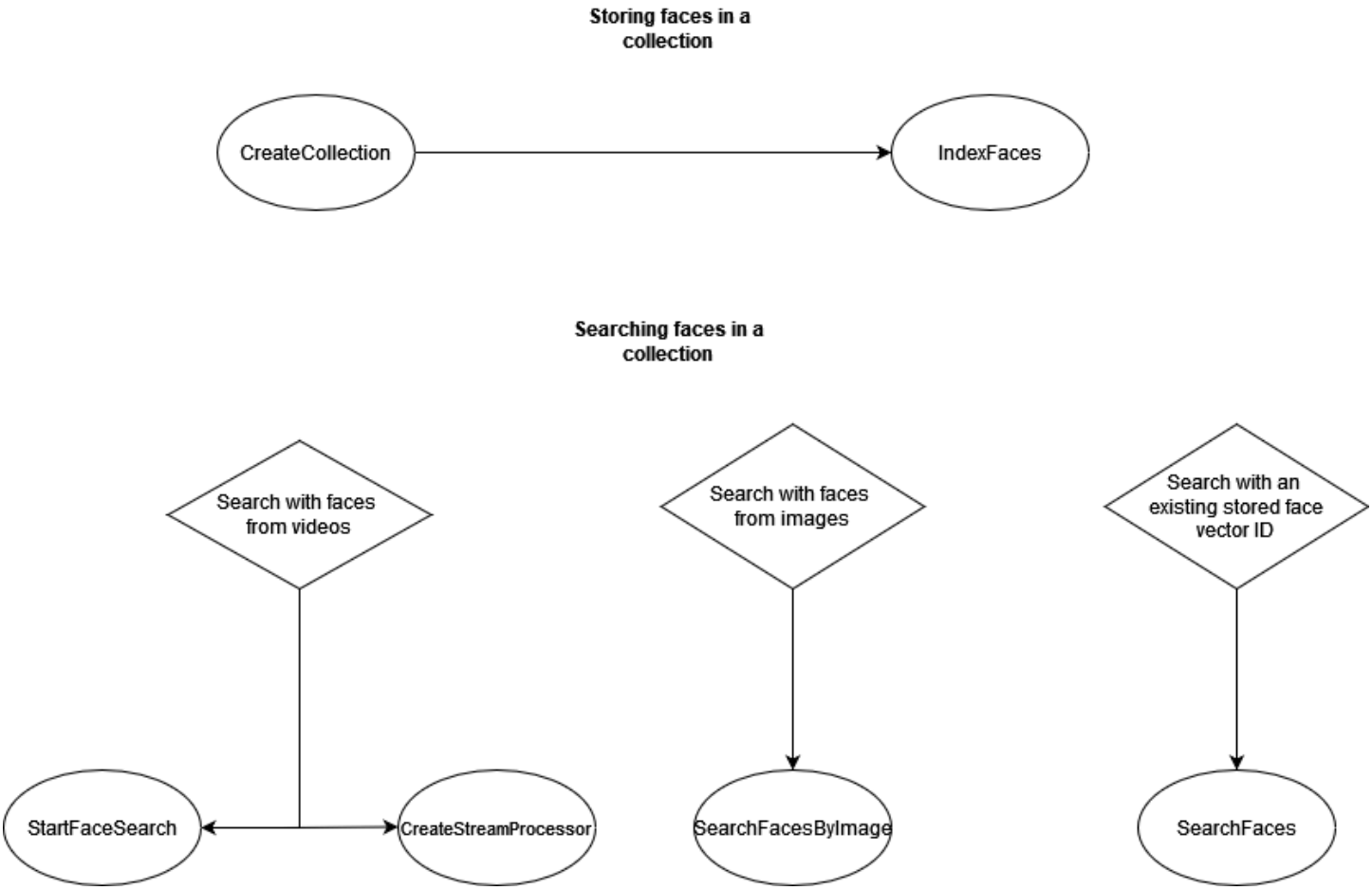
Les diagrammes suivants montrent l'ordre des opérations d'appel, en fonction de vos objectifs en matière d'utilisation des collections :

Pour une correspondance maximale avec les vecteurs utilisateur :





Pour une correspondance de haute précision avec des vecteurs faciaux individuels :



Vous pouvez utiliser les collections dans différents scénarios. Par exemple, vous pouvez créer une collection de visages qui stocke les visages détectés à partir d'images de badges d'employés numérisées et d'identifiants émis par le gouvernement à l'aide des opérations `IndexFaces` et `AssociateFaces`. Lorsqu'un employé entre dans le bâtiment, une image du visage est capturée et envoyée à l'opération `SearchUsersByImage`. Si la correspondance des visages offre un score de similarité suffisamment élevé (par exemple, 99 %), vous pouvez authentifier l'employé.

Gestion des collections de visages, des visages et des utilisateurs

La collection de visages est la ressource utilisée par Amazon Rekognition pour contenir des informations relatives aux visages et aux utilisateurs. Vous utilisez différentes opérations de l'API Rekognition pour gérer la collection elle-même, ainsi que tous les visages ou vecteurs de visages stockés dans la collection.

Gérer une collection

La collection de visages est la principale ressource Amazon Rekognition, et chaque collection de visages que vous créez possède un Amazon Resource Name (ARN) unique. Vous créez chaque collection de visages dans une AWS région spécifique dans votre compte. Au moment d'être créée, une collection est associée à la version la plus récente du modèle de détection de visages. Pour de plus amples informations, veuillez consulter Comprendre le versionnement des modèles.

Vous pouvez effectuer les opérations de gestion suivantes dans une collection :

- Créer une collection avec CreateCollection. Pour de plus amples informations, veuillez consulter Création d'une collection.
- Répertorier les collections disponibles avec ListCollections. Pour de plus amples informations, veuillez consulter Créer une liste de collections.
- Décrire une collection avec DescribeCollection. Pour de plus amples informations, veuillez consulter Description d'une collection.
- Supprimer une collection avec DeleteCollection. Pour de plus amples informations, veuillez consulter Suppression d'une collection.

Gestion des visages dans une collection

Une fois que vous avez créé une collection de visages, vous pouvez y stocker des visages. Amazon Rekognition propose les opérations suivantes pour gérer les visages d'une collection :

- L'opération IndexFaces détecte les visages dans l'image d'entrée (JPEG ou PNG) et les ajoute à la collection de visages spécifiée. Un ID de visage unique est renvoyé pour chaque visage détecté dans l'image. Une fois que vous avez conservé les visages, vous pouvez explorer la collection pour y rechercher des correspondances de visages. Pour de plus amples informations, veuillez consulter Ajout de visages à une collection.
- L'opération ListFaces dresse la liste des visages présents dans une collection. Pour de plus amples informations, veuillez consulter Ajout de visages à une collection.
- L'opération DeleteFaces permet de supprimer des visages d'une collection. Pour de plus amples informations, veuillez consulter Suppression de visages d'une collection.

Gérer les utilisateurs dans une collection

Après avoir enregistré plusieurs vecteurs de visage d'une même personne, vous pouvez améliorer la précision en associant tous ces vecteurs de visage dans un vecteur utilisateur unique. Vous pouvez utiliser les opérations suivantes pour gérer vos utilisateurs :

- CreateUser- L'opération crée un nouvel utilisateur dans une collection avec un ID utilisateur unique fourni.
- AssociateUsers- Ajoutez 1 à 100 identifiants faciaux uniques à un identifiant utilisateur. Après avoir associé au moins un identifiant facial à un utilisateur, vous pouvez rechercher des correspondances avec cet utilisateur dans votre collection.
- ListUsers- Répertorie les utilisateurs d'une collection.
- DeleteUsers- Supprime un utilisateur d'une collection avec l'ID utilisateur fourni.
- DisassociateFaces- Supprime un ou plusieurs identifiants faciaux d'un utilisateur.

Utilisation de seuils de similarité pour associer et faire correspondre des visages

Les seuils de similarité sont utilisés à la fois pour associer et faire correspondre des visages. Ce qui suit est un guide d'utilisation des seuils de similarité pour les deux cas d'utilisation.

Utilisation de seuils de similarité pour associer des visages

Lorsque vous associez des visages à l'aide de cette AssociateFacesopération, il est important de vous assurer que les visages associés à un utilisateur proviennent tous de la même personne. Pour vous aider, le paramètre `UserMatchThreshold` indique le niveau de confiance minimal requis pour que le nouveau visage soit associé à un visage `UserID` qui contient déjà un `FaceID`. Cela permet de s'assurer que les `FaceIds` sont associés au `UserID` correct. La valeur est comprise entre 0 et 100 et la valeur par défaut est 75.

Utilisation des seuils de similarité pour faire correspondre les visages

Nous vous permettons de contrôler les résultats de toutes les opérations de recherche (CompareFaces, SearchFaces, SearchFacesByImage, SearchUsers, SearchUsersByImage) en fournissant un seuil de similarité comme paramètre d'entrée.

`FaceMatchThreshold`, est l'attribut en entrée de seuil de similarité pour `SearchFaces` et `SearchFacesByImage`. Il contrôle le nombre de résultats renvoyés en fonction de la ressemblance avec le visage mis en correspondance. L'attribut de seuil de similarité pour `SearchUsers` et `SearchUsersByImage` est `UserMatchThreshold`. Il contrôle le nombre de résultats renvoyés en fonction de la ressemblance avec le visage mis en correspondance. L'attribut de seuil est `SimilarityThreshold` pour `CompareFaces`.

Les réponses dont la valeur d'attribut de réponse `Similarity` est inférieure au seuil ne sont pas renvoyées. Ce seuil est important pour calibrer votre cas d'utilisation, car il peut déterminer le nombre de faux positifs inclus dans vos résultats de correspondance. Ce seuil contrôle la sensibilité de vos résultats de recherche ; plus le seuil est bas, plus la sensibilité est élevée.

Tous les systèmes d'apprentissage automatique (Machine Learning) sont probabilistes. Vous devez faire preuve de discernement dans la configuration du seuil de similarité approprié, en fonction de votre cas d'utilisation. Par exemple, si vous cherchez à développer une application photo pour identifier les membres de famille qui se ressemblent, vous pouvez choisir un seuil plus bas (par exemple, 80 %). En revanche, pour de nombreux cas d'utilisation d'application des lois, nous vous recommandons d'utiliser une valeur de seuil élevée de 99 % ou plus pour réduire le risque d'erreur d'identification.

En plus de `FaceMatchThreshold` et `UserMatchThreshold`, vous pouvez utiliser l'attribut de réponse `Similarity` comme moyen pour réduire risque d'erreur d'identification. Par exemple, vous pouvez choisir d'utiliser un seuil bas (tel que 80 %) pour renvoyer plus de résultats. Ensuite, vous pouvez utiliser l'attribut de réponse Similarité (pourcentage de similarité) pour affiner le choix et filtrer les bonnes réponses dans votre application. Là encore, l'utilisation d'une similarité plus élevée (par exemple, 99 % et plus) réduit le risque d'erreur d'identification.

Conseils pour l'indexation des visages dans les scénarios courants

Les conseils ci-dessous vous permettront d'utiliser `IndexFaces` dans des scénarios courants.

Applications de sécurité publique ou critique

- Appelez IndexFaces avec des images ne contenant qu'un seul visage par image et associez l'ID de visage renvoyé à l'identifiant de la personne dans l'image.
- Vous pouvez utiliser DetectFaces avant l'indexation pour vérifier que l'image ne contient qu'un seul visage. Si plusieurs visages sont détectés, soumettez à nouveau l'image après avoir vérifié qu'elle

ne contient qu'un seul visage. Cela évite d'indexer plusieurs visages par inadvertance et de les associer à la même personne.

Applications de partage de photos et de réseaux sociaux

- Pour les cas d'utilisation tels que les albums de photos de famille, vous devez appeler `IndexFaces` sans appliquer de restrictions sur les images qui contiennent plusieurs visages. Dans ce cas, vous devez identifier chaque personne dans chaque photo et utiliser ces informations pour trier les photos par les personnes qu'elles contiennent.

Utilisation générale

- Pour améliorer la qualité de la mise en correspondance, indexez plusieurs images différentes de la même personne, notamment avec des attributs faciaux différents (position du visage, pilosité faciale, etc.), créez un utilisateur et associez les différents visages à cet utilisateur.
- Intégrez un processus de vérification pour que les mises en correspondance ayant échoué puissent être indexées avec l'identifiant de visage approprié afin d'améliorer les mises en correspondance de visages suivantes.
- Pour plus d'informations sur la qualité des images, consultez Recommandations pour les images d'entrée de comparaison faciale.

Recherche de visages et d'utilisateurs dans une collection

Après avoir créé une collection de visages et enregistré des vecteurs de visages sous forme de vecteurs and/or utilisateur, vous pouvez rechercher des correspondances de visages dans une collection de visages. Avec Amazon Rekognition vous pouvez rechercher des visages dans une collection qui correspondent à l'un des éléments suivants :

- Un ID de visage fourni (SearchFaces). Pour de plus amples informations, veuillez consulter Recherche d'un visage avec un ID de visage.
- Le visage le plus grand dans une image fournie (SearchFacesByImage). Pour de plus amples informations, veuillez consulter Recherche d'un visage à l'aide d'une image.
- Des visages dans une vidéo stockée. Pour de plus amples informations, veuillez consulter Recherche de visages dans des vidéos stockées.

- Des visages dans une vidéo en streaming. Pour de plus amples informations, veuillez consulter Utilisation des vidéos streaming.

Vous pouvez utiliser l'opération `CompareFaces` pour comparer un visage d'une image source avec les visages de l'image cible. La portée de cette comparaison se limite aux visages détectés dans l'image cible. Pour de plus amples informations, veuillez consulter Comparaison de visages dans des images.

Les différentes opérations de recherche présentées dans la liste suivante comparent un visage (identifié par un `FaceId` ou une image d'entrée) à tous les visages stockés dans une collection de visages donnée :

- SearchFaces
- SearchFacesByImage
- SearchUsers
- SearchUsersByImage

Cas d'utilisation impliquant la sécurité publique

Outre les recommandations répertoriées dans les Bonnes pratiques pour les capteurs, images d'entrée et vidéos et les Conseils pour l'indexation des visages dans les scénarios courants, vous devez utiliser les bonnes pratiques suivantes lors du déploiement des systèmes de détection et de comparaison faciale dans les cas d'utilisation qui impliquent la sécurité publique. Tout d'abord, vous devez utiliser des seuils de confiance de 99 % ou plus pour réduire les erreurs et les faux positifs. Ensuite, vous devez faire appel à des personnes pour vérifier les résultats envoyés par un système de détection ou de comparaison des visages, et vous ne devez pas prendre de décisions basées sur les résultats du système sans passer par une étape de vérification humaine. Les systèmes de détection et de comparaison des visages doivent servir d'outil pour aider à restreindre le champ et permettre aux personnes de procéder à une vérification rapide et d'étudier les options. Enfin, nous vous recommandons d'être transparent sur l'utilisation des systèmes de détection et de comparaison faciale dans ces cas d'utilisation, et, dans la mesure du possible, d'informer les utilisateurs finaux et les personnes concernées de l'utilisation de ces systèmes, d'obtenir les autorisations d'utilisation associées, et de fournir un mécanisme par lequel les utilisateurs finaux et les personnes concernées peuvent laisser des commentaires pour améliorer le système.

Si vous êtes un organisme chargé de l'application de la loi qui utilise la fonction de comparaison des visages Amazon Rekognition dans le cadre d'enquêtes criminelles, vous devez respecter les exigences énoncées dans les Conditions de service AWS. Cela inclut les éléments suivants :

- Disposer d'un personnel dûment formé pour examiner toutes les décisions d'agir susceptibles d'affecter les libertés civiles ou les droits de l'homme équivalents d'une personne.
- Former le personnel à l'utilisation responsable des systèmes de reconnaissance faciale.
- Rendre publique votre utilisation des systèmes de reconnaissance faciale.
- Ne pas utiliser Amazon Rekognition à des fins de surveillance soutenue d'une personne sans étude indépendante ou circonstances urgentes.

Dans tous les cas, les correspondances de la comparaison faciale doivent être étudiées dans le contexte d'autres preuves irréfutables, et ne doivent pas être utilisées comme élément déterminant unique pour prendre une décision. Toutefois, si la comparaison faciale est utilisée pour des scénarios autres que l'application de la loi (par exemple, pour déverrouiller un téléphone ou authentifier l'identité d'un employé pour accéder à un bâtiment privé sécurisé), ces décisions ne nécessitent pas d'audit manuel, car elles n'ont pas d'impact sur les libertés civiles ou les droits de l'homme équivalents d'une personne.

Si vous envisagez d'utiliser un système de détection ou de comparaison faciale pour les cas d'utilisation qui impliquent la sécurité publique, vous devez respecter les bonnes pratiques mentionnées précédemment. Vous devez également consulter les ressources publiées relatives à l'utilisation de la comparaison faciale. Ces ressources incluent le document Face Recognition Policy Development Template For Use In Criminal Intelligence and Investigative Activities fourni par le Bureau of Justice Assistance du Département de la Justice des États-Unis. Le modèle fournit plusieurs ressources relatives à la biométrie et à la comparaison faciale. Il est conçu pour fournir aux organismes chargés de l'application de la loi et aux organismes de sécurité publique un cadre pour développer des stratégies de comparaison faciale qui sont conformes aux lois applicables, réduisent les risques de confidentialité et établissent la responsabilité et la surveillance des entités. Le document Best Privacy Practices for Commercial Use of Facial Recognition rédigé par la National Telecommunications and Information Administration et le document Best Practices for Common Uses of Facial Recognition rédigé par le personnel de la Federal Trade Commission figurent parmi les autres ressources disponibles. Il est possible que d'autres ressources soient élaborées et publiées dans le futur. Vous devez vous informer en continu sur ce sujet important.

Pour rappel, vous devez respecter l'ensemble des lois applicables quant à leur utilisation des services AWS, et vous ne pouvez pas utiliser un service AWS d'une manière qui enfreint les droits

d'autrui ou qui peut porter préjudice à autrui. Cela signifie que vous ne pouvez pas utiliser les services AWS pour les cas d'utilisation de sécurité publique d'une manière qui fait preuve de discrimination illégale à l'égard d'une personne ou qui viole la procédure officielle, la vie privée ou les libertés individuelles d'une personne. Vous devez, si nécessaire, obtenir des conseils juridiques appropriés pour passer en revue les exigences légales ou les questions relatives à votre cas d'utilisation.

Utiliser Amazon Rekognition pour renforcer la sécurité publique

Amazon Rekognition peut aider dans les scénarios de sécurité publique et d'application de la loi, tels que la recherche d'enfants perdus, la lutte contre le trafic d'êtres humains ou la prévention des crimes. Dans les scénarios de sécurité publique et d'application de la loi, selon les cas, prenez en considération les éléments suivants :

- Utilisez Amazon Rekognition comme première étape pour trouver des correspondances possibles. Les réponses fournies par les opérations de détection des visages de Amazon Rekognition vous permettent d'obtenir rapidement un ensemble de correspondances potentielles à prendre en compte.
- N'utilisez pas les réponses Amazon Rekognition pour prendre des décisions autonomes pour les scénarios qui nécessitent d'être analysés par une personne. Si vous êtes un organisme chargé de l'application de la loi qui utilise Amazon Rekognition pour identifier une personne dans le cadre d'une enquête criminelle, et que des mesures seront prises en fonction de l'identification susceptibles d'affecter les libertés civiles de cette personne ou ses droits de l'homme équivalents, la décision d'agir doit être prise par un une personne dûment formée sur la base de son examen indépendant de la preuve d'identification.
- Utilisez un seuil de similarité de 99 % pour les scénarios dans lesquels des correspondances de similarité de visages très précises sont nécessaires. Par exemple pour l'authentification de l'accès à un bâtiment.
- Lorsque le respect des droits civiques est en jeu, par exemple dans les cas d'utilisation impliquant l'application de la loi, utilisez des seuils de confiance de 99 % ou plus, et faites vérifier les prédictions de la comparaison faciale par un intervenant humain afin de garantir que les droits civiques sont respectés.
- Utilisez un seuil de similarité inférieur à 99 % pour les scénarios qui tirent avantage d'un ensemble plus important de correspondances potentielles. Par exemple pour la recherche de personnes disparues. Si nécessaire, vous pouvez utiliser l'attribut de réponse Similarité afin de déterminer le niveau de similarité des correspondances potentielles par rapport à la personne que vous voulez reconnaître.

- Élaborez un plan pour gérer les correspondances de visage de type faux positif renvoyées par Amazon Rekognition. Par exemple, améliorez la correspondance en utilisant plusieurs images de la même personne lorsque vous créez l'index avec l'opération IndexFaces. Pour de plus amples informations, veuillez consulter Conseils pour l'indexation des visages dans les scénarios courants.

Dans d'autres cas d'utilisation (comme les réseaux sociaux), nous vous recommandons de faire preuve de discernement pour évaluer si les résultats d'Amazon Rekognition nécessitent une vérification par un opérateur humain. De plus, en fonction des exigences de votre application, le seuil de similarité peut être inférieur.

Création d'une collection

Vous pouvez utiliser l'opération CreateCollection pour créer une collection.

Pour de plus amples informations, veuillez consulter Gérer une collection.

Pour créer une collection (SDK)

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Utilisez les exemples suivants pour appeler l'opération `CreateCollection`.

   Java

   L'exemple suivant crée une collection et affiche son Amazon Resource Name (ARN).

   Remplacez la valeur d'`collectionId` par le nom de la collection que vous souhaitez créer.

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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
//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

Cette AWS CLI commande affiche la sortie JSON pour l'opération `create-collection` CLI.

Remplacez la valeur d'`collection-id` par le nom de la collection que vous souhaitez créer.

Remplacez la valeur de `profile_name` par le nom de votre profil de développeur.

```
aws rekognition create-collection --profile profile-name --collection-id
 "collection-name"
```

Python

L'exemple suivant crée une collection et affiche son Amazon Resource Name (ARN).

Remplacez la valeur de `collection_id` par le nom de la collection que vous souhaitez créer. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

import boto3

def create_collection(collection_id):
    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')

    # Create a collection
    print('Creating collection:' + collection_id)
    response = client.create_collection(CollectionId=collection_id)
    print('Collection ARN: ' + response['CollectionArn'])
    print('Status code: ' + str(response['StatusCode']))
    print('Done...')

def main():
    collection_id = "collection-id"
    create_collection(collection_id)

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

.NET

L'exemple suivant crée une collection et affiche son Amazon Resource Name (ARN).

Remplacez la valeur de `collectionId` par le nom de la collection que vous souhaitez créer.

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

Dans l'exemple suivant, remplacez la valeur de `region` par le nom de la région associée à votre compte et remplacez la valeur de `collectionName` par le nom souhaité pour votre collection.

Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
//Copyright 2018 Amazon.com, Inc. or its 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 demande d'opération

La valeur d’entrée de `CreationCollection` est le nom de la collection que vous souhaitez créer.

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

CreateCollection réponse à l'opération

Amazon Rekognition crée la collection et renvoie l'Amazon Resource Name (ARN) de la collection que vous venez de créer.

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

Baliser des collections

Vous pouvez identifier, organiser, rechercher et filtrer les collections Amazon Rekognition à l'aide de balises. Chaque balise est une étiquette composée d'une clé définie par l'utilisateur et d'une valeur.

Vous pouvez également utiliser des balises pour contrôler l'accès à une collection à l'aide de Identity and Access Management (IAM). Pour plus d'informations, consultez la section Contrôle de l'accès aux AWS ressources à l'aide de balises de ressources.

Rubriques

- Ajouter des balises à une nouvelle collection
- Ajouter des balises à une collection existante
- Répertorier les balises d'une collection
- Supprimer des balises d'une collection

Ajouter des balises à une nouvelle collection

Vous pouvez ajouter des balises à une collection lors de sa création à l'aide de l'opération `CreateCollection`. Spécifiez une ou plusieurs balises dans le paramètre d'entrée du tableau `Tags`.

AWS CLI

Remplacez la valeur de `profile_name` par le nom de votre profil de développeur.

```
aws rekognition create-collection --collection-id "collection-name" --tags
 "{"key1":"value1","key2":"value2"}" --profile profile-name
```

Pour les appareils Windows :

```
aws rekognition create-collection --collection-id "collection-name" --tags
 "{\"key1\":\"value1\",\"key2\":\"value2\"}" --profile profile-name
```

Python

Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
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()
```

Ajouter des balises à une collection existante

Pour ajouter une ou plusieurs balises à une collection existante, utilisez l'opération `TagResource`. Spécifiez l'Amazon Resource Name (ARN) (`ResourceArn`) de la collection et les balises (`Tags`) que vous voulez ajouter. L'exemple suivant vous montre comment ajouter deux balises.

AWS CLI

Remplacez la valeur de `profile_name` par le nom de votre profil de développeur.

```
aws rekognition tag-resource --resource-arn collection-arn --tags
 "{"key1":"value1","key2":"value2"}" --profile profile-name
```

Pour les appareils Windows :

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

Python

Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
# Copyright 2018 Amazon.com, Inc. or its 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**
>
> Si vous ne connaissez pas le nom de la ressource Amazon de la collection, vous pouvez utiliser l'opération `DescribeCollection`.

Répertorier les balises d'une collection

Pour répertorier les balises attachées à une collection, utilisez l'opération `ListTagsForResource` et spécifiez l'ARN de la collection (`ResourceArn`). La réponse est une carte des clés de balise et des valeurs attachées à la collection spécifiée.

AWS CLI

Remplacez la valeur de `profile_name` par le nom de votre profil de développeur.

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

Python

Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
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()
```

La sortie affiche une liste de balises attachées à la collection :

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

Supprimer des balises d'une collection

Pour supprimer une ou plusieurs balises d'une collection, utilisez l'opération `UntagResource`. Spécifiez l'ARN du modèle (`ResourceArn`) et les clés de balise (`Tag-Keys`) que vous souhaitez supprimer.

AWS CLI

Remplacez la valeur de `profile_name` par le nom de votre profil de développeur.

```
aws rekognition untag-resource --resource-arn resource-arn --profile profile-name --
tag-keys "key1" "key2"
```

Vous pouvez aussi spécifier les clés de balise dans ce format :

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

Python

Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
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()
```

Créer une liste de collections

Vous pouvez utiliser l'opération ListCollections pour dresser la liste des collections situées dans la région que vous utilisez.

Pour de plus amples informations, veuillez consulter Gérer une collection.

Pour dresser la liste des collections (SDK)

1. Si vous ne l'avez pas déjà fait :
   a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.
   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.
2. Utilisez les exemples suivants pour appeler l'opération `ListCollections`.

   Java

   L'exemple suivant dresse la liste des collections présentes dans la région actuelle.

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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
//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

Cette AWS CLI commande affiche la sortie JSON pour l'opération `list-collections` CLI. Remplacez la valeur de `profile_name` par le nom de votre profil de développeur.

```
aws rekognition list-collections --profile profile-name
```

Python

L'exemple suivant dresse la liste des collections présentes dans la région actuelle.

Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

L’exemple suivant dresse la liste des collections présentes dans la région actuelle.

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

Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
//Copyright 2018 Amazon.com, Inc. or its 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 demande d'opération

La valeur d’entrée de `ListCollections` est le nombre maximal de collections à renvoyer.

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

Si la réponse renvoie plus de collections que `MaxResults` n’en demande, vous recevez un jeton dont vous pouvez vous servir pour obtenir l’ensemble de résultats suivant dans un appel ultérieur à `ListCollections`. Par exemple :

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

```
}
```

ListCollections réponse à l'opération

Amazon Rekognition renvoie un tableau de collections (`CollectionIds`). Un tableau distinct (`FaceModelVersions`) indique la version du modèle facial utilisé pour analyser les visages dans chaque collection. Par exemple, dans la réponse JSON suivante, la collection `MyCollection` analyse les visages à l'aide de la version 2.0 du modèle facial. La collection `AnotherCollection` utilise la version 3.0 du modèle facial. Pour de plus amples informations, veuillez consulter Comprendre le versionnement des modèles.

`NextToken` est le jeton utilisé pour obtenir l'ensemble de résultats suivant dans un appel ultérieur à `ListCollections`.

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

Description d'une collection

Vous pouvez utiliser l'opération DescribeCollection pour obtenir les informations suivantes sur une collection :

- Le nombre de visages indexés dans la collection.
- La version du modèle utilisé avec la collection. Pour de plus amples informations, veuillez consulter the section called "Comprendre le versionnement des modèles".
- L'Amazon Resource Name (ARN) de la collection.
- La date et l'heure de création de la collection.

Pour décrire une collection (SDK)

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Utilisez les exemples suivants pour appeler l'opération `DescribeCollection`.

   Java

   Cet exemple décrit une collection.

   Remplacez la valeur `collectionId` par l'ID de la collection souhaitée.

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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l’exemple complet ici.

```
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

Cette AWS CLI commande affiche la sortie JSON pour l'opération `describe-collection` CLI. Remplacez la valeur de `collection-id` par l'ID de la collection souhaitée. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
aws rekognition describe-collection --collection-id collection-name --profile
 profile-name
```

Python

Cet exemple décrit une collection.

Remplacez la valeur `collection_id` par l'ID de la collection souhaitée. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

Cet exemple décrit une collection.

Remplacez la valeur `collectionId` par l'ID de la collection souhaitée.

```
//Copyright 2018 Amazon.com, Inc. or its 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 demande d'opération

La valeur d'entrée de `DescribeCollection` est l'ID de la collection souhaitée, comme l'illustre l'exemple JSON suivant.

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

DescribeCollection réponse à l'opération

La réponse inclut :

- Le nombre de visages indexés dans la collection, `FaceCount`.
- La version du modèle utilisé avec la collection, `FaceModelVersion`. Pour de plus amples informations, veuillez consulter the section called "Comprendre le versionnement des modèles".

- L'Amazon Resource Name (ARN) de la collection, `CollectionARN`.
- La date et l'heure de création de la collection, `CreationTimestamp`. La valeur de `CreationTimestamp` est le nombre de millisecondes depuis l'heure d'époque Unix jusqu'à la création de la collection. L'heure d'époque Unix est 00:00:00 UTC (temps universel coordonné), jeudi 1er janvier 1970. Pour plus d'informations, consultez Heure Unix.

```
{
    "CollectionARN": "arn:aws:rekognition:us-east-1:nnnnnnnnnnnn:collection/
MyCollection",
    "CreationTimestamp": 1.533422155042E9,
    "FaceCount": 200,
    "UserCount" : 20,
    "FaceModelVersion": "1.0"
}
```

Suppression d'une collection

Vous pouvez utiliser l'opération DeleteCollection pour supprimer une collection.

Pour de plus amples informations, veuillez consulter Gérer une collection.

Pour supprimer une collection (SDK)

1. Si vous ne l'avez pas déjà fait :
   a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.
   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.
2. Utilisez les exemples suivants pour appeler l'opération `DeleteCollection`.

   Java

   Cet exemple supprime une collection.

   Remplacez la valeur de `collectionId` par la collection que vous souhaitez supprimer.

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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
// 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

Cette AWS CLI commande affiche la sortie JSON pour l'opération `delete-collection` CLI. Remplacez la valeur de `collection-id` par le nom de la collection que vous souhaitez supprimer. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
aws rekognition delete-collection --collection-id collection-name --profile
 profile-name
```

Python

Cet exemple supprime une collection.

Remplacez la valeur de `collection_id` par la collection que vous souhaitez supprimer. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

Cet exemple supprime une collection.

Remplacez la valeur de `collectionId` par la collection que vous souhaitez supprimer.

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

Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
//Copyright 2018 Amazon.com, Inc. or its 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 demande d'opération

La valeur d'entrée de `DeleteCollection` est l'ID de la collection à supprimer, comme l'illustre l'exemple JSON suivant.

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

DeleteCollection réponse à l'opération

La réponse `DeleteCollection` contient un code de statut HTTP qui indique la réussite ou l'échec de l'opération. `200` est renvoyée si la collection est supprimée avec succès.

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

Ajout de visages à une collection

Vous pouvez utiliser l'opération IndexFaces pour détecter les visages sur une image et les ajouter à une collection. Pour chaque visage détecté, Amazon Rekognition extrait les caractéristiques faciales et stocke ces informations dans une base de données. Par ailleurs, la commande stocke les métadonnées de chaque visage détecté dans la collection de visages spécifiée. Amazon Rekognition ne stocke pas les octets d'image réels.

Pour en savoir plus sur la fourniture de visages adaptés pour l'indexation, consultez Recommandations pour les images d'entrée de comparaison faciale.

Pour chaque visage, l'opération `IndexFaces` conserve les informations suivantes :

- Caractéristiques faciales multidimensionnelles : `IndexFaces` utilise l'analyse du visage pour extraire des informations multidimensionnelles sur les traits du visage et stocke ces informations dans la collection de visages. Vous ne pouvez pas accéder directement à ces informations. Cependant, Amazon Rekognition utilise ces informations lors de la recherche de correspondances dans une collection de visages.

- Métadonnées : les métadonnées de chaque visage incluent un cadre de sélection, un niveau d'assurance (que la sélection contient un visage), les ID attribués par Amazon Rekognition (ID

de visage et ID d'image) et un ID d'image externe (si vous l'avez fourni) dans la demande. Ces informations vous sont renvoyées en réponse à l'appel de l'API `IndexFaces`. Pour obtenir un exemple, consultez l'élément `face` dans l'exemple de réponse suivant.

Le service renvoie les métadonnées en réponse aux appels d'API suivants :

- `ListFaces`
- Opérations de recherche de visages – Les réponses pour SearchFaces et SearchFacesByImage renvoient le degré de fiabilité de la correspondance de chaque visage, ainsi que les métadonnées du visage rapproché.

Le nombre de visages indexés par `IndexFaces` dépend de la version du modèle de détection de visages qui est associée à la collection d'entrée. Pour de plus amples informations, veuillez consulter Comprendre le versionnement des modèles.

Les informations sur les visages indexés sont renvoyées dans un tableau d'objets FaceRecord.

Vous pouvez associer des visages indexés à l'image sur laquelle ils ont été détectés. Par exemple, vous pouvez gérer un index côté client composé d'images et de visages présents sur celles-ci. Pour associer des visages à une image, spécifiez l'identifiant de l'image dans le paramètre de la demande `ExternalImageId`. L'ID de l'image peut être le nom de fichier ou un autre ID que vous créez.

En plus des informations précédentes que l'API conserve dans la collection de visages, l'API renvoie également des détails faciaux qui ne sont pas gardés dans la collection. (Consultez l'élément `faceDetail` dans l'exemple de réponse suivant).

> **Note**
>
> `DetectFaces` renvoie les mêmes informations, ce qui signifie que vous n'avez pas besoin d'appeler `DetectFaces` et `IndexFaces` pour la même image.

Filtrage des visages

L' IndexFaces opération permet de filtrer les visages indexés à partir d'une image. Avec `IndexFaces`, vous pouvez spécifier un nombre maximal de visages à indexer, ou vous pouvez choisir d'indexer uniquement les visages détectés avec une haute qualité.

Vous pouvez spécifier le nombre maximal de visages indexés par `IndexFaces` à l'aide du paramètre d'entrée `MaxFaces`. Cela s'avère utile lorsque vous souhaitez indexer les plus grands visages d'une image et pas les plus petits, comme ceux des personnes debout en arrière-plan.

Par défaut, choisit une barre de qualité `IndexFaces` utilisée pour filtrer les visages. Vous pouvez utiliser le paramètre d'entrée `QualityFilter` pour définir explicitement la barre de qualité. Les valeurs sont :

- `AUTO` : Amazon Rekognition choisit la barre de qualité utilisée pour filtrer les visages (valeur par défaut).
- `LOW` : tous les visages, à l'exception des visages de qualité inférieure, sont indexés.
- `MEDIUM`
- `HIGH` : seuls les visages de qualité supérieure sont indexés.
- `NONE` : aucun visage n'est filtré en fonction de la qualité.

`IndexFaces` filtre les visages pour les raisons suivantes :

- Le visage est trop petit par rapport aux dimensions de l'image.
- Le visage est trop flou.
- L'image est trop sombre.
- Le visage prend un pose extrême.
- Le visage n'est pas suffisamment détaillé pour la recherche de visage.

> **Note**
>
> Pour utiliser le filtrage de qualité, vous avez besoin d'une collection associée à la version 3, ou ultérieure, du modèle facial. Pour obtenir la version du modèle facial associé à une collection, appelez DescribeCollection.

Les informations sur les visages qui ne sont pas indexés par `IndexFaces` sont renvoyées dans un tableau d'objets UnindexedFace. Le tableau `Reasons` contient une liste des raisons pour lesquelles un visage n'est pas indexé. Par exemple, la valeur `EXCEEDS_MAX_FACES` correspond à un visage qui n'est pas indexé, car le nombre de visages spécifié par `MaxFaces` a déjà été détecté.

Pour de plus amples informations, veuillez consulter Gestion des visages dans une collection.

Pour ajouter des visages à une collection (SDK)

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur avec `AmazonRekognitionFullAccess` et autorisations `AmazonS3ReadOnlyAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Chargez une image (contenant un ou plusieurs visages) dans votre compartiment Amazon S3.

   Pour en savoir plus, consultez Téléchargement d'objets dans Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

3. Utilisez les exemples suivants pour appeler l'opération `IndexFaces`.

   Java

   Cet exemple affiche les identificateurs de visage pour les visages ajoutés à la collection.

   Remplacez la valeur de `collectionId` par le nom de la collection à laquelle vous souhaitez ajouter un visage. Remplacez la valeur de `bucket` et de `photo` par le nom du compartiment Amazon S3 et le nom de l'image utilisés à l'étape 2. Le paramètre `.withMaxFaces(1)` limite le nombre de visages indexés à 1. Supprimez ou modifiez sa valeur en fonction de vos besoins.

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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
//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

Cette AWS CLI commande affiche la sortie JSON pour l'opération `index-faces` CLI.

Remplacez la valeur de `collection-id` par le nom de la collection dans laquelle vous souhaitez stocker le visage. Remplacez la valeur de `Bucket` et de `Name` par le compartiment Amazon S3, et le fichier image utilisé à l'étape 2. Le paramètre `max-faces` limite le nombre de visages indexés à 1. Supprimez ou modifiez sa valeur en fonction de vos besoins. Remplacez la valeur de `profile_name` dans la ligne qui crée la session Rekognition par le nom de votre profil de développeur.

```
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
```

Si vous accédez à la CLI sur un périphérique Windows, utilisez des guillemets doubles au lieu de guillemets simples et évitez les guillemets doubles internes par une barre oblique inverse (c'est-à-dire \) pour corriger les erreurs d'analyse que vous pourriez rencontrer. Par exemple, consultez ce qui suit :

```
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

Cet exemple affiche les identificateurs de visage pour les visages ajoutés à la collection.

Remplacez la valeur de `collectionId` par le nom de la collection à laquelle vous souhaitez ajouter un visage. Remplacez la valeur de `bucket` et de `photo` par le nom du compartiment Amazon S3 et le nom de l'image utilisés à l'étape 2. Le paramètre d'entrée `MaxFaces` limite le nombre de visages indexés à 1. Supprimez ou modifiez sa valeur en fonction de vos besoins. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

Cet exemple affiche les identificateurs de visage pour les visages ajoutés à la collection.

Remplacez la valeur de `collectionId` par le nom de la collection à laquelle vous souhaitez ajouter un visage. Remplacez la valeur de `bucket` et de `photo` par le nom du compartiment Amazon S3 et le nom de l'image utilisés à l'étape 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 demande d'opération

La valeur d'entrée de `IndexFaces` est l'image à indexer et la collection à laquelle ajouter le ou les visages.

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "ExternalImageId": "input.jpg",
    "DetectionAttributes": [
        "DEFAULT"
    ],
    "MaxFaces": 1,
    "QualityFilter": "AUTO"
}
```

IndexFaces réponse à l'opération

`IndexFaces` renvoie des informations sur les visages détectés dans l'image. Par exemple, la réponse JSON ci-dessous comprend les attributs de détection par défaut pour les visages détectés dans l'image d'entrée. L'exemple montre également des visages non indexés parce que la valeur du paramètre d'entrée `MaxFaces` a été dépassée : le tableau `Reasons` contient EXCEEDS_MAX_FACES. Si un visage n'est pas indexé pour des raisons de qualité, `Reasons` contient des valeurs telles que LOW_SHARPNESS ou LOW_BRIGHTNESS. Pour de plus amples informations, veuillez consulter UnindexedFace.

```
{
    "FaceModelVersion": "3.0",
    "FaceRecords": [
        {
            "Face": {
```

```
                "BoundingBox": {
                    "Height": 0.3247932195663452,
                    "Left": 0.5055555701255798,
                    "Top": 0.2743072211742401,
                    "Width": 0.2144444358348846
                },
                "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": 0.2144444358348846
                },
                "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"
            ]
        }
    ]
}
```

Pour obtenir toutes les informations relatives aux visages, spécifiez « ALL » pour le paramètre `DetectionAttributes` de la demande. Ainsi, dans l'exemple de réponse suivant, notez la présence d'informations supplémentaires dans l'élément `faceDetail`, qui ne sont pas conservées sur le serveur :

- 25 repères faciaux (par rapport à seulement cinq dans l'exemple précédent)
- Dix attributs faciaux (lunettes, barbe, occlusion, direction du regard, etc.)
- Emotions (voir l'élément `emotion`)

L'élément `face` fournit des métadonnées qui sont conservées sur le serveur.

`FaceModelVersion` est la version du modèle facial associé à la collection. Pour de plus amples informations, veuillez consulter Comprendre le versionnement des modèles.

`OrientationCorrection` est l'orientation estimée de l'image. Les informations de correction de l'orientation ne sont pas renvoyées si vous utilisez une version du modèle de détection des

visages supérieure à la version 3. Pour de plus amples informations, veuillez consulter Obtention de l'orientation d'une image et des coordonnées du cadre de délimitation.

L'exemple de réponse suivant montre le JSON renvoyé lorsque vous spécifiez ["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": []
}
```

Répertorier les visages et les utilisateurs associés dans une collection

Vous pouvez utiliser cette ListFacesopération pour répertorier les visages et leurs utilisateurs associés dans une collection.

Pour de plus amples informations, veuillez consulter Gestion des visages dans une collection.

Pour dresser la liste des visages présents dans une collection (SDK)

1. Si vous ne l'avez pas déjà fait :
   a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.
   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.
2. Utilisez les exemples suivants pour appeler l'opération `ListFaces`.

   Java

   Cet exemple affiche la liste des visages présents dans une collection.

   Remplacez la valeur de `collectionId` par la collection souhaitée.

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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
// 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

Cette AWS CLI commande affiche la sortie JSON pour l'opération `list-faces` CLI. Remplacez la valeur de `collection-id` par le nom de la collection dont vous souhaitez dresser la liste. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
aws rekognition list-faces --collection-id "collection-id"  --profile profile-
name
```

Python

Cet exemple affiche la liste des visages présents dans une collection.

Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

Cet exemple affiche la liste des visages présents dans une collection.

Remplacez la valeur de `collectionId` par la collection souhaitée.

```
//Copyright 2018 Amazon.com, Inc. or its 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 demande d'opération

L'entrée `ListFaces` est l'ID de la collection pour laquelle vous souhaitez répertorier les visages. `MaxResultsest` le nombre maximum de faces à renvoyer. ListFaces prend également en compte une liste d'identifiants faciaux avec lesquels filtrer les résultats, et un identifiant utilisateur fourni pour répertorier uniquement les visages associés à l'utilisateur donné.

```
{
    "CollectionId": "MyCollection",
    "MaxResults": 1
}
```

Si la réponse renvoie plus de visages que `MaxResults` n’en demande, vous recevez un jeton dont vous pouvez vous servir pour obtenir l’ensemble de résultats suivant dans un appel ultérieur à `ListFaces`. Par exemple :

```
{
    "CollectionId": "MyCollection",
    "NextToken": "sm+5ythT3aeEVIR4WA....",
    "MaxResults": 1
}
```

ListFaces réponse à l'opération

La réponse de `ListFaces` correspond à des informations sur les métadonnées faciales stockées dans la collection spécifiée.

- FaceModelVersion— La version du modèle de visage associée à la collection. Pour de plus amples informations, veuillez consulter Comprendre le versionnement des modèles.
- Faces : informations sur les visages présents dans la collection. Il s'agit notamment d'informations concernant BoundingBox, la fiabilité, les identificateurs d'images et l'ID de visage. Pour de plus amples informations, consultez Face.
- NextToken— Le jeton utilisé pour obtenir la prochaine série de résultats.


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

Suppression de visages d'une collection

Vous pouvez utiliser l'opération DeleteFaces pour supprimer des visages d'une collection. Pour de plus amples informations, veuillez consulter Gestion des visages dans une collection.

Pour supprimer des visages d'une collection

1. Si vous ne l'avez pas déjà fait :
   a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.
   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.
2. Utilisez les exemples suivants pour appeler l'opération `DeleteFaces`.

   Java

   Dans cet exemple, seul un visage est supprimé d'une collection.

   Remplacez la valeur de `collectionId` par la collection qui contient le visage que vous souhaitez supprimer. Remplacez la valeur de `faces` par l'ID du visage que vous souhaitez supprimer. Pour supprimer plusieurs visages, ajoutez les ID de visage au tableau `faces`.

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

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
```

```
import com.amazonaws.services.rekognition.model.DeleteFacesRequest;
import com.amazonaws.services.rekognition.model.DeleteFacesResult;

import java.util.List;


public class DeleteFacesFromCollection {
   public static final String collectionId = "MyCollection";
   public static final String faces[] = {"xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx"};

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

      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();


      DeleteFacesRequest deleteFacesRequest = new DeleteFacesRequest()
              .withCollectionId(collectionId)
              .withFaceIds(faces);

      DeleteFacesResult
 deleteFacesResult=rekognitionClient.deleteFaces(deleteFacesRequest);


      List < String > faceRecords = deleteFacesResult.getDeletedFaces();
      System.out.println(Integer.toString(faceRecords.size()) + " face(s)
 deleted:");
      for (String face: faceRecords) {
         System.out.println("FaceID: " + face);
      }
   }
}
```

Java V2

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
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

Cette AWS CLI commande affiche la sortie JSON pour l'opération `delete-faces` CLI. Remplacez la valeur de `collection-id` par le nom de la collection qui contient le visage que vous souhaitez supprimer. Remplacez la valeur de `face-ids` par le tableau des ID de visages que vous souhaitez supprimer. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
aws rekognition delete-faces --collection-id "collection-id" --face-ids "faceid"
 --profile profile-name
```

Python

Dans cet exemple, seul un visage est supprimé d'une collection.

Remplacez la valeur de `collectionId` par la collection qui contient le visage que vous souhaitez supprimer. Remplacez la valeur de `faces` par l'ID du visage que vous souhaitez supprimer. Pour supprimer plusieurs visages, ajoutez les ID de visage au tableau `faces`. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

Dans cet exemple, seul un visage est supprimé d'une collection.

Remplacez la valeur de `collectionId` par la collection qui contient le visage que vous souhaitez supprimer. Remplacez la valeur de `faces` par l'ID du visage que vous souhaitez supprimer. Pour supprimer plusieurs visages, ajoutez les ID de visage à la liste `faces`.

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

using System;
using System.Collections.Generic;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class DeleteFaces
{
    public static void Example()
    {
        String collectionId = "MyCollection";
        List<String> faces = new List<String>() { "xxxxxxxx-xxxx-xxxx-xxxx-
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 demande d'opération

La valeur d'entrée de `DeleteFaces` est l'ID de la collection qui contient les visages, ainsi qu'un tableau d'ID correspondant aux visages à supprimer.

```
{
    "CollectionId": "MyCollection",
    "FaceIds": [
        "daf29cac-f910-41e9-851f-6eeb0e08f973"
    ]
}
```

DeleteFaces réponse à l'opération

La réponse `DeleteFaces` contient un tableau d'ID correspondant aux visages qui ont été supprimés.

```
{
    "DeletedFaces": [
        "daf29cac-f910-41e9-851f-6eeb0e08f973"
    ]
}
```

Si les identifiants faciaux fournis dans la saisie sont actuellement associés à un utilisateur, ils seront renvoyés UnsuccessfulFaceDeletions avec des raisons valables.

```
{
   "DeletedFaces": [
        "daf29cac-f910-41e9-851f-6eeb0e08f973"
    ],
     "UnsuccessfulFaceDeletions" : [
        {
           "FaceId" : "0b683aed-a0f1-48b2-9b5e-139e9cc2a757",
           "UserId" : "demoUser1",
           "Reason" : ["ASSOCIATED_TO_AN_EXISTING_USER"]
        }
     ]
}
```

Création d'un utilisateur

Vous pouvez utiliser cette CreateUseropération pour créer un nouvel utilisateur dans une collection à l'aide d'un ID utilisateur unique que vous fournissez. Vous pouvez ensuite associer plusieurs visages à l'utilisateur nouvellement créé.

Pour créer un utilisateur (SDK)

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un compte utilisateur IAM; avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Utilisez les exemples suivants pour appeler l'opération `CreateUser`.

   Java

   Cet exemple de code Java crée un utilisateur.

```
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

Cette AWS CLI commande crée un utilisateur à l'aide de l'opération `create-user` CLI.

```
aws rekognition create-user --user-id user-id --collection-id collection-name --
region region-name
--client-request-token request-token
```

Python

Cet exemple de code Python crée un utilisateur.

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

Cet exemple de code Go utilise le AWS Go SDK V2 et crée un utilisateur.

```
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")

}
```

Suppression d’un utilisateur

Vous pouvez utiliser cette DeleteUseropération pour supprimer un utilisateur d'une collection, en fonction de l'userID fourni. Notez que toutes les faces associées à l’UserID sont dissociées de l’UserID avant que l’UserID spécifié ne soit supprimé.

Pour supprimer un utilisateur (SDK)

1. Si vous ne l’avez pas déjà fait :

a. Créez ou mettez à jour un compte utilisateur IAM; avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Utilisez les exemples suivants pour appeler l'opération `DeleteUser`.

Java

Cet exemple de code Java crée un utilisateur.

```
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

Cette AWS CLI commande supprime un utilisateur à l'aide de l'opération `create-user` CLI.

```
aws rekognition delete-user --collection-id MyCollection
--user-id user-id --collection-id collection-name --region region-name
```

Python

Cet exemple de code Python crée un utilisateur.

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

Associer des visages à un utilisateur

Vous pouvez utiliser cette AssociateFacesopération pour associer plusieurs visages individuels à un seul utilisateur. Pour associer un visage à un utilisateur, vous devez d'abord créer une collection et un utilisateur. Notez que les vecteurs de visage doivent résider dans la même collection que le vecteur utilisateur.

Pour associer des visages (SDK)

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Utilisez les exemples suivants pour appeler l'opération `AssociateFaces`.

   Java

   Cet exemple de code Java associe un visage à un utilisateur.

```
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

Cette AWS CLI commande associe un visage à un utilisateur à l'aide de l'opération `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

Cet exemple de code Python associe un visage à un utilisateur.

```
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 réponse à l'opération

La réponse pour `AssociateFaces` inclut le `UserStatus`, qui est le statut de la demande de dissociation, ainsi qu'une liste des personnes `FaceIds` à associer. Une liste de `UnsuccessfulFaceAssociations` est également renvoyée. Après avoir soumis une demande à `AssociateFaces`, l'opération peut prendre environ une minute.

Pour cette raison, le UserStatus est renvoyé, qui peut prendre les valeurs suivantes :

- CRÉÉ : indique que « l'utilisateur » a été créé avec succès et qu'aucun visage ne lui est associé actuellement. « Utilisateur » sera dans cet état avant qu'un appel « AssociateFaces » réussi ne soit effectué.
- MISE À JOUR - Indique que « l'utilisateur » est mis à jour pour refléter les nouveaux associated/disassociated visages et qu'il deviendra ACTIF dans quelques secondes. Les résultats de recherche peuvent contenir « Utilisateur » dans cet état et les clients peuvent choisir de l'ignorer dans les résultats renvoyés.
- ACTIF : indique que le nom « utilisateur » est mis à jour pour refléter tous les associated/disassociated visages et qu'il est dans un état consultable.

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

Dissocier des visages d’un utilisateur

Vous pouvez utiliser cette DisassociateFacesopération pour supprimer l'association entre un identifiant utilisateur et un identifiant facial.

Pour associer des visages (SDK)

1. Si vous ne l’avez pas déjà fait :

    a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

    b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Utilisez les exemples suivants pour appeler l’opération `DisassociateFaces`.

Java

Cet exemple Java supprime l'association entre un FaceID et un UserID avec l'opération `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

Cette AWS CLI commande supprime l'association entre un FaceID et un UserID lors de l'opération. `DisassociateFaces`

```
aws rekognition disassociate-faces --face-ids list-of-face-ids
--user-id user-id --collection-id collection-name --region region-name
```

Python

L'exemple suivant supprime l'association entre un FaceID et un UserID avec l'opération `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 réponse à l'opération

La réponse pour `DisassociateFaces` inclut le `UserStatus`, qui est le statut de la demande de dissociation, ainsi qu’une liste des `FaceIds` à dissocier. Une liste de `UnsuccessfulFaceDisassociations` est également renvoyée. Après avoir soumis une demande à DisassociateFaces, l'opération peut prendre environ une minute. Pour cette raison, le UserStatus est renvoyé, qui peut prendre les valeurs suivantes :

- CRÉÉ : indique que « l’utilisateur » a été créé avec succès et qu’aucun visage ne lui est associé actuellement. « Utilisateur » sera dans cet état avant qu'un appel « AssociateFaces » réussi ne soit effectué.

- MISE À JOUR - Indique que « l'utilisateur » est mis à jour pour refléter les nouveaux associated/disassociated visages et qu'il deviendra ACTIF dans quelques secondes. Les résultats de recherche peuvent contenir « Utilisateur » dans cet état et les clients peuvent choisir de l'ignorer dans les résultats renvoyés.
- ACTIF : indique que le nom « utilisateur » est mis à jour pour refléter tous les associated/disassociated visages et qu'il est dans un état consultable.

```
{
    "UserStatus": "UPDATING",
    "DisassociatedFaces": [
        {
            "FaceId": "c92265d4-5f9c-43af-a58e-12be0ce02bc3"
        }
    ],
    "UnsuccessfulFaceDisassociations": [
        {
            "Reasons": [
                "ASSOCIATED_TO_A_DIFFERENT_IDENTITY"
            ],
            "FaceId": "f5817d37-94f6-4335-bfee-6cf79a3d806e",
            "UserId": "demoUser1"
        }
    ]
}
```

Création d'une liste de visages d'une collection

Vous pouvez utiliser l'ListUsersopération pour répertorier UserIds et le UserStatus. Pour voir les FaceID associés à un UserID, utilisez l'ListFacesopération.

Pour répertorier les utilisateurs (SDK)

1. Si vous ne l'avez pas déjà fait :
    a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.
    b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Utilisez les exemples suivants pour appeler l'opération `ListUsers`.

   Java

   Cet exemple Java répertorie les utilisateurs d'une collection utilisant l'opération `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

Cette AWS CLI commande répertorie les utilisateurs d'une collection avec l'`ListUsers`opération.

```
aws rekognition list-users --collection-id collection-id --max-results number-
of-max-results
```

Python

L'exemple suivant répertorie les utilisateurs d'une collection avec l'opération `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 réponse à l'opération

La réponse à une demande d' ListUsers inclusion d'une liste `Users` des éléments de la collection ainsi que `UserStatus` de l'utilisateur. `UsedId`


```
{
    "NextToken": "B1asJT3bAb/ttuGgPFV8BZoBZyGQzlUHXbuTNLh48a6enU7kXKw43hpOwizW7LOk/
Gk7Em09lznoq6+FcDCcSq2olrn7A98BLkt5keu+ZRVRUTyrXtT6J7Hmp
+ieQ2an6Zu0qzPfcdPeaJ9eAxG2d0WNrzJgi5hvmjoiSTTfKX3MQz1sduWQkvAAs4hZfhZoKFahFlqWofshCXa/
FHAAY3PL1PjxXbkNeSSMq8V7i1MlKCdrPVykCv9MokpPt7jtNvKPEZGUhxgBTFMxNWLEcFnzAiCWDg91dFy/
LalshPjXA9UVc5Gx9vIJNQ/
eO3cQRghAkCT3FOAiXsLAnA015ODTomZpWWVpqB21wKpI3LYmfAVFrDPGzpbTVlRmLsJm41bkmnBBBw9+DHz1Jn7zW
+qc5Fs3yaHu0f51Xg==",
    "Users": [
        {
            "UserId": "demoUser4",
            "UserStatus": "CREATED"
        },
        {
            "UserId": "demoUser2",
            "UserStatus": "CREATED"
        }
    ]
}
```

Recherche d'un visage avec un ID de visage

Vous pouvez utiliser cette SearchFacesopération pour rechercher des utilisateurs dans une collection qui correspondent au plus grand visage d'une image fournie.

L'ID de visage est renvoyé dans la réponse de l'opération IndexFaces lorsque le visage est détecté et ajouté à une collection. Pour de plus amples informations, veuillez consulter Gestion des visages dans une collection.

Pour rechercher un visage dans une collection à l'aide de son ID (SDK)

1. Si vous ne l'avez pas déjà fait :
   a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.
   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.
2. Utilisez les exemples suivants pour appeler l'opération `SearchFaces`.

   Java

   Cet exemple affiche des informations sur les visages qui correspondent à un visage identifié par son ID.

   Remplacez la valeur de `collectionID` par la collection qui contient le visage en question. Remplacez la valeur de `faceId` par l'identificateur du visage que vous souhaitez rechercher.

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

}
```

Exécutez l’exemple de code. Des informations sur les visages correspondants s’affichent à l’écran.

Java V2

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
// 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

Cette AWS CLI commande affiche la sortie JSON pour l'opération `search-faces` CLI. Remplacez la valeur d'`face-id` par l'identifiant de visage à rechercher et remplacez la valeur de `collection-id` par la collection dans laquelle vous souhaitez effectuer la recherche. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
aws rekognition search-faces --face-id face-id --collection-id "collection-id"
 --profile profile-name
```

Python

Cet exemple affiche des informations sur les visages qui correspondent à un visage identifié par son ID.

Remplacez la valeur de `collectionID` par la collection qui contient le visage en question. Remplacez la valeur de `faceId` par l'identificateur du visage que vous souhaitez rechercher. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

Cet exemple affiche des informations sur les visages qui correspondent à un visage identifié par son ID.

Remplacez la valeur de `collectionID` par la collection qui contient le visage en question. Remplacez la valeur de `faceId` par l'identificateur du visage que vous souhaitez rechercher.

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

Exécutez l’exemple de code. Des informations sur les visages correspondants s’affichent à l’écran.

SearchFaces demande d'opération

A partir d’un ID de visage donné (chaque visage stocké dans la collection de visages possède un ID de visage), `SearchFaces` recherche dans la collection de visages spécifiée des visages similaires. La réponse n’inclut pas le visage que vous recherchez. Elle ne comprend que des visages similaires. Par défaut, `SearchFaces` renvoie les visages pour lesquels l’algorithme détecte une similarité de plus de 80 %. La similarité indique à quel point le visage correspond au visage en entrée. Le cas échéant, vous pouvez l’utiliser `FaceMatchThreshold` pour spécifier une autre valeur.

```
{
    "CollectionId": "MyCollection",
    "FaceId": "0b683aed-a0f1-48b2-9b5e-139e9cc2a757",
    "MaxFaces": 2,
```

```
    "FaceMatchThreshold": 99
}
```

SearchFaces réponse à l'opération

L'opération renvoie un tableau des correspondances de visage qui ont été trouvées, ainsi que l'ID de visage que vous avez fourni en entrée.

```
{
    "SearchedFaceId": "7ecf8c19-5274-5917-9c91-1db9ae0449e2",
    "FaceMatches": [ list of face matches found ]
}
```

Pour chaque correspondance de visage trouvée, la réponse inclut la similarité et les métadonnées faciales, comme l'illustre l'exemple de réponse suivant :

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

Recherche d'un visage à l'aide d'une image

Vous pouvez utiliser l'opération SearchFacesByImage afin de rechercher des visages correspondant au visage le plus grand d'une image fournie.

Pour de plus amples informations, veuillez consulter Recherche de visages et d'utilisateurs dans une collection.

Pour rechercher un visage dans une collection à l'aide d'une image (SDK)

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur avec `AmazonRekognitionFullAccess` et autorisations `AmazonS3ReadOnlyAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Chargez une image (qui contient un ou plusieurs visages) dans votre compartiment S3.

   Pour en savoir plus, consultez Chargement d'objets dans Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

3. Utilisez les exemples suivants pour appeler l'opération `SearchFacesByImage`.

   Java

   Cet exemple affiche des informations sur les visages qui correspondent au visage le plus grand dans une image. L'exemple de code spécifie les paramètres `FaceMatchThreshold` et `MaxFaces` pour limiter les résultats renvoyés dans la réponse.

   Dans l'exemple suivant, modifiez les éléments suivants : remplacez la valeur de `collectionId` par la collection à explorer, remplacez la valeur de `bucket` par le

compartiment contenant l'image d'entrée, puis remplacez la valeur de `photo` par l'image d'entrée.

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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
// 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

Cette AWS CLI commande affiche la sortie JSON pour l'opération `search-faces-by-image` CLI. Remplacez la valeur de `Bucket` par le compartiment S3 utilisé à l'étape 2. Remplacez la valeur de `Name` par le nom de fichier image utilisé à l'étape 2. Remplacez la valeur de `collection-id` par la collection dans laquelle vous souhaitez effectuer la recherche. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
aws rekognition search-faces-by-image --image '{"S3Object":{"Bucket":"bucket-
name","Name":"image-name"}}' \
--collection-id "collection-id" --profile profile-name
```

Si vous accédez à la CLI sur un périphérique Windows, utilisez des guillemets doubles au lieu de guillemets simples et évitez les guillemets doubles internes par une barre oblique inverse (c'est-à-dire \) pour corriger les erreurs d'analyse que vous pourriez rencontrer. Par exemple, consultez ce qui suit :

```
aws rekognition search-faces-by-image --image "{\"S3Object\":{\"Bucket\":
\"bucket-name\",\"Name\":\"image-name\"}}" \
--collection-id "collection-id" --profile profile-name
```

Python

Cet exemple affiche des informations sur les visages qui correspondent au visage le plus grand dans une image. L'exemple de code spécifie les paramètres `FaceMatchThreshold` et `MaxFaces` pour limiter les résultats renvoyés dans la réponse.

Dans l'exemple suivant, modifiez les éléments suivants : remplacez la valeur de `collectionId` par la collection à explorer, puis remplacez la valeur de `bucket` et de `photo` par le nom du compartiment S3 et le nom de l'image utilisés à l'étape 2. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

Cet exemple affiche des informations sur les visages qui correspondent au visage le plus grand dans une image. L'exemple de code spécifie les paramètres `FaceMatchThreshold` et `MaxFaces` pour limiter les résultats renvoyés dans la réponse.

Dans l'exemple suivant, modifiez les éléments suivants : remplacez la valeur de `collectionId` par la collection à explorer, puis remplacez la valeur de `bucket` et de `photo` par le nom du compartiment S3 et le nom de l'image utilisés à l'étape 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 demande d'opération

Les paramètres d’entrée de `SearchFacesImageByImage` correspondent à la collection à explorer et à l’emplacement de l’image source. Dans cet exemple, l’image source est stockée dans un compartiment Amazon S3 (`S3Object`). Sont également spécifiés le nombre maximal de visages à

renvoyer (`Maxfaces`) et le niveau de fiabilité minimal auquel doit correspondre un visage pour être renvoyé (`FaceMatchThreshold`).

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "MaxFaces": 2,
    "FaceMatchThreshold": 99
}
```

SearchFacesByImage réponse à l'opération

A partir d'une image d'entrée donnée (.jpeg ou .png), l'opération détecte dans un premier temps le visage dans l'image d'entrée, puis recherche dans la collection de visages spécifiée des visages similaires.

> **Note**
>
> Si le service détecte plusieurs visages dans l'image d'entrée, il utilise le visage détecté le plus grand pour explorer la collection de visages.

L'opération renvoie un tableau des correspondances de visage qui ont été trouvées, ainsi que des informations sur le visage d'entrée. Ces informations incluent notamment le cadre de délimitation, ainsi que la valeur de fiabilité, qui indique le niveau de certitude que le cadre de délimitation contient un visage.

Par défaut, `SearchFacesByImage` renvoie les visages pour lesquels l'algorithme détecte une similarité de plus de 80 %. La similarité indique à quel point le visage correspond au visage en entrée. Le cas échéant, vous pouvez l'utiliser `FaceMatchThreshold` pour spécifier une autre valeur. Pour chaque correspondance de visage trouvée, la réponse inclut la similarité et les métadonnées faciales, comme l'illustre l'exemple de réponse suivant :

```
{
```

```
    "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
}
```


Recherche d’utilisateurs (identifiant facial/ID utilisateur)

Vous pouvez utiliser cette SearchUsersopération pour rechercher des utilisateurs dans une collection spécifiée qui correspondent à un identifiant de visage ou à un identifiant d'utilisateur fourni. L'opération répertorie les résultats `UserIds` classés selon le score de similarité le plus élevé au-dessus du score demandé UserMatchThreshold. L'ID utilisateur est créé lors de l' CreateUsers opération. Pour de plus amples informations, veuillez consulter Gérer les utilisateurs dans une collection.

Pour rechercher des utilisateurs (SDK)

1. Si vous ne l’avez pas déjà fait :

a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Utilisez les exemples suivants pour appeler l'opération `SearchUsers`.

Java

Cet exemple Java recherche les utilisateurs d'une collection à l'aide de l'opération `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

Cette AWS CLI commande recherche les utilisateurs d'une collection avec l'`SearchUsers`opération.

```
aws rekognition search-users --face-id face-id --collection-id collection-id --
region region-name
```

Python

L'exemple suivant recherche les utilisateurs d'une collection avec l'opération `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 demande d'opération

À partir d'un FaceID ou d'un UserID SearchUsers , recherche les correspondances entre utilisateurs dans le CollectionId spécifié. Par défaut, SearchUsers renvoie les UserIds pour lesquels le score de similarité est supérieur à 80 %. La similitude indique dans quelle mesure l'UserID correspond au FaceID ou UserID fourni. Si plusieurs UserID sont renvoyés, ils sont listés par ordre du score de similarité du plus élevé au plus faible. Vous pouvez éventuellement utiliser le UserMatchThreshold pour spécifier une valeur différente. Pour de plus amples informations, veuillez consulter Gérer les utilisateurs dans une collection.

Voici un exemple de SearchUsers demande utilisant `UserId` :

```
{
   "CollectionId": "MyCollection",
   "UserId": "demoUser1",
```

```
    "MaxUsers": 2,
    "UserMatchThreshold": 99
}
```

Voici un exemple de SearchUsers demande utilisant `FaceId` :

```
{
    "CollectionId": "MyCollection",
    "FaceId": "bff43c40-cfa7-4b94-bed8-8a08b2205107",
    "MaxUsers": 2,
    "UserMatchThreshold": 99
}
```

SearchUsers réponse à l'opération

Si vous effectuez une recherche avec un`FaceId`, la réponse SearchUsers inclut le `FaceId` pour`SearchedFace`, ainsi qu'une liste de `UserMatches` `UserId` et `UserStatus` pour chaque utilisateur.

```
{
    "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"
}
```

Si vous recherchez avec a`UserId`, la réponse pour SearchUsers inclut le `UserId` pour`SearchedUser`, en plus des autres éléments de réponse.

```
{
    "SearchedUser": {
        "UserId": "demoUser1"
    },
    "UserMatches": [
        {
            "User": {
                "UserId": "demoUser2",
                "UserStatus": "ACTIVE"
            },
            "Similarity": 99.97946166992188
        }
    ],
    "FaceModelVersion": "6"
}
```

Recherche d’utilisateurs (image)

`SearchUsersByImage` recherche l’ID de collection spécifié pour les utilisateurs dans une collection qui correspond au plus grand visage détecté dans une image fournie. Par défaut, SearchUsersByImage renvoie les UserIds pour lesquels le score de similarité est supérieur à 80 %. La similarité indique dans quelle mesure l’UserID correspond au plus grand visage détecté dans l’image fournie. Si plusieurs UserID sont renvoyés, ils sont listés par ordre du score de similarité du plus élevé au plus faible. Vous pouvez éventuellement utiliser le UserMatchThreshold pour spécifier une valeur différente. Pour de plus amples informations, veuillez consulter the section called “Gérer les utilisateurs dans une collection”.

Pour rechercher des utilisateurs par image (SDK)

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Utilisez les exemples suivants pour appeler l'opération `SearchUsersByImage`.

   Java

   Cet exemple Java recherche les utilisateurs d'une collection sur la base d'une image en entrée, à l'aide de l'opération `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

Cette AWS CLI commande recherche les utilisateurs d'une collection en fonction d'une image d'entrée, avec l'`SearchUsersByImage`opération.

```
aws rekognition search-users-by-image --image '{"S3Object":
{"Bucket":"s3BucketName","Name":"file-name"}}' --collection-id MyCollectionId --
region region-name
```

Python

L'exemple suivant recherche les utilisateurs d'une collection sur la base d'une image d'entrée, avec l'opération `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 demande d'opération

Les paramètres d'entrée de `SearchUsersByImage` correspondent à la collection à explorer, et à l'emplacement de l'image source. Dans cet exemple, l'image source est stockée dans un compartiment Amazon S3 (`S3Object`). Sont également spécifiés le nombre maximal de visages à renvoyer (`MaxUsers`) et le niveau de fiabilité minimal auquel doit correspondre un visage pour être renvoyé (`UserMatchThreshold`).

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "MaxUsers": 2,
    "UserMatchThreshold": 99
}
```

SearchUsersByImage réponse à l'opération

La réponse pour `SearchUsersByImage` inclut un `FaceDetail` objet pour le`SearchedFace`, ainsi qu'une liste UserMatches contenant le `UserIdSimilarity`, et `UserStatus` pour chacun d'entre eux. Si l'image d'entrée contient plusieurs visages, une liste de visages UnsearchedFaces sera également renvoyée.

```
{
```

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

Recherche de visages dans des vidéos stockées

Vous pouvez rechercher dans une collection des visages qui correspondent à ceux des personnes détectées dans une vidéo stockée ou en streaming. Cette section couvre la recherche de visages dans une vidéo stockée. Pour de plus amples informations sur la recherche de visages dans une vidéo en streaming, consultez Utilisation des vidéos streaming.

Les visages que vous recherchez doivent d'abord être indexés dans une collection en utilisant IndexFaces. Pour de plus amples informations, veuillez consulter Ajout de visages à une collection.

La recherche de visages Vidéo Amazon Rekognition suit le même flux de travail asynchrone que les autres opérations de Vidéo Amazon Rekognition qui analysent les vidéos stockées dans un compartiment Amazon S3. Pour commencer à rechercher des visages dans une vidéo enregistrée, appelez StartFaceSearchet fournissez l'identifiant de la collection que vous souhaitez rechercher. Vidéo Amazon Rekognition publie l'état d'achèvement de l'analyse vidéo dans une rubrique Amazon Simple Notification Service (Amazon SNS). Si l'analyse vidéo aboutit, appelez GetFaceSearch pour obtenir les résultats de recherche. Pour de plus amples informations sur le démarrage de l'analyse vidéo et l'obtention des résultats, consultez Appeler les opérations de Vidéo Amazon Rekognition.

La procédure suivante montre comment rechercher dans une collection des visages qui correspondent à ceux des personnes détectées dans une vidéo. La procédure indique également comment obtenir les données de suivi des personnes présentant une correspondance dans la vidéo. La procédure s'appuie sur le code figurant dans Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK), qui utilise une file d'attente Amazon Simple Queue Service (Amazon SQS) pour obtenir le statut d'achèvement d'une demande d'analyse vidéo.

Pour rechercher dans une vidéo des correspondances de visage (SDK)

1. Créez une collection.
2. Indexez un visage dans la collection.
3. Effectuez une Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK).
4. Ajoutez le code suivant à la classe `VideoDetect` que vous avez créée à l'étape 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);

    }
```

Dans la fonction `main`, remplacez les lignes :

```
        StartLabelDetection(bucket, video);

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

avec :

```
        String collection="collection";
        StartFaceSearchCollection(bucket, video, collection);

        if (GetSQSMessageSuccess()==true)
         GetFaceSearchCollectionResults();
```

Java V2

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
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()
```

Dans la fonction `main`, remplacez les lignes :

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

avec :

```
    collection='tests'
    analyzer.StartFaceSearchCollection(collection)

    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetFaceSearchCollectionResults()
```

Si vous avez déjà exécuté un exemple vidéo autre que Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK), le code à remplacer peut être différent.

5. Remplacez la valeur de `collection` par le nom de la collection que vous avez créée à l'étape 1.
6. Exécutez le code. Une liste des personnes présentes dans la vidéo dont les visages correspondent à ceux de la collection d'entrée s'affiche à l'écran. Les données de suivi pour chaque personne faisant l'objet d'une correspondance sont également affichées.

GetFaceSearch réponse à l'opération

Vous trouverez ci-dessous un exemple de réponse JSON de `GetFaceSearch`.

La réponse inclut un tableau des personnes (`Persons`) détectées dans la vidéo parce que leur visage correspondait à un visage de la collection d'entrée. Un élément de liste, PersonMatch, existe pour chaque occurrence de détection d'une personne dans la vidéo. Chaque `PersonMatch` comprend une liste de correspondances de visages provenant de la collection d'entrée, FaceMatch, des informations sur la personne correspondante, PersonDetail, et l'heure où la personne a été détectée comme étant une correspondance dans la vidéo.

```
{
    "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
    }
}
```

Recherche de visages dans une collection en vidéo streaming

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vous pouvez utiliser Vidéo Amazon Rekognition pour détecter et reconnaître des visages dans une collection de vidéos diffusées en streaming. Avec Amazon Rekognition Video, vous pouvez créer un processeur de flux CreateStreamProcessor() pour démarrer et gérer l'analyse du streaming vidéo.

Pour détecter un visage connu dans un flux vidéo (recherche faciale), Vidéo Amazon Rekognition utilise Amazon Kinesis Video Streams pour recevoir et traiter un flux vidéo. Les résultats d'analyse représentent une sortie de Vidéo Amazon Rekognition vers un flux de données Kinesis, ensuite lue par votre application cliente.

Pour utiliser Vidéo Amazon Rekognition avec une vidéo en streaming, votre application doit implémenter les éléments suivants :

- Un flux vidéo Kinesis pour envoyer des vidéos en streaming à Vidéo Amazon Rekognition. Pour plus d'informations, consultez le Guide du développeur Amazon Kinesis Video Streams.
- Un processeur de flux Vidéo Amazon Rekognition pour gérer l'analyse de la vidéo en streaming. Pour de plus amples informations, veuillez consulter Présentation des opérations du processeur de flux Vidéo Amazon Rekognition.

- Un consommateur de flux de données Kinesis chargé de lire les résultats d'analyse que Vidéo Amazon Rekognition envoie au flux de données Kinesis. Pour plus d'informations, consultez Consommateurs de flux de données Kinesis.

Cette section contient des informations sur la création d'une application qui crée le flux vidéo Kinesis et les autres ressources nécessaires, diffuse des vidéos sur Vidéo Amazon Rekognition et reçoit les résultats de l'analyse.

Rubriques

- Configuration de vos ressources Vidéo Amazon Rekognition et Amazon Kinesis
- Recherche de visages dans une vidéo en streaming
- Streaming à l'aide d'un plugin GStreamer
- Résolution des problèmes de streaming vidéo

Configuration de vos ressources Vidéo Amazon Rekognition et Amazon Kinesis

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Les procédures suivantes décrivent les étapes à suivre pour approvisionner le flux vidéo Kinesis et les autres ressources utilisées pour reconnaître les visages dans une vidéo en streaming.

Conditions préalables

Pour exécuter cette procédure, vous devez l'avoir AWS SDK pour Java installé. Pour de plus amples informations, veuillez consulter Premiers pas avec Amazon Rekognition. L'utilisateur Compte AWS que vous utilisez doit disposer d'autorisations d'accès à l'API Amazon Rekognition. Pour de plus amples informations, veuillez consulter Actions définies par Amazon Rekognition dans le Guide de l'utilisateur IAM.

Pour reconnaître des visages dans un flux vidéo (AWS SDK)

1. Si ce n'est pas déjà fait, créez une fonction du service IAM pour permettre à Vidéo Amazon Rekognition d'accéder à vos flux vidéo Kinesis et à vos flux de données Kinesis. Notez l'ARN. Pour de plus amples informations, veuillez consulter Donner accès aux streams à l'aide de AmazonRekognitionServiceRole.
2. Créez une collection et notez l'identifiant de collection que vous avez utilisé.
3. Indexez les visages à rechercher dans la collection que vous avez créée lors de l'étape 2.
4. Créez un flux vidéo stream et notez l'Amazon Resource Name (ARN) du flux.
5. Créer un flux de données Kinesis. Ajoutez le nom du flux au début AmazonRekognitionet notez l'ARN du flux.

Vous pouvez ensuite créer le processeur de flux de recherche faciale et démarrer le processeur de flux en utilisant le nom de processeur de flux que vous avez choisi.

> **Note**
>
> Vous ne devez démarrer le processeur de diffusion qu'après avoir vérifié que vous pouvez intégrer du contenu multimédia dans le flux vidéo Kinesis.

Diffusion de vidéos sur Vidéo Amazon Rekognition

Pour diffuser une vidéo dans Vidéo Amazon Rekognition, vous devez utiliser le SDK Amazon Kinesis Video Streams pour créer et utiliser un flux vidéo Kinesis. L'opération `PutMedia` écrit des fragments de données vidéo dans un flux vidéo utilisé par Vidéo Amazon Rekognition. Chaque fragment de donnée vidéo dure environ 2–10 secondes et contient une trame d'images vidéo autonome. Amazon Rekognition Video H.264 prend en charge les vidéos codées, qui peuvent comporter trois types d'images (I, B et P). Pour en savoir plus, consultez Inter Frame. Le premier cadre du fragment doit être un I-frame. Une image I-frame peut être décodée indépendamment de toute autre trame.

Au fur et à mesure que les données arrivent dans le flux vidéo Kinesis, Kinesis Video Streams attribue un numéro unique au fragment. Pour un exemple, voir Exemple PutMedia d'API.

- Si vous diffusez à partir d'une source codée en Matroska (MKV), utilisez cette PutMediaopération pour diffuser la vidéo source dans le flux vidéo Kinesis que vous avez créé. Pour plus d'informations, consultez PutMedia l'exemple d'API.

- Si vous diffusez à partir de la caméra d'un appareil, consultez Streaming à l'aide d'un plugin GStreamer.

Octroi à Vidéo Amazon Rekognition d'un accès à vos ressources

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vous utilisez un rôle de service Gestion des identités et des accès AWS (IAM) pour donner à Amazon Rekognition Video un accès en lecture aux flux vidéo Kinesis. Si vous utilisez un processeur de flux de recherche faciale, vous utilisez une fonction du service IAM pour autoriser Vidéo Amazon Rekognition à accéder en écriture aux flux de données Kinesis. Si vous utilisez un processeur de flux de surveillance de la sécurité, vous utilisez des rôles IAM pour permettre à Vidéo Amazon Rekognition d'accéder à votre compartiment Amazon S3 et à une rubrique Amazon SNS.

Octroi d'un accès aux processeurs de flux de recherche faciale

Vous pouvez créer une politique d'autorisation qui autorise Vidéo Amazon Rekognition à accéder à des flux vidéo et à des flux de données Kinesis individuels.

Pour autoriser Vidéo Amazon Rekognition d'accéder à un processeur de flux de recherche faciale

1. Créez une nouvelle stratégie d'autorisations avec l'éditeur de stratégie IAM JSON, et utilisez la stratégie suivante. Remplacez `video-arn` par l'ARN du flux vidéo Kinesis souhaité. Si vous utilisez un processeur de flux de recherche faciale, remplacez `data-arn` par l'ARN du flux de données Kinesis souhaité.
2. Créez une fonction du service IAM;, ou mettez à jour une fonction du service IAM existant. Utilisez les informations suivantes pour créer la fonction du service IAM :
   1. Choisissez Rekognition pour le nom du service.
   2. Choisissez Rekognition pour le cas d'utilisation de la fonction de service.
   3. Attachez la stratégie d'autorisations que vous avez créée à l'étape 1.

3. Notez l'ARN de la fonction du service. Vous en aurez besoin pour démarrer les opérations d'analyse vidéo.

Donner accès aux streams à l'aide de AmazonRekognitionServiceRole

Vous pouvez utiliser la politique d'autorisation `AmazonRekognitionServiceRole` comme option alternative pour configurer l'accès aux flux vidéo et aux flux de données Kinesis. IAM présente le cas d'utilisation de la fonction du service Rekognition qui, lorsqu'elle est utilisée avec la stratégie d'autorisations `AmazonRekognitionServiceRole`, peut écrire dans plusieurs flux de données Kinesis et lire à partir de tous vos flux de données Kinesis. Pour autoriser Amazon Rekognition Video à accéder en écriture à plusieurs flux de données Kinesis, vous pouvez ajouter, par exemple, au début des noms des flux de données Kinesis. AmazonRekognition`AmazonRekognitionMyDataStreamName`

Pour autoriser Vidéo Amazon Rekognition à accéder à votre flux vidéo Kinesis et à votre flux de données Kinesis

1. Créez une fonction du service IAM. Utilisez les informations suivantes pour créer la fonction du service IAM :
   1. Choisissez Rekognition pour le nom du service.
   2. Choisissez Rekognition pour le cas d'utilisation de la fonction de service.
   3. Choisissez la politique AmazonRekognitionServiceRoled'autorisation, qui donne à Amazon Rekognition Video un accès en écriture aux flux de données Kinesis préfixés AmazonRekognitionet un accès en lecture à tous vos flux vidéo Kinesis.
2. Pour garantir votre Compte AWS sécurité, limitez l'étendue de l'accès de Rekognition aux seules ressources que vous utilisez. Pour ce faire, attachez une politique de confiance à votre fonction du service IAM. Pour plus d'informations sur la procédure à utiliser, consultez Cross-service prévention confuse des adjoints.
3. Notez l'Amazon Resource Name (ARN) de la fonction du service. Vous en aurez besoin pour démarrer les opérations d'analyse vidéo.

Recherche de visages dans une vidéo en streaming

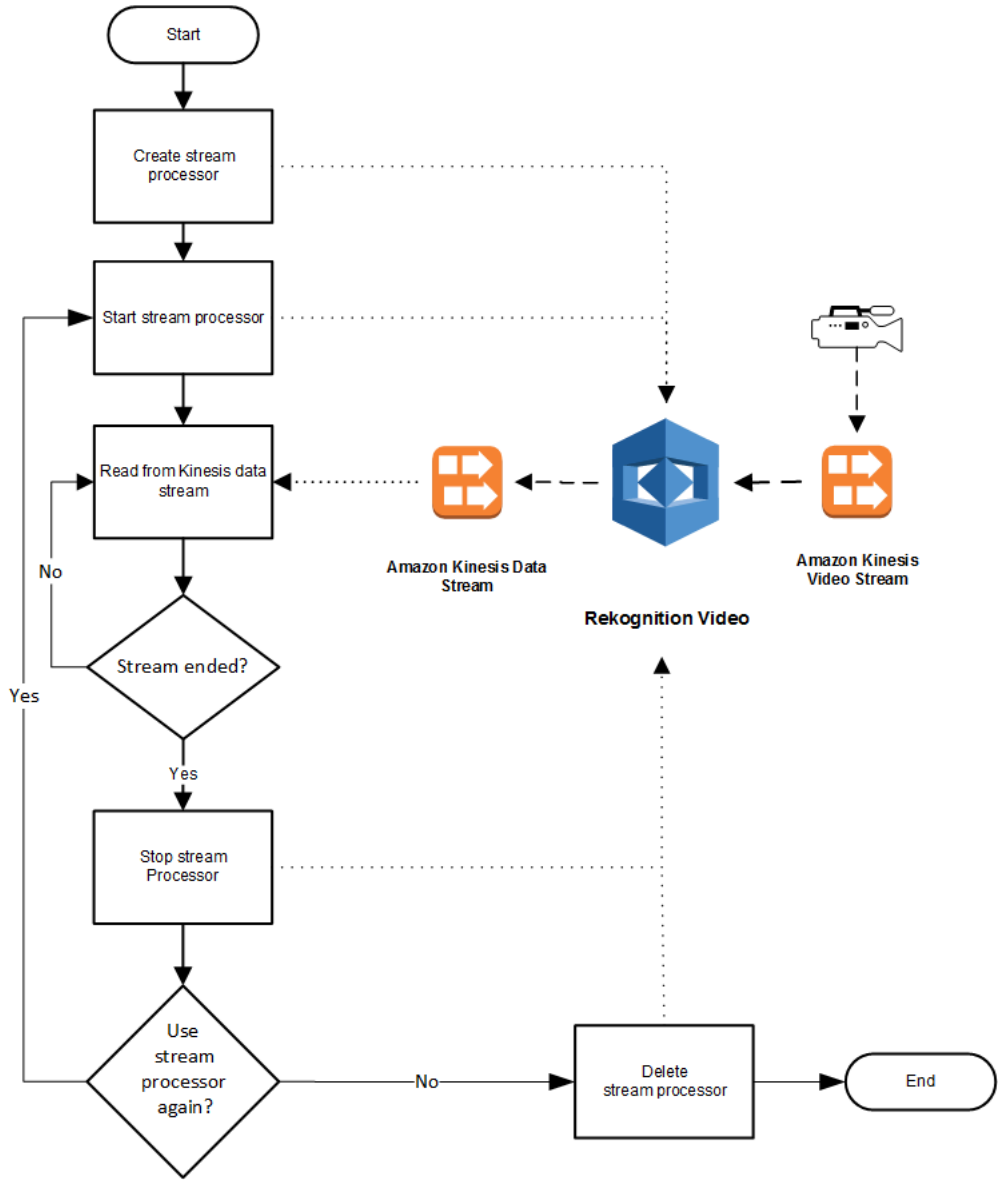
> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vidéo Amazon Rekognition peut rechercher, dans une collection, des visages qui correspondent à des visages ayant été détectés dans une vidéo en streaming. Pour en savoir plus sur les collections, consultez Recherche de visages dans une collection.

Rubriques

- Création du processeur de flux de recherche faciale Vidéo Amazon Rekognition
- Démarrage du processeur de flux de recherche faciale Vidéo Amazon Rekognition
- Utilisation de processeurs de flux pour la recherche faciale (exemple Java V2)
- Utilisation de processeurs de flux pour la recherche faciale (exemple Java V1)
- Lecture des résultats d’analyse de vidéo en streaming
- Afficher les résultats de Rekognition avec Kinesis Video Streams en local
- Comprendre l'enregistrement de trame JSON de reconnaissance faciale Kinesis

Le diagramme suivant montre comment Vidéo Amazon Rekognition; détecte et reconnaît les visages dans une vidéo en streaming.



Création du processeur de flux de recherche faciale Vidéo Amazon Rekognition

Avant de pouvoir analyser une vidéo en streaming, vous devez créer un processeur de streaming Amazon Rekognition Video (). CreateStreamProcessor Le processeur de flux contient des informations sur le flux de données Kinesis et le flux vidéo Kinesis. Il comprend également l'identifiant de la collection contenant les visages que vous voulez reconnaître dans la vidéo en streaming d'entrée. Vous définissez aussi un nom pour le processeur de flux. L'exemple suivant est un exemple JSON de requête `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
                }
        }
}
```

Voici un exemple de réponse de `CreateStreamProcessor`.

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

Démarrage du processeur de flux de recherche faciale Vidéo Amazon Rekognition

Pour démarrer l'analyse de vidéo en streaming, appelez StartStreamProcessor avec le nom de processeur de flux que vous avez spécifié dans `CreateStreamProcessor`. L’exemple suivant est un exemple JSON de requête `StartStreamProcessor`.

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

Si le processeur de flux démarre avec succès, une réponse HTTP 200 est renvoyée avec un corps de JSON vide.

Utilisation de processeurs de flux pour la recherche faciale (exemple Java V2)

L'exemple de code suivant montre comment appeler diverses opérations du processeur de flux, telles que CreateStreamProcessoret StartStreamProcessor, à l'aide du AWS SDK pour Java version 2.

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
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.");
    }
}
```

Utilisation de processeurs de flux pour la recherche faciale (exemple Java V1)

L'exemple de code suivant montre comment appeler diverses opérations du processeur de flux, telles que CreateStreamProcessoret StartStreamProcessor, à l'aide de Java V1. L'exemple inclut une classe de gestionnaire de processeur de flux (StreamManager) qui fournit des méthodes pour appeler les opérations du processeur de flux. La classe de démarrage (Starter) crée un StreamManager objet et appelle diverses opérations.

Pour configurer l’exemple:

1. Attribuez les valeurs souhaitées aux champs de membre de la classe Starter.
2. Dans la fonction `main` de la classe Starter, supprimez la mise en commentaire de l’appel de fonction souhaité.

Classe Starter

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

// Starter class. Use to create a StreamManager class
// and call stream processor operations.
package com.amazonaws.samples;
import com.amazonaws.samples.*;

public class Starter {

 public static void main(String[] args) {


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

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

StreamManager classe

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

// Stream manager class. Provides methods for calling
// Stream Processor operations.
package com.amazonaws.samples;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.CreateStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.CreateStreamProcessorResult;
import com.amazonaws.services.rekognition.model.DeleteStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.DeleteStreamProcessorResult;
import com.amazonaws.services.rekognition.model.DescribeStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.DescribeStreamProcessorResult;
import com.amazonaws.services.rekognition.model.FaceSearchSettings;
import com.amazonaws.services.rekognition.model.KinesisDataStream;
import com.amazonaws.services.rekognition.model.KinesisVideoStream;
import com.amazonaws.services.rekognition.model.ListStreamProcessorsRequest;
import com.amazonaws.services.rekognition.model.ListStreamProcessorsResult;
import com.amazonaws.services.rekognition.model.StartStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.StartStreamProcessorResult;
import com.amazonaws.services.rekognition.model.StopStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.StopStreamProcessorResult;
import com.amazonaws.services.rekognition.model.StreamProcessor;
import com.amazonaws.services.rekognition.model.StreamProcessorInput;
import com.amazonaws.services.rekognition.model.StreamProcessorOutput;
import com.amazonaws.services.rekognition.model.StreamProcessorSettings;

public class StreamManager {

    private String streamProcessorName;
    private String kinesisVideoStreamArn;
    private String kinesisDataStreamArn;
    private String roleArn;
    private String collectionId;
    private float matchThreshold;
```

```
    private AmazonRekognition rekognitionClient;


    public StreamManager(String spName,
      String kvStreamArn,
      String kdStreamArn,
      String iamRoleArn,
      String collId,
      Float threshold){
     streamProcessorName=spName;
     kinesisVideoStreamArn=kvStreamArn;
     kinesisDataStreamArn=kdStreamArn;
     roleArn=iamRoleArn;
     collectionId=collId;
     matchThreshold=threshold;
     rekognitionClient=AmazonRekognitionClientBuilder.defaultClient();

    }

    public void createStreamProcessor() {
     //Setup input parameters
        KinesisVideoStream kinesisVideoStream = new
KinesisVideoStream().withArn(kinesisVideoStreamArn);
        StreamProcessorInput streamProcessorInput =
                new StreamProcessorInput().withKinesisVideoStream(kinesisVideoStream);
        KinesisDataStream kinesisDataStream = new
KinesisDataStream().withArn(kinesisDataStreamArn);
        StreamProcessorOutput streamProcessorOutput =
                new StreamProcessorOutput().withKinesisDataStream(kinesisDataStream);
        FaceSearchSettings faceSearchSettings =
                new
FaceSearchSettings().withCollectionId(collectionId).withFaceMatchThreshold(matchThreshold);
        StreamProcessorSettings streamProcessorSettings =
                new StreamProcessorSettings().withFaceSearch(faceSearchSettings);

        //Create the stream processor
        CreateStreamProcessorResult createStreamProcessorResult =
rekognitionClient.createStreamProcessor(
                new
CreateStreamProcessorRequest().withInput(streamProcessorInput).withOutput(streamProcessorOutpu

.withSettings(streamProcessorSettings).withRoleArn(roleArn).withName(streamProcessorName));

        //Display result
```

```
        System.out.println("Stream Processor " + streamProcessorName + " created.");
        System.out.println("StreamProcessorArn - " +
createStreamProcessorResult.getStreamProcessorArn());
    }

    public void startStreamProcessor() {
        StartStreamProcessorResult startStreamProcessorResult =
                rekognitionClient.startStreamProcessor(new
StartStreamProcessorRequest().withName(streamProcessorName));
        System.out.println("Stream Processor " + streamProcessorName + " started.");
    }

    public void stopStreamProcessor() {
        StopStreamProcessorResult stopStreamProcessorResult =
                rekognitionClient.stopStreamProcessor(new
StopStreamProcessorRequest().withName(streamProcessorName));
        System.out.println("Stream Processor " + streamProcessorName + " stopped.");
    }

    public void deleteStreamProcessor() {
        DeleteStreamProcessorResult deleteStreamProcessorResult = rekognitionClient
                .deleteStreamProcessor(new
DeleteStreamProcessorRequest().withName(streamProcessorName));
        System.out.println("Stream Processor " + streamProcessorName + " deleted.");
    }

    public void describeStreamProcessor() {
        DescribeStreamProcessorResult describeStreamProcessorResult = rekognitionClient
                .describeStreamProcessor(new
DescribeStreamProcessorRequest().withName(streamProcessorName));

        //Display various stream processor attributes.
        System.out.println("Arn - " +
describeStreamProcessorResult.getStreamProcessorArn());
        System.out.println("Input kinesisVideo stream - "
                +
describeStreamProcessorResult.getInput().getKinesisVideoStream().getArn());
        System.out.println("Output kinesisData stream - "
                +
describeStreamProcessorResult.getOutput().getKinesisDataStream().getArn());
        System.out.println("RoleArn - " + describeStreamProcessorResult.getRoleArn());
        System.out.println(
                "CollectionId - " +
describeStreamProcessorResult.getSettings().getFaceSearch().getCollectionId());
```

```
        System.out.println("Status - " + describeStreamProcessorResult.getStatus());
        System.out.println("Status message - " +
 describeStreamProcessorResult.getStatusMessage());
        System.out.println("Creation timestamp - " +
 describeStreamProcessorResult.getCreationTimestamp());
        System.out.println("Last update timestamp - " +
 describeStreamProcessorResult.getLastUpdateTimestamp());
    }

    public void listStreamProcessors() {
        ListStreamProcessorsResult listStreamProcessorsResult =
                rekognitionClient.listStreamProcessors(new
 ListStreamProcessorsRequest().withMaxResults(100));

        //List all stream processors (and state) returned from Rekognition
        for (StreamProcessor streamProcessor :
 listStreamProcessorsResult.getStreamProcessors()) {
            System.out.println("StreamProcessor name - " + streamProcessor.getName());
            System.out.println("Status - " + streamProcessor.getStatus());
        }
    }
}
```

Lecture des résultats d’analyse de vidéo en streaming

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vous pouvez utiliser la bibliothèque client Amazon Kinesis Data Streams pour consommer des résultats d’analyse envoyés vers le flux de sortie Amazon Kinesis Data Streams. Pour en savoir plus, consultez Lecture de données d’un flux de données Kinesis. Vidéo Amazon Rekognition place un enregistrement de trame JSON pour chaque trame analysée dans le flux de sortie Kinesis. Vidéo Amazon Rekognition n’analyse pas toutes les images qui lui sont transmises via le flux vidéo Kinesis.

Un enregistrement de trame envoyé à un flux de données Kinesis comprend des informations sur le fragment de flux vidéo Kinesis où se trouve la trame dans le fragment et les visages reconnus dans la trame. Il inclut également des informations de statut pour le processeur de flux. Pour de plus amples informations, veuillez consulter Comprendre l'enregistrement de trame JSON de reconnaissance faciale Kinesis.

La bibliothèque d'analyseurs Amazon Kinesis Video Streams contient des exemples de tests qui utilisent les résultats de Vidéo Amazon Rekognition et les intègre au flux vidéo Kinesis d'origine. Pour de plus amples informations, veuillez consulter Afficher les résultats de Rekognition avec Kinesis Video Streams en local.

Vidéo Amazon Rekognition diffuse les informations d'analyse Vidéo Amazon Rekognition vers le flux de données Kinesis. L'exemple suivant est un exemple JSON pour un enregistrement unique.

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

Dans l’exemple JSON, notez les éléments suivants :

- InputInformation— Informations sur le flux vidéo Kinesis utilisé pour diffuser des vidéos sur Amazon Rekognition Video. Pour de plus amples informations, veuillez consulter InputInformation.
- StreamProcessorInformation— Informations sur l'état du processeur de streaming Amazon Rekognition Video. La seule valeur possible pour le champ `Status` est RUNNING. Pour de plus amples informations, veuillez consulter StreamProcessorInformation.
- FaceSearchResponse— Contient des informations sur les visages de la vidéo en streaming qui correspondent aux visages de la collection d'entrées. FaceSearchResponsecontient un DetectedFace objet, qui est un visage détecté dans l'image vidéo analysée. Pour chaque visage détecté, le tableau `MatchedFaces` contient une série d’objets visage correspondants (MatchedFace) trouvés dans la collection d’entrée, avec un pourcentage de similarité.

Mappage du flux vidéo Kinesis au flux de données Kinesis

Vous pouvez vouloir mapper les trames de flux vidéo Kinesis aux trames analysées envoyées au flux de données Kinesis. Par exemple, lors de l’affichage d’une vidéo en streaming, vous pouvez souhaiter afficher des cadres autour des visages des personnes reconnues. Les coordonnées du cadre de délimitation sont envoyées au flux de données Kinesis comme faisant partie de l’enregistrement de reconnaissance faciale Kinesis. Pour afficher correctement le cadre de délimitation, vous devez mapper les informations horaires qui sont envoyées avec l’enregistrement de reconnaissance faciale Kinesis aux trames correspondantes dans le flux vidéo Kinesis.

La technique que vous utilisez pour mapper le flux vidéo Kinesis au flux de données Kinesis varie selon que vous diffusez du contenu multimédia en direct (vidéo de streaming en direct) ou du contenu multimédia archivé (vidéo stockée).

Mappage en cas de diffusion de contenu multimédia en direct

Pour mapper une trame de flux vidéo Kinesis à une trame de flux de données Kinesis

1. Définissez le paramètre d'entrée `FragmentTimeCodeType` de l'PutMediaopération sur`RELATIVE`.
2. Appelez `PutMedia` pour diffuser du contenu multimédia en direct dans le flux vidéo Kinesis.
3. Lorsque vous recevez un enregistrement de reconnaissance faciale Kinesis du flux de données Kinesis, stockez les valeurs de `ProducerTimestamp` et `FrameOffsetInSeconds` du champ KinesisVideo.
4. Calculez l'horodatage système qui correspond à la trame du flux vidéo Kinesis en ajoutant les valeurs des champs `ProducerTimestamp` et `FrameOffsetInSeconds`.

Mappage en cas de diffusion de contenu multimédia archivé

Pour mapper une trame de flux vidéo Kinesis à une trame de flux de données Kinesis

1. Appelez PutMediapour transférer du contenu multimédia archivé dans le flux vidéo Kinesis.
2. Lorsque vous recevez un objet `Acknowledgement` dans la réponse de l'opération `PutMedia`, stockez la valeur du champ `FragmentNumber` à partir du champ Payload. `FragmentNumber` est le numéro de fragment du cluster MKV.
3. Lorsque vous recevez un enregistrement de reconnaissance faciale Kinesis du flux de données Kinesis, stockez les valeurs de champ `FrameOffsetInSeconds` du champ KinesisVideo.
4. Calculez le mappage en utilisant les valeurs `FrameOffsetInSeconds` et `FragmentNumber` stockées aux étapes 2 et 3. `FrameOffsetInSeconds` est le décalage dans le fragment avec le paramètre spécifique `FragmentNumber` qui est envoyé au Amazon Kinesis Data Streams. Pour plus d'informations sur l'obtention des trames vidéo pour un numéro de fragment donné, consultez le contenu multimédia archivé Amazon Kinesis Video Streams.

Afficher les résultats de Rekognition avec Kinesis Video Streams en local

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.

Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vous pouvez consulter les résultats d'Amazon Rekognition Video Video Streams affichés dans votre flux d'Amazon Kinesis Video Streams à l'aide des exemples de tests de la bibliothèque d'analyseurs Amazon Kinesis Video Streams fournis sur - Rekognition Examples. KinesisVideo `KinesisVideoRekognitionIntegrationExample` affiche les cadres de délimitations au-dessus des visages détectés et affiche la vidéo localement via JFrame. Ce processus suppose que vous avez correctement connecté une entrée multimédia provenant de la caméra de l'appareil à un flux vidéo Kinesis, et que vous avez démarré un processeur de flux Amazon Rekognition. Pour de plus amples informations, veuillez consulter Streaming à l'aide d'un plugin GStreamer.

Étape 1 : installation de la bibliothèque d'analyseurs Kinesis Video Streams

Pour créer un répertoire et télécharger le référentiel Github, exécutez la commande suivante :

```
$ git clone https://github.com/aws/amazon-kinesis-video-streams-parser-library.git
```

Accédez au répertoire de la bibliothèque et exécutez la commande Maven suivante pour effectuer une installation propre :

```
$ mvn clean install
```

Étape 2 : configuration de l'exemple de test d'intégration de Kinesis Video Streams et Rekognition

Ouvrez le fichier `KinesisVideoRekognitionIntegrationExampleTest.java`. Supprimez le `@Ignore` juste après l'en-tête de la classe. Renseignez les champs de données avec les informations provenant de vos ressources Amazon Kinesis et Amazon Rekognition. Pour de plus amples informations, veuillez consulter Configuration de vos ressources Vidéo Amazon Rekognition et Amazon Kinesis. Si vous diffusez une vidéo sur votre flux vidéo Kinesis, supprimez le paramètre `inputStream`.

Consultez l'exemple de code suivant :

```
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();
```

Étape 3 : exécution de l’exemple de test d’intégration de Kinesis Video Streams et Rekognition

Assurez-vous que votre flux vidéo Kinesis reçoit une entrée multimédia si vous le diffusez, et commencez à analyser votre flux avec un processeur de diffusion vidéo Amazon Rekognition en cours d’exécution. Pour de plus amples informations, veuillez consulter Présentation des opérations du processeur de flux Vidéo Amazon Rekognition. Exécutez la classe `KinesisVideoRekognitionIntegrationExampleTest` en tant que test JUnit. Après un court laps de temps, une nouvelle fenêtre s’ouvre avec un flux vidéo issu de votre flux vidéo Kinesis, avec des cadres de délimitation dessinés au-dessus des visages détectés.

> **Note**
>
> Les visages de la collection utilisée dans cet exemple doivent avoir un ID d'image externe (le nom du fichier) spécifié dans ce format pour que les étiquettes des encadrés affichent un texte significatif : PersonName1-Trusted, PersonName2-Intruder, PersonName3-Neutral, etc.

Les étiquettes peuvent également être codées par couleur et sont personnalisables dans le FaceType.java fichier.

Comprendre l'enregistrement de trame JSON de reconnaissance faciale Kinesis

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vidéo Amazon Rekognition peut reconnaître des visages dans une vidéo diffusée en streaming. Pour chaque trame analysée, Vidéo Amazon Rekognition transmet un enregistrement de trame JSON à un flux de données . Vidéo Amazon Rekognition n'analyse pas toutes les images qui lui sont transmises via le flux vidéo Kinesis.

L'enregistrement de trame JSON contient des informations sur les flux d'entrée et de sortie, le statut du processeur de flux, ainsi que des informations sur les visages reconnus dans les trames analysées. Cette section contient des informations de référence pour l'enregistrement de fréquence JSON.

Voici la syntaxe JSON pour un enregistrement de flux de données Kinesis. Pour de plus amples informations, veuillez consulter Utilisation des vidéos streaming.

> **Note**
>
> L'API de Vidéo Amazon Rekognition compare les visages de votre flux d'entrée à une collection de visages, puis renvoie les correspondances les plus proches trouvées, ainsi qu'un score de similarité.

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

Enregistrement JSON

L'enregistrement JSON contient inclut des informations sur une trame traitée par Vidéo Amazon Rekognition. L'enregistrement contient des informations sur la vidéo streaming, le statut de la trame analysée et les visages reconnus dans la trame.

InputInformation

Informations sur le flux vidéo Kinesis utilisé pour diffuser des vidéos dans Vidéo Amazon Rekognition.

Type : objet InputInformation

StreamProcessorInformation

Informations sur le processeur de flux Vidéo Amazon Rekognition. Comprend des informations sur le statut en cours du processeur de flux.

Type : objet StreamProcessorInformation

FaceSearchResponse

Informations sur les visages détectés dans la trame vidéo en streaming et sur les visages correspondants trouvés dans la collection d'entrée.

Type : tableau d'objets FaceSearchResponse

InputInformation

Informations sur un flux vidéo source utilisé par Vidéo Amazon Rekognition. Pour de plus amples informations, veuillez consulter Utilisation des vidéos streaming.

KinesisVideo

Type : objet KinesisVideo

KinesisVideo

Informations sur le flux vidéo Kinesis qui diffuse la vidéo source sur Vidéo Amazon Rekognition. Pour de plus amples informations, veuillez consulter Utilisation des vidéos streaming.

StreamArn

Le Amazon Resource Name (ARN) du flux vidéo Kinesis.

Type : Chaîne

FragmentNumber

Le fragment de vidéo streaming qui contient la trame que cet enregistrement représente.

Type : Chaîne

ProducerTimestamp

L'horodatage Unix côté producteur du fragment. Pour de plus amples informations, veuillez consulter PutMedia.

Type : Number

ServerTimestamp

L'horodatage Unix côté serveur du fragment. Pour de plus amples informations, veuillez consulter PutMedia.

Type : Number

FrameOffsetInSeconds

Le décalage de la trame (en secondes) dans le fragment.

Type : Number

StreamProcessorInformation

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Informations de statut sur le processeur de flux.

Statut

Le statut en cours du processeur de flux. La seule valeur possible est RUNNING.

Type : Chaîne

FaceSearchResponse

Informations sur un visage détecté dans la trame vidéo en streaming et sur les visages dans une collection qui correspondent au visage détecté. Vous spécifiez la collection dans un appel à CreateStreamProcessor. Pour de plus amples informations, veuillez consulter Utilisation des vidéos streaming.

DetectedFace

Détails d'un visage détecté dans une trame vidéo analysée.

Type : objet DetectedFace

MatchedFaces

Tableau de détails de visages pour des visages dans une collection qui correspond au visage détecté dans `DetectedFace`.

Type : tableau d'objets MatchedFace

DetectedFace

Informations sur un visage détecté dans une trame vidéo en streaming. Les visages correspondants dans la collection d'entrée sont disponibles dans le champ d'objet MatchedFace.

BoundingBox

Le cadre de délimitation s'accorde avec un visage détecté dans une trame vidéo analysée. L' BoundingBox objet possède les mêmes propriétés que l' BoundingBox objet utilisé pour l'analyse d'image.

Type : objet BoundingBox

Fiabilité

Correspond au niveau de confiance (1 à 100) de Vidéo Amazon Rekognition quant au fait que le visage détecté est bien un visage. 1 correspond au niveau de confiance le plus faible, 100 au niveau le plus élevé.

Type : Number

Traits caractéristiques

Un tableau de traits caractéristiques de visage.

Type : tableau d'objets repères

Expression

Indique l'expression du visage selon les axes vertical, horizontal, et l'orientation.

Type : objet d'expression

Qualité

Identifie la luminosité et la netteté de l'image du visage.

Type : objet ImageQuality

MatchedFace

Informations sur un visage correspondant à un visage détecté dans une trame vidéo analysée.

Visage

Informations de correspondance de visages pour un visage de la collection d'entrée qui correspond au visage dans l'objet DetectedFace.

Type : objet facial

Similarité

Le niveau de confiance (1 à 100) auquel les visages correspondent. 1 est le niveau de confiance le plus faible, 100 le plus élevé.

Type : Number

Streaming à l'aide d'un plugin GStreamer

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vidéo Amazon Rekognition peut analyser une vidéo diffusée en direct à partir de la caméra d'un appareil. Pour accéder à l'entrée multimédia depuis une source de périphérique, vous devez installer GStreamer. GStreamer est un logiciel infrastructure multimédia tiers qui connecte les sources multimédia et les outils de traitement dans des pipelines de flux de travail. Vous devez également installer le plugin Amazon Kinesis Video Streams Producer pour Gstreamer. Ce processus suppose que vous avez correctement configuré vos ressources Vidéo Amazon Rekognition et Amazon Kinesis. Pour de plus amples informations, veuillez consulter Configuration de vos ressources Vidéo Amazon Rekognition et Amazon Kinesis.

Étape 1 : installer Gstreamer

Téléchargez et installez Gstreamer, un logiciel de plateforme multimédia tiers. Vous pouvez utiliser un logiciel de gestion de paquets tel que Homebrew (Gstreamer sur Homebrew) ou vous le procurer directement depuis le site Web de Freedesktop.

Vérifiez la réussite de l'installation de Gstreamer en lançant un flux vidéo avec une source de test depuis votre terminal de ligne de commande.

```
$ gst-launch-1.0 videotestsrc ! autovideosink
```

Étape 2 : installation du plugin Kinesis Video Streams Producer

Dans cette section, vous allez télécharger la bibliothèque Amazon Kinesis Video Streams Producer et installer le plugin Kinesis Video Streams Gstreamer.

Créez un répertoire et clonez l'exemple de code source à partir du référentiel GitHub. Assurez-vous d'inclure le paramètre `--recursive`.

```
$ git clone --recursive https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp.git
```

Suivez les instructions fournies par la bibliothèque pour configurer et créer le projet. Assurez-vous d'utiliser les commandes spécifiques à la plate-forme pour votre système d'exploitation. Utilisez le paramètre `-DBUILD_GSTREAMER_PLUGIN=ON` lorsque vous exécutez `cmake` pour installer le plug-in Kinesis Video Streams Gstreamer. Ce projet nécessite les packages supplémentaires suivants inclus dans l'installation : GCC ou Clang, Curl, Openssl et Log4CPlus. Si votre compilation échoue à cause d'un package manquant, vérifiez que le package est installé et qu'il figure dans votre PATH. Si vous rencontrez une erreur « Impossible d'exécuter le programme compilé en C » lors de la construction, réexécutez la commande de construction. Parfois, le compilateur C correct n'est pas trouvé.

Vérifiez l'installation du plug-in Kinesis Video Streams en exécutant la commande suivante.

```
$ gst-inspect-1.0 kvssink
```

Les informations suivantes, telles que les informations relatives à l'usine et au plug-in, doivent apparaître :

```
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/

  ...
```

Étape 3 : Exécuter Gstreamer avec le plug-in Kinesis Video Streams

Avant de commencer le streaming depuis la caméra d'un appareil vers Kinesis Video Streams, vous devrez peut-être convertir la source multimédia en un codec acceptable pour Kinesis Video Streams. Pour déterminer les spécifications et les capacités de formatage des appareils actuellement connectés à votre machine, exécutez la commande suivante.

```
$ gst-device-monitor-1.0
```

Pour commencer le streaming, lancez Gstreamer à l'aide de l'exemple de commande suivant, et ajoutez vos informations d'identification et les informations Amazon Kinesis Video Streams. Vous devez utiliser les clés d'accès et la région correspondant à la fonction du service IAM que vous avez créée lorsque vous autorisez Amazon Rekognition à accéder à vos flux Kinesis. Pour de plus amples informations sur les clés d'accès, veuillez consulter Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM. Vous pouvez également ajuster les paramètres des arguments du format vidéo en fonction de votre utilisation et de leur disponibilité sur votre appareil.

```
$ 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"
```

Pour plus de commandes de lancement, voir Exemples de commandes de lancement de GStreamer.

> **Note**
>
> Si votre commande de lancement se termine par une erreur non liée à la négociation, vérifiez le résultat du Device Monitor, et assurez-vous que les valeurs des paramètres `videoconvert`correspondent aux fonctionnalités valides de votre appareil.

Au bout de quelques secondes, un flux vidéo provenant de la caméra de votre appareil s'affiche sur votre flux vidéo Kinesis. Pour commencer à détecter et à associer des visages avec Amazon Rekognition, démarrez votre processeur de streaming Vidéo Amazon Rekognition. Pour de plus amples informations, veuillez consulter Présentation des opérations du processeur de flux Vidéo Amazon Rekognition.

Résolution des problèmes de streaming vidéo

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Cette rubrique fournit des informations permettant de résoudre les problèmes d'utilisation de Vidéo Amazon Rekognition avec des vidéos en streaming.

Rubriques

- Je ne sais pas si mon processeur de flux a été créé avec succès
- Je ne sais pas si j'ai configuré mon processeur de flux correctement
- Mon processeur de flux ne renvoie pas de résultats.
- L'état de mon processeur de flux est FAILED
- Mon processeur de flux ne renvoie pas les résultats attendus

Je ne sais pas si mon processeur de flux a été créé avec succès

Utilisez la AWS CLI commande suivante pour obtenir la liste des processeurs de flux et leur état actuel.

```
aws rekognition list-stream-processors
```

Vous pouvez obtenir des informations supplémentaires à l'aide de la AWS CLI commande suivante. Remplacez `stream-processor-name` par le nom du processeur de flux requis.

```
aws rekognition describe-stream-processor --name stream-processor-name
```

Je ne sais pas si j'ai configuré mon processeur de flux correctement

Si votre code ne donne pas des résultats d'analyse à partir de Vidéo Amazon Rekognition, votre processeur de flux n'est peut-être pas configuré correctement. Procédez comme suit pour vérifier que votre processeur de flux est configuré correctement et capable de produire des résultats.

Pour déterminer si votre solution est configurée correctement

1. Exécutez la commande suivante pour vérifier que votre processeur de flux est en cours d'exécution. Remplacez `stream-processor-name` par le nom de votre processeur de flux. Le processeur de flux est en cours d'exécution si la valeur de `Status` est `RUNNING`. Si le statut est `RUNNING` et que vous n'obtenez pas de résultats, consultez Mon processeur de flux ne renvoie pas de résultats.. Si l'état est `FAILED`, consultez L'état de mon processeur de flux est FAILED.

   ```
   aws rekognition describe-stream-processor --name stream-processor-name
   ```

2. Si votre processeur de flux est en cours d'exécution, exécutez le Bash ou la PowerShell commande suivante pour lire les données du flux de données Kinesis en sortie.

   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. Utilisez l'outil de décodage sur le site web Base64 Decode pour décoder la sortie en chaîne compréhensible par les utilisateurs. Pour plus d'informations, consultez Etape 3 : Obtenir l'enregistrement.
4. Si les commandes fonctionnent et que vous voyez des résultats de détection de visages dans les flux de données Kinesis, cela signifie que votre solution a été configurée correctement. Si la commande échoue, vérifiez les autres suggestions de résolution de problèmes et consultez Octroi à Vidéo Amazon Rekognition d'un accès à vos ressources.

Vous pouvez également utiliser le AWS Lambda plan « kinesis-process-record » pour enregistrer les messages du flux de données Kinesis à des fins de visualisation continue. CloudWatch Cela entraîne des coûts supplémentaires pour AWS Lambda et. CloudWatch

Mon processeur de flux ne renvoie pas de résultats.

Plusieurs raisons peuvent expliquer pourquoi votre processeur de flux ne renvoie pas de résultats.

Raison 1 : votre processeur de flux n'est pas configuré correctement

Il est possible que votre processeur de flux ne soit pas configuré correctement. Pour de plus amples informations, veuillez consulter Je ne sais pas si j'ai configuré mon processeur de flux correctement.

Raison 2 : votre processeur de flux n'est pas dans l'état RUNNING

Pour résoudre un problème d'état du processeur de flux

1. Vérifiez l'état du processeur de flux à l'aide de la AWS CLI commande suivante.

```
aws rekognition describe-stream-processor --name stream-processor-name
```

2. Si la valeur de `Status` est `STOPPED`, démarrez votre processeur de flux à l'aide de la commande suivante :

```
aws rekognition start-stream-processor --name stream-processor-name
```

3. Si la valeur de `Status` est `FAILED`, consultez L'état de mon processeur de flux est FAILED.

4. Si la valeur de `Status` est `STARTING`, attendez 2 minutes et vérifiez l'état en répétant l'étape 1. Si la valeur de l'état est toujours `STARTING`, procédez comme suit :

   a. Supprimez le processeur de flux à l'aide de la commande suivante.

   ```
   aws rekognition delete-stream-processor --name stream-processor-name
   ```

   b. Créez processeur de flux avec la même configuration. Pour de plus amples informations, veuillez consulter Utilisation des vidéos streaming.

   c. Si le problème persiste, contactez le AWS Support.

5. Si la valeur de `Status` est `RUNNING`, consultez Raison 3 : il n'y a pas de données actives dans le flux vidéo Kinesis.

Raison 3 : il n'y a pas de données actives dans le flux vidéo Kinesis

Pour vérifier s'il n'y a pas de données actives dans le flux vidéo Kinesis

1. Connectez-vous à la Console de gestion AWS console https://console.aws.amazon.com/kinesisvideo/Amazon Kinesis Video Streams et ouvrez-la à l'adresse.
2. Sélectionnez le flux vidéo Kinesis qui est l'entrée du processeur de flux Amazon Rekognition.
3. Si l'aperçu indique No data on stream, cela signifie qu'il n'y a pas de données à traiter dans le flux d'entrée par Vidéo Amazon Rekognition.

Pour plus d'informations sur la production de vidéos avec Kinesis Video Streams, consultez Bibliothèques de production de flux vidéo Kinesis.

L'état de mon processeur de flux est FAILED

Vous pouvez vérifier l'état d'un processeur de flux à l'aide de la AWS CLI commande suivante.

```
aws rekognition describe-stream-processor --name stream-processor-name
```

Si la valeur de l'état est FAILED, vérifiez les informations de résolution de problèmes pour les messages d'erreur suivants.

Erreur : »Accès refusé à « Role »

Le rôle IAM qui est utilisé par le processeur de flux n'existe pas ou Vidéo Amazon Rekognition n'a pas l'autorisation d'assumer ce rôle.

Pour résoudre les problèmes d'accès au rôle IAM

1. Connectez-vous à la console IAM Console de gestion AWS et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Dans le panneau de navigation gauche, choisissez Roles (Rôles) et vérifiez que le rôle existe.
3. Si le rôle existe, vérifiez qu'il est conforme à la politique AmazonRekognitionServiceRoled'autorisation.
4. Si le rôle n'existe pas ou n'a pas les autorisations appropriées, consultez Octroi à Vidéo Amazon Rekognition d'un accès à vos ressources.
5. Démarrez le processeur de flux à l'aide de la AWS CLI commande suivante.

```
aws rekognition start-stream-processor --name stream-processor-name
```

Erreur : « Accès refusé à Kinesis Video or Accès refusé à Kinesis Data »

Le rôle n'a pas accès aux opérations d'API Kinesis Video Streams `GetMedia` et `GetDataEndpoint`. Il est également possible qu'il n'ait pas accès aux opérations d'API flux de données Kinesis `PutRecord` et `PutRecords`.

Pour résoudre les problèmes relatifs aux autorisations d'API

1. Connectez-vous à la console IAM Console de gestion AWS et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Ouvrez le rôle et assurez-vous que la stratégie d'autorisations suivante lui est attachée.
3. S'il manque l'une des autorisations, mettez à jour la stratégie. Pour de plus amples informations, veuillez consulter Octroi à Vidéo Amazon Rekognition d'un accès à vos ressources.

Erreur : « Diffusez ***nom du flux vidéo d'entrée*** n'existe pas »

L'entrée flux vidéo Kinesis du processeur de flux n'existe pas ou n'a pas été configurée correctement.

Pour résoudre les problèmes de flux de vidéo Kinesis

1. Utilisez la commande suivante pour vérifier que le flux existe.

```
aws kinesisvideo list-streams
```

2. Si le flux existe, vérifiez les points suivants.

   - L'Amazon Resource Name (ARN) est identique à l'ARN du flux d'entrée pour le processeur de flux.
   - Le flux vidéo Kinesis doit se trouver dans la même région que le processeur de flux.

   Si le processeur de flux n'est pas configuré correctement, supprimez-le à l'aide de la AWS CLI commande suivante.

```
aws rekognition delete-stream-processor --name stream-processor-name
```

3. Créez un processeur de flux avec les flux vidéo Kinesis. Pour de plus amples informations, veuillez consulter Création du processeur de flux de recherche faciale Vidéo Amazon Rekognition.

Erreur : « Collection non trouvée »

La collection Amazon Rekognition utilisée par le processeur de flux pour trouver les correspondances de visages n'existe pas ou la mauvaise collection est utilisée.

Pour vérifier qu'il s'agit de la bonne collection

1. Utilisez la AWS CLI commande suivante pour déterminer si la collection requise existe. Passez `region` à la AWS région dans laquelle vous utilisez votre processeur de streaming.

```
aws rekognition list-collections --region region
```

   Si la collection requise n'existe pas, créez-en une nouvelle et ajoutez des informations sur le visage. Pour de plus amples informations, veuillez consulter Recherche de visages dans une collection.

2. Dans votre appel à CreateStreamProcessor, vérifiez que la valeur du paramètre d'entrée `CollectionId` est correcte.

3. Démarrez le processeur de flux à l'aide de la AWS CLI commande suivante.

```
aws rekognition start-stream-processor --name stream-processor-name
```

Erreur : « Stream ***nom-flux de données kinésien de sortie*** sous compte ***id-compte*** non trouvé »

Le flux de données Kinesis de sortie utilisé par le processeur de flux n'existe pas dans votre région Compte AWS ou ne se trouve pas dans la même AWS région que votre processeur de flux.

Pour résoudre les problèmes de flux de données Kinesis

1. Utilisez la AWS CLI commande suivante pour déterminer si le flux de données Kinesis existe. Passez `region` à la AWS région dans laquelle vous utilisez votre processeur de streaming.

```
aws kinesis list-streams --region region
```

2. Si le flux de données Kinesis existe, vérifiez que le nom de flux de données Kinesis est le même que celui du flux de sortie qui est utilisé par le processeur de flux.
3. Si le flux de données Kinesis n'existe pas, il se peut qu'il existe dans une autre AWS région. Le flux de données Kinesis doit se trouver dans la même région que le processeur de flux.
4. Le cas échéant, créez un flux de données Kinesis.

   a. Créez un flux de données Kinesis avec le même nom que celui utilisé par le processeur de flux. Pour plus d'informations, consultez Etape 1 : Création d'un flux de données.

   b. Démarrez le processeur de flux à l'aide de la AWS CLI commande suivante.

```
aws rekognition start-stream-processor --name stream-processor-name
```

Mon processeur de flux ne renvoie pas les résultats attendus

Si votre processeur de flux ne renvoie pas les correspondances de visages attendues, utilisez les informations suivantes.

- Recherche de visages dans une collection
- Recommandations pour la configuration de la caméra (vidéo en streaming)

Tracé du parcours de personnes

> **Note**
>
> Avis de fin de support : le 31 octobre 2025, AWS cessera de prendre en charge Amazon Rekognition People Pathing. Après le 31 octobre 2025, vous ne pourrez plus utiliser la fonctionnalité Rekognition People Pathing. Pour plus d'informations, consultez ce billet de blog.

Vidéo Amazon Rekognition peut créer un suivi des chemins empruntés par les personnes dans les vidéos et fournir des informations telles que :

- Emplacement de la personne dans le cadre vidéo au moment où son chemin fait l'objet d'un suivi.
- Des repères faciaux tels que la position de l'œil gauche, lorsqu'ils sont détectés.

Le tracé du parcours des personnes par Vidéo Amazon Rekognition dans les vidéos stockées est une opération asynchrone. Pour commencer à suivre le chemin des personnes dans les vidéos, appelez StartPersonTracking. Vidéo Amazon Rekognition publie l'état d'achèvement de l'opération d'analyse vidéo dans une rubrique Amazon Simple Notification Service. Si l'analyse vidéo est réussie, appelez GetPersonTrackingpour obtenir les résultats de l'analyse vidéo. Pour en savoir plus sur les appels d'opérations d'API Vidéo Amazon Rekognition consultez Appeler les opérations de Vidéo Amazon Rekognition.

La procédure suivante montre comment suivre le chemin des personnes via une vidéo stockée dans un compartiment Amazon S3. L'exemple s'appuie sur le code figurant dans Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK), qui utilise une file d'attente Amazon Simple Queue Service pour obtenir le statut d'achèvement d'une demande d'analyse vidéo.

Pour détecter des personnes dans une vidéo stockée dans un compartiment Amazon S3 (SDK)

1. Effectuez une Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK).
2. Ajoutez le code suivant à la classe `VideoDetect` que vous avez créée à l'étape 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);

        }
```

Dans la fonction `main`, remplacez les lignes :

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

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

avec :

```
        StartPersonDetection(amzn-s3-demo-bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetPersonDetectionResults();
```

Java V2

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
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
```

Dans la fonction `main`, remplacez les lignes :

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

avec :

```
    analyzer.StartPersonPathing()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetPersonPathingResults()
```

CLI

Exécutez la AWS CLI commande suivante pour démarrer le suivi des utilisateurs dans une vidéo.

```
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
```

Mettez à jour les valeurs suivantes :

- Remplacez `amzn-s3-demo-bucket` et `video-name` par le nom du compartiment Amazon S3 et le nom du fichier que vous avez spécifiés à l'étape 2.
- Remplacez `region-name` par la région AWS que vous utilisez.
- Remplacez la valeur de `profile-name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.
- Remplacez `topic-ARN` par l'ARN de la rubrique Amazon SNS que vous avez créée à l'étape 3 de Configuration de Vidéo Amazon Rekognition.

- Remplacez `role-ARN` par l'ARN de la fonction de service IAM que vous avez créé à l'étape 7 de Configuration de Vidéo Amazon Rekognition.

Si vous accédez à la CLI sur un périphérique Windows, utilisez des guillemets doubles au lieu de guillemets simples et évitez les guillemets doubles internes par une barre oblique inverse (c'est-à-dire \) pour corriger les erreurs d'analyse que vous pourriez rencontrer. Pour un exemple, voir ci-dessous :

```
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
```

Après avoir exécuté l'exemple de code suivant, copiez l'`jobID` renvoyé et saisissez-le dans la commande suivante `GetPersonTracking` pour obtenir vos résultats, en remplaçant `job-id-number` par `jobID` que vous avez reçu précédemment :

```
aws rekognition get-person-tracking --job-id job-id-number
```

> **Note**
>
> Si vous avez déjà exécuté un exemple vidéo autre que Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK), le code à remplacer peut être différent.

3. Exécutez le code. Les identifiants uniques des personnes suivies s'affichent avec le moment, en secondes, où les chemins des personnes ont été suivis.

GetPersonTracking réponse à l'opération

`GetPersonTracking` renvoie un tableau `Persons` constitué d'objets PersonDetection qui contiennent des informations sur les personnes détectées dans la vidéo et sur les moments où leurs chemins sont suivis.

Vous pouvez trier `Persons` à l'aide du paramètre d'entrée `SortBy`. Spécifiez `TIMESTAMP` pour trier les éléments en fonction du moment où les chemins des personnes sont suivis dans la vidéo. Spécifiez `INDEX` pour effectuer un tri en fonction des personnes suivies dans la vidéo. Dans chaque ensemble de résultats relatifs à une personne, les éléments sont triés par ordre décroissant de fiabilité de la précision du suivi de chemin. Par défaut, `Persons` est renvoyé trié par `TIMESTAMP`. L'exemple suivant constitue la réponse JSON de `GetPersonDetection`. Les résultats sont triés en fonction du moment, en millisecondes depuis le début de la vidéo, où les chemins des personnes sont suivis dans la vidéo. Dans la réponse, notez les points suivants :

- Informations sur la personne : l'élément de tableau `PersonDetection` contient des informations sur la personne détectée. Par exemple, le moment où la personne a été détectée (`Timestamp`), la position de la personne dans le cadre vidéo au moment où elle a été détectée (`BoundingBox`), ainsi que le niveau de fiabilité de Vidéo Amazon Rekognition quant à la précision de la détection de la personne (`Confidence`).

  Les caractéristiques faciales ne sont pas retournées à chaque moment horodaté où le chemin de la personne est suivi. En outre, dans certaines circonstances, le corps d'une personne suivie peut ne pas être visible. Dans ce cas, seul l'emplacement du visage est renvoyé.
- Informations sur la pagination : l'exemple montre une page d'informations sur la détection de personne. Vous pouvez spécifier le nombre d'éléments de personne à renvoyer dans le paramètre d'entrée `MaxResults` pour `GetPersonTracking`. Si le nombre de résultats est supérieur à `MaxResults`, `GetPersonTracking` renvoie un jeton (`NextToken`) utilisé pour obtenir la page de résultats suivante. Pour de plus amples informations, veuillez consulter Obtenir les résultats de l'analyse de Vidéo Amazon Rekognition.
- Index : identifiant unique pour l'identification de la personne tout au long de la vidéo.
- Informations sur la vidéo : la réponse comprend des informations sur le format de la vidéo (`VideoMetadata`) dans chaque page d'informations renvoyée par `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
    }
}
```

Détection des équipements de protection individuelle

Amazon Rekognition peut détecter les équipements de protection individuelle (EPI) portés par des personnes sur une image. Vous pouvez utiliser ces informations pour améliorer les pratiques de sécurité au travail. Par exemple, vous pouvez utiliser la détection des EPI pour déterminer si les travailleurs d'un chantier de construction portent des couvre-chefs ou si le personnel médical porte des couvre-visages et des couvre-mains. L'image suivante montre certains types d'EPI qui peuvent être détectés.

Pour détecter le PPE dans une image, vous devez appeler l'DetectProtectiveEquipmentAPI et transmettre une image d'entrée. La réponse est une structure JSON qui inclut les éléments suivants.

- Les personnes détectées sur l'image.
- Parties du corps où l'EPI est porté (visage, tête, main gauche et main droite).
- Les types d'EPI détectés sur des parties du corps (couvre-visage, couvre-mains et couvre-tête).
- Pour les articles d'EPI détectés, un indicateur montrant si l'EPI couvre ou non la partie du corps correspondante.

Des cadres de délimitation sont renvoyés pour indiquer l'emplacement des personnes et des articles d'EPI détectés sur l'image.

Vous pouvez éventuellement demander un résumé des articles de protection individuelle et des personnes détectés sur une image. Pour de plus amples informations, veuillez consulter Présentation des EPI détectés dans une image.

> **Note**
>
> La détection Amazon Rekognition EPI n'effectue pas de reconnaissance faciale ni de comparaison faciale, et ne permet pas d'identifier les personnes détectées.

Types d'EPI

DetectProtectiveEquipmentdétecte les types d'EPI suivants. Si vous souhaitez détecter d'autres types d'EPI dans les images, pensez à utiliser Étiquettes personnalisées Amazon Rekognition pour créer un modèle personnalisé. Pour de plus amples informations, veuillez consulter Étiquettes personnalisées Amazon Rekognition.

Couvre-visage

`DetectProtectiveEquipment` peut détecter les masques faciaux courants tels que les masques chirurgicaux, les masques N95 et les masques en tissu.

Couvre-main

`DetectProtectiveEquipment` peut détecter les couvre-mains tels que les gants chirurgicaux et les gants de sécurité.

Couvre-tête

`DetectProtectiveEquipment` peut détecter les casques de sécurité et les casques.

L'API indique qu'une tête, une main ou un masque facial ont été détectés sur une image. L'API ne renvoie aucune information sur le type de protection spécifique. Par exemple, un « gant chirurgical » pour le type de couvre-main.

Fiabilité de détection des EPI

Amazon Rekognition prédit la présence d'EPI, de personnes et de parties du corps sur une image. L'API fournit un score (50-100) qui indique le niveau de confiance d'Amazon Rekognition quant à l'exactitude d'une prédiction.

> **Note**
>
> Si vous prévoyez d'utiliser l'opération `DetectProtectiveEquipment` pour prendre une décision qui a une incidence sur les droits, la confidentialité ou l'accès aux services d'une personne, nous vous recommandons de transmettre le résultat à un humain pour examen et validation avant de passer à l'action.

Présentation des EPI détectés dans une image

Vous pouvez éventuellement demander un résumé des articles de protection individuelle et des personnes détectés sur une image. Vous pouvez spécifier une liste des équipements de protection requis (couvre-visage, couvre-main ou couvre-chef) et un seuil de confiance minimum (par exemple, 80 %). La réponse inclut un résumé consolidé par identifiant (ID) par image des personnes possédant l'EPI requis, des personnes n'ayant pas l'EPI requis et des personnes pour lesquelles aucune détermination n'a pu être prise.

Le résumé vous permet de répondre rapidement à des questions telles que Combien de personnes ne portent pas de couvre-visage ? ou Est-ce que tout le monde porte un EPI ? Chaque personne détectée dans le résumé a un identifiant unique. Vous pouvez utiliser l'identifiant pour obtenir des informations telles que l'emplacement du cadre de délimitation d'une personne ne portant pas d'EPI.

> **Note**
>
> L'identifiant est généré de manière aléatoire sur la base d'une analyse par image et n'est pas cohérent entre les images ou les analyses multiples de la même image.

Vous pouvez résumer les couvre-visages, les couvre-chefs, les couvre-mains ou une combinaison de votre choix. Pour spécifier les types d'EPI requis, voir Spécifier les exigences de synthèse. Vous pouvez également spécifier un niveau de confiance minimum (50-100) qui doit être atteint pour que les détections soient incluses dans le résumé.

Pour plus d'informations sur la réponse récapitulative de `DetectProtectiveEquipment`, voir Comprendre la DetectProtectiveEquipment réponse.

Tutoriel : Création d'un AWS Lambda fonction qui détecte les images avec le PPE

Vous pouvez créer une AWS Lambda fonction qui détecte les équipements de protection individuelle (EPI) dans les images situées dans un compartiment Amazon S3. Consultez le GitHub référentiel d'exemples du SDK de AWS documentation pour ce didacticiel Java V2.

Comprendre l'API de détection des équipements de protection individuelle

Les informations suivantes décrivent l'DetectProtectiveEquipmentAPI. Pour obtenir un exemple de code, consultez Détection d'un équipement de protection individuelle dans une image.

Fourniture d'une image

Vous fournissez une image d'entrée (format JPG ou PNG) sous forme d'octets d'image ou en référençant une image stockée dans un compartiment S3.

Nous recommandons d'utiliser des images où le visage de la personne fait face à l'appareil photo.

Si votre image d'entrée n'est pas pivotée à 0 degré, nous vous recommandons de la faire pivoter à 0 degré avant de la soumettre à `DetectProtectiveEquipment`. Les images au format JPG peuvent contenir des informations d'orientation dans des métadonnées Exif (Exchangeable Image File Format). Vous pouvez utiliser ces informations pour écrire le code permettant de faire pivoter l'image. Pour en savoir plus, consultez Exif Version 2.32. Les images au format PNG ne contiennent aucune information d'orientation de l'image.

Pour transmettre une image depuis un compartiment Amazon S3, utilisez un utilisateur disposant d'au moins des AmazonS3ReadOnlyAccess privilèges. Utiliser un utilisateur ayant des privilèges `AmazonRekognitionFullAccess` pour appeler `DetectProtectiveEquipment.`

Dans l'exemple JSON d'entrée suivant, l'image est transmise dans un compartiment Amazon S3. Pour de plus amples informations, veuillez consulter Travail avec les images. L'exemple demande un résumé de tous les types d'EPI (couvre-tête, couvre-main et couvre-visage) avec un niveau de confiance de détection minimum (`MinConfidence`) de 80 %. Vous devez spécifier une valeur

`MinConfidence` comprise entre 50 et 100 % car `DetectProtectiveEquipment` ne renvoie des prédictions que lorsque le niveau de confiance de détection est compris entre 50 % et 100 %. Si vous spécifiez une valeur inférieure à 50 %, les résultats sont identiques si vous spécifiez une valeur de 50 %. Pour de plus amples informations, veuillez consulter Spécifier les exigences de synthèse.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "worker.jpg"
        }
    },
    "SummarizationAttributes": {
        "MinConfidence": 80,
        "RequiredEquipmentTypes": [
            "FACE_COVER",
            "HAND_COVER",
            "HEAD_COVER"
        ]
    }
}
```

Si vous disposez d'une grande collection d'images à traiter pour le processus, pensez à utiliser AWS Batch afin de traiter les appels de `DetectProtectiveEquipment` par lots en arrière-plan.

Spécifier les exigences de synthèse

Vous pouvez éventuellement utiliser le paramètre d'entrée `SummarizationAttributes` (ProtectiveEquipmentSummarizationAttributes) pour demander des informations récapitulatives sur les types d'EPI détectés dans une image.

Pour spécifier les types de PPE à résumer, utilisez le tableau `RequiredEquipmentTypes`. Dans le tableau, incluez un ou plusieurs des éléments suivants : `FACE_COVER`, `HAND_COVER` ou `HEAD_COVER`.

Utilisez le champ `MinConfidence` pour spécifier un niveau de confiance de détection minimal (50 à 100). Le résumé n'inclut pas les personnes, les parties du corps, les parties du corps couvertes et les articles d'EPI détectés avec un niveau de confiance inférieur à `MinConfidence`.

Pour plus d'informations sur la réponse en résumé de `DetectProtectiveEquipment` consultez Comprendre la DetectProtectiveEquipment réponse.

Comprendre la DetectProtectiveEquipment réponse

`DetectProtectiveEquipment` renvoie un tableau des personnes détectées dans l'image d'entrée. Pour chaque personne, des informations sur les parties du corps détectées et les articles d'EPI détectés sont renvoyées. Le JSON de l'image suivante d'un travailleur portant un couvre-chef, un couvre-main et un couvre-visage est le suivant.

Dans JSON, notez les éléments suivants :

- Personnes détectées : `Persons` ensemble de personnes détectées sur l'image (y compris les personnes ne portant pas d'EPI). `DetectProtectiveEquipment` peut détecter l'EPI sur un maximum de 15 personnes détectées sur une image. Chaque ProtectiveEquipmentPersonobjet du tableau contient un identifiant personnel, un cadre de délimitation pour la personne, des parties du corps détectées et des articles d'EPI détectés. La valeur `Confidence` dans `ProtectiveEquipmentPerson` indique le pourcentage de confiance d'Amazon Rekognition quant à la présence d'une personne dans le cadre de délimitation.
- Parties du corps : `BodyParts` ensemble de parties du corps (ProtectiveEquipmentBodyPart) détectées sur une personne (y compris les parties du corps non couvertes par un EPI). Chaque `ProtectiveEquipmentBodyPart` inclut le nom (`Name`) de la partie du corps détectée. `DetectProtectEquipment` peut détecter le visage, la tête, les parties du corps gauches et droites. Le champ `Confidence` dans `ProtectiveEquipmentBodyPart` indique le pourcentage de confiance d'Amazon Rekognition quant à la précision de détection de la partie du corps.

- Articles EPI : le tableau `EquipmentDetections` d'un objet `ProtectiveEquipmentBodyPart` contient un ensemble d'articles EPI détectés. Chaque EquipmentDetectionobjet contient les champs suivants.
  - `Type` : le type d'EPI détecté.
  - `BoundingBox` : un cadre de délimitation autour de l'EPI détecté.
  - `Confidence` : la confiance d'Amazon Rekognition quant au fait que le boîtier de sélection contient l'EPI détecté.
  - `CoversBodyPart` : indique si l'EPI détecté se trouve sur la partie du corps correspondante.

  Le CoversBodyPartchamp `Value` est une valeur booléenne qui indique si l'EPI détecté se trouve sur la partie du corps correspondante. Le champ `Confidence` indique le niveau de confiance dans la prédiction. Vous pouvez utiliser `CoversBodyPart` pour filtrer les cas où l'EPI détecté se trouve sur l'image, mais pas réellement sur la personne.

  > **Note**
  >
  > `CoversBodyPart` n'indique pas ou n'implique pas que la personne est correctement protégée par l'équipement de protection, ou que l'équipement de protection lui-même est correctement porté.

- Informations récapitulatives : `Summary` contient les informations récapitulatives spécifiées dans le paramètre d'entrée `SummarizationAttributes`. Pour de plus amples informations, veuillez consulter Spécifier les exigences de synthèse.

  `Summary`est un objet de type ProtectiveEquipmentSummaryqui contient les informations suivantes.
  - `PersonsWithRequiredEquipment` : tableau des identifiants de personnes pour lesquels chaque personne répond aux critères suivants.
    - La personne porte tous les équipements de protection individuelle spécifiés dans le paramètre d'entrée `SummarizationAttributes`.
    - Le niveau de `Confidence` pour la personne (`ProtectiveEquipmentPerson`), la partie du corps (`ProtectiveEquipmentBodyPart`), l'équipement de protection (`EquipmentDetection`) est égal ou supérieur au seuil de confiance minimum spécifié (`MinConfidence`).
    - La valeur de `CoversBodyPart` pour tous les articles de protection individuelle est vraie.
  - `PersonsWithoutRequiredEquipment` : tableau des identifiants des personnes répondant à l'un des critères suivants.

- Les valeurs `Confidence` pour la personne (`ProtectiveEquipmentPerson`), la partie du corps (`ProtectiveEquipmentBodyPart`) et la couverture des parties du corps (`CoversBodyPart`) sont supérieures au seuil de confiance minimum spécifié (`MinConfidence`), mais il manque un ou plusieurs EPI spécifiés (`SummarizationAttributes`) à la personne.
- La valeur de `CoversBodyPart` est fausse pour tout PPE spécifié (`SummarizationAttributes`) dont la valeur `Confidence` est supérieure au seuil de confiance minimum spécifié (`MinConfidence`). La personne possède également tous les équipements de protection individuelle spécifiés (`SummarizationAttributes`) et les valeurs `Confidence` pour la personne (`ProtectiveEquipmentPerson`), la partie du corps (`ProtectiveEquipmentBodyPart`) et l'équipement de protection (`EquipmentDetection`) sont supérieures ou égales au seuil de confiance minimum (`MinConfidence`).

- `PersonsIndeterminate` : tableau des identifiants des personnes détectées lorsque la valeur `Confidence` de la personne (`ProtectiveEquipmentPerson`), de la partie du corps (`ProtectiveEquipmentBodyPart`), de l'équipement de protection (`EquipmentDetection`) ou du `CoversBodyPart` booléen est inférieure au seuil de confiance minimum spécifié (`MinConfidence`).

Utilisez la taille du tableau pour obtenir le nombre d'un résumé particulier. Par exemple, la taille de `PersonsWithRequiredEquipment` indique le nombre de personnes détectées comme portant le type d'EPI spécifié.

Vous pouvez utiliser l'identifiant de la personne pour obtenir de plus amples informations sur une personne, telles que l'emplacement de la personne dans le cadre de délimitation. L'ID de personne correspond au champ ID d'un objet (`ProtectiveEquipmentPerson`) renvoyé dans `Persons` (tableau de `ProtectiveEquipmentPerson`). Vous pouvez ensuite obtenir le cadre de délimitation et d'autres informations à partir de l'objet `ProtectiveEquipmentPerson` correspondant.

```
{
   "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": []
    }
}
```

Détection d'un équipement de protection individuelle dans une image

Pour détecter la présence d'équipements de protection individuelle (EPI) sur des personnes sur une image, utilisez l'opération API DetectProtectiveEquipmentsans stockage.

Vous pouvez fournir l'image d'entrée sous la forme d'un tableau d'octets d'image (octets d'image encodés en base64) ou en tant qu'objet Amazon S3 en utilisant l'AWS SDK ou l' AWS Command Line Interface (AWS CLI). Ces exemples utilisent une image stockée dans un compartiment Amazon S3. Pour de plus amples informations, veuillez consulter Travail avec les images.

Pour détecter la présence d'EPI sur des personnes figurant sur une image

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur avec `AmazonRekognitionFullAccess` et autorisations `AmazonS3ReadOnlyAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Chargez une image (contenant une ou plusieurs personnes portant l'EPI) dans votre compartiment S3.

   Pour en savoir plus, consultez Chargement d'objets dans Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

3. Utilisez les exemples suivants pour appeler l'opération `DetectProtectiveEquipment`. Pour plus d'informations sur l'affichage des cadres de délimitation dans une image, consultez Affichage de cadres de délimitation.

Java

Cet exemple affiche des informations sur les articles de protection individuelle détectés sur les personnes détectées sur une image.

Remplacez la valeur de `amzn-s3-demo-bucket` par le nom du compartiment Amazon S3 qui contient votre image. Remplacez la valeur de `photo` par le nom de votre fichier image.

```
//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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l’exemple complet ici.

```
//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

Cette AWS CLI commande demande un résumé du PPE et affiche la sortie JSON pour l'opération `detect-protective-equipment` CLI.

Remplacez `amzn-s3-demo-bucket` par le nom de l'image d'un compartiment Amazon S3 qui contient une image. Remplacez `input.jpg` par le nom de l'image que vous souhaitez utiliser.

```
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']"
```

Cette AWS CLI commande affiche la sortie JSON pour l'opération `detect-protective-equipment` CLI.

Remplacez `amzn-s3-demo-bucket` par le nom de l'image d'un compartiment Amazon S3 qui contient une image. Remplacez `input.jpg` par le nom de l'image que vous souhaitez utiliser.

```
aws rekognition detect-protective-equipment \
  --image "S3Object={Bucket=amzn-s3-demo-bucket,Name=input.jpg}"
```

Python

Cet exemple affiche des informations sur les articles de protection individuelle détectés sur les personnes détectées sur une image.

Remplacez la valeur de `amzn-s3-demo-bucket` par le nom du compartiment Amazon S3 qui contient votre image. Remplacez la valeur de `photo` par le nom de votre fichier image. Remplacez la valeur de `profile_name` dans la ligne qui crée la session Rekognition par le nom de votre profil de développeur.

```
# 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()
```

Exemple : dessiner des cadres de délimitation autour des masques

Les exemples suivants vous montrent comment dessiner des cadres de délimitation autour des masques détectés sur des personnes. Pour un exemple utilisant AWS Lambda Amazon DynamoDB, consultez le référentiel d'exemples du SDK AWS de documentation. GitHub

Pour détecter les masques faciaux, vous utilisez l'opération d'API DetectProtectiveEquipmentnon liée au stockage. L'image est chargée à partir du système de fichiers local. Vous pouvez fournir à `DetectProtectiveEquipment` l'image d'entrée sous la forme d'un tableau d'octets d'image (octets d'image encodés en base64). Pour de plus amples informations, veuillez consulter Travail avec les images.

L'exemple affiche un cadre de délimitation autour des couvre-visages détectés. Le cadre de délimitation est vert si le couvre-visage couvre entièrement la partie du corps. Dans le cas contraire, un cadre de délimitation rouge s'affiche. À titre d'avertissement, un cadre de délimitation jaune s'affiche à l'intérieur du cadre de délimitation de masque si le niveau de confiance de détection est inférieur à la valeur de confiance spécifiée. Si aucun masque facial n'est détecté, un cadre de délimitation rouge est dessiné autour de la personne.

La sortie d'image est similaire à ce qui suit.

Pour afficher les cadres de délimitation sur les masques détectés

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Utilisez les exemples suivants pour appeler l'opération `DetectProtectiveEquipment`. Pour plus d'informations sur l'affichage des cadres de délimitation dans une image, consultez Affichage de cadres de délimitation.

   Java

   Dans la fonction `main`, modifiez les éléments suivants :

- La valeur du `photo` vers le chemin et du nom de fichier d'un fichier image local (PNG ou JPEG).
- La valeur du `confidence` correspondant au niveau de confiance souhaité (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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l’exemple complet ici.

```
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

Dans la fonction `main`, modifiez les éléments suivants :

- La valeur du `photo` vers le chemin et du nom de fichier d'un fichier image local (PNG ou JPEG).
- La valeur du `confidence` correspondant au niveau de confiance souhaité (50-100).
- Remplacez la valeur de `profile_name` dans la ligne qui crée la session Rekognition par le nom de votre profil de développeur.

```
#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

Dans l’exemple de CLI suivant, modifiez la valeur des arguments répertoriés ci-dessous :

- La valeur du `photo` vers le chemin et du nom de fichier d’un fichier image local (PNG ou JPEG).
- La valeur du `confidence` correspondant au niveau de confiance souhaité (50-100).
- Remplacez la valeur de `profile_name` dans la ligne qui crée la session Rekognition par le nom de votre profil de développeur.

```
                                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"]}"
```

Si vous accédez à la CLI sur un périphérique Windows, utilisez des guillemets doubles au lieu de guillemets simples et évitez les guillemets doubles internes par une barre oblique inverse (c'est-à-dire \) pour corriger les erreurs d'analyse que vous pourriez rencontrer. Par exemple, consultez ce qui suit :

```
                                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\"]}"
```

Reconnaissance de célébrités

Amazon Rekognition permet aux clients de reconnaître automatiquement des dizaines de milliers de personnalités connues sur des images et des vidéos à l'aide de la fonctionnalité machine learning. Les métadonnées fournies par l'API de reconnaissance des célébrités réduisent considérablement les efforts manuels répétitifs nécessaires pour étiqueter le contenu et le rendre facilement consultable.

En raison de la prolifération rapide du contenu d'images et de vidéos, les entreprises du secteur des médias ont souvent du mal à organiser, rechercher et utiliser leurs catalogues de médias à grande échelle. Les chaînes d'information et les diffuseurs sportifs ont souvent besoin de trouver rapidement des images et des vidéos afin de répondre à l'actualité et de créer une programmation pertinente. L'insuffisance des métadonnées complique ces tâches, mais avec Amazon Rekognition, vous pouvez étiqueter automatiquement de gros volumes de contenu nouveau ou archivé afin de faciliter les recherches pour un ensemble complet de célébrités internationales de renom, telles que des acteurs, des sportifs et des créateurs de contenu en ligne.

Le système de reconnaissance des célébrités Amazon Rekognition est conçu pour être utilisé exclusivement dans les cas où vous pensez qu'une célébrité est connue sur une image ou une vidéo. Pour plus d'informations sur la reconnaissance de visages n'appartenant pas à des célébrités, consultez Recherche de visages dans une collection.

> **Note**
>
> Si vous êtes une célébrité et que vous ne souhaitez pas être inclus dans cette fonctionnalité, contactez l'assistance AWS ou envoyez un e-mail à l'adresse `<rekognition-celebrity-opt-out@amazon.com>`.

Rubriques

- Comparaison entre reconnaissance de célébrités et recherche faciale
- Reconnaissance de célébrités sur une image
- Reconnaissance de célébrités dans une vidéo stockée
- Obtention d'informations sur une célébrité

Comparaison entre reconnaissance de célébrités et recherche faciale

Amazon Rekognition offre une fonctionnalité de reconnaissance des célébrités et de reconnaissance faciale. Ces fonctionnalités sont différentes dans leurs cas d'utilisation et leurs bonnes pratiques.

La reconnaissance des célébrités s'accompagne d'une formation préalable et de la capacité de reconnaître plus de cent mille célébrités dans des domaines tels que le sport, les médias, la politique et les affaires. Cette fonctionnalité est conçue pour vous aider à identifier un petit ensemble susceptible de contenir une célébrité en particulier dans de grands volumes d'images ou de vidéos. Elle n'est pas destinée à être utilisée pour mettre en correspondance les visages de différentes personnes qui ne sont pas des célébrités. Dans les cas où la précision de la mise en correspondance de célébrités est importante, nous vous recommandons également de faire appel à des opérateurs humains pour examiner cette plus petite quantité de contenu marqué afin de garantir un niveau de précision élevé et une application adéquate du jugement humain. La reconnaissance de célébrités ne doit pas être utilisée de manière à impacter les libertés individuelles de façon négative.

Au contraire, la reconnaissance faciale est une fonctionnalité plus générale qui vous permet de créer vos propres collections de visages avec vos propres vecteurs de visages afin de vérifier des identités ou rechercher une personne, qu'il s'agisse d'une célébrité ou non. La reconnaissance faciale peut être utilisée pour des applications telles que l'authentification ouvrant l'accès à un bâtiment, la sécurité publique et les réseaux sociaux. Dans toutes ces applications, il est recommandé d'utiliser les bonnes pratiques, les seuils de confiance appropriés (99 % inclus pour les cas d'utilisation de sécurité publique) et une vérification par un opérateur humain dans les cas où la précision de la mise en correspondance est importante.

Pour de plus amples informations, veuillez consulter Recherche de visages dans une collection.

Reconnaissance de célébrités sur une image

Pour reconnaître des célébrités sur des images et obtenir des informations supplémentaires sur les personnes qui ont été reconnues, utilisez l'opération d'API hors stockage RecognizeCelebrities. Par exemple, sur les réseaux sociaux ou dans les secteurs de l'actualité et du divertissement où la collecte rapide d'informations est cruciale, vous pouvez utiliser l'opération `RecognizeCelebrities` pour identifier jusqu'à 64 célébrités dans une image et renvoyer des liens vers des pages web de celles-ci, le cas échéant. Amazon Rekognition ne se souvient pas de l'image sur laquelle il a détecté une célébrité. Votre application doit stocker ces informations.

Si vous n'avez pas stocké les informations supplémentaires sur une célébrité que `RecognizeCelebrities` a renvoyées et que vous voulez éviter une nouvelle analyse d'une image afin d'obtenir ces informations, utilisez GetCelebrityInfo. Pour appeler `GetCelebrityInfo`, vous avez besoin de l'identifiant unique attribué par Amazon Rekognition à chaque célébrité. L'identifiant est renvoyé dans le cadre de la réponse `RecognizeCelebrities` pour chaque célébrité reconnue sur une image.

Si vous disposez d'une grande collection d'images à traiter pour la reconnaissance des célébrités, pensez à utiliser AWS Batch afin de traiter les appels de `RecognizeCelebrities` par lots en arrière-plan. Lorsque vous ajoutez une nouvelle image à votre collection, vous pouvez utiliser une AWS Lambda fonction permettant de reconnaître les célébrités en les appelant `RecognizeCelebrities` lorsque l'image est chargée dans un compartiment S3.

Appel RecognizeCelebrities

Vous pouvez fournir l'image d'entrée sous forme de tableau d'octets d'image (octets d'image codés en base64) ou sous forme d'objet Amazon S3, en utilisant le AWS Command Line Interface (AWS CLI) ou le SDK AWS. Au cours de la AWS CLI procédure, vous chargez une image au format .jpg ou .png dans un compartiment S3. Dans les procédures du AWS SDK, vous utilisez une image chargée depuis votre système de fichiers local. Pour plus d'informations sur les recommandations en matière d'image d'entrée, consultez Travail avec les images.

Pour exécuter cette procédure, vous avez besoin d'une image qui contient un ou plusieurs visages de célébrités.

Pour reconnaître des célébrités sur une image

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur avec `AmazonRekognitionFullAccess` et autorisations `AmazonS3ReadOnlyAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Utilisez les exemples suivants pour appeler l'opération `RecognizeCelebrities`.

   Java

   Cet exemple affiche des informations sur les célébrités qui sont détectées dans une image.

Remplacez la valeur de `photo` par le chemin et le nom d'un fichier image qui contient un ou plusieurs visages de célébrités.

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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
//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

Cette AWS CLI commande affiche la sortie JSON pour l'opération `recognize-celebrities` CLI.

Remplacez la valeur de `amzn-s3-demo-bucket` par le nom d'un compartiment S3; qui contient une image. Remplacez la valeur de `input.jpg` par le nom d'un fichier image qui contient un ou plusieurs visages de célébrités.

Remplacez la valeur de `profile_name` par le nom de votre profil de développeur.

```
aws rekognition recognize-celebrities \
  --image "S3Object={Bucket=amzn-s3-demo-bucket,Name=input.jpg}"
```

Si vous accédez à la CLI sur un périphérique Windows, utilisez des guillemets doubles au lieu de guillemets simples et évitez les guillemets doubles internes par une barre oblique inverse (c'est-à-dire \) pour corriger les erreurs d'analyse que vous pourriez rencontrer. Par exemple, consultez ce qui suit :

```
                                    aws rekognition recognize-celebrities --
image \
                                    "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --profile profile-name
```

Python

Cet exemple affiche des informations sur les célébrités qui sont détectées dans une image.

Remplacez la valeur de `photo` par le chemin et le nom d'un fichier image qui contient un ou plusieurs visages de célébrités.

Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

Cet exemple affiche des informations sur les célébrités qui sont détectées dans une image.

Remplacez la valeur de `photo` par le chemin et le nom d'un fichier image qui contient un ou plusieurs visages de célébrités. Remplacez la valeur de `amzn-s3-demo-bucket` par le nom du compartiment S3 qui contient le fichier image. Remplacez la valeur `REGION` par le nom de la région associée à votre utilisateur. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
// 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

Cet exemple affiche des informations sur les célébrités qui sont détectées dans une image.

Remplacez la valeur de `photo` par le chemin et le nom d'un fichier image qui contient un ou plusieurs visages de célébrités (format .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. Enregistrez la valeur de l'un des ID de célébrité qui sont affichés. Vous en aurez besoin pour Obtention d'informations sur une célébrité.

RecognizeCelebrities demande d'opération

La valeur d'entrée de `RecognizeCelebrities` est une image. Dans cet exemple, l'image est transmise sous forme d'octets d'image. Pour de plus amples informations, veuillez consulter Travail avec les images.

```
{
    "Image": {
        "Bytes": "/AoSiyvFpm....."
    }
}
```

RecognizeCelebrities réponse à l'opération

L'exemple ci-dessous représente l'entrée et la sortie JSON pour `RecognizeCelebrities`.

`RecognizeCelebrities` renvoie un tableau de célébrités reconnues et un tableau de visages non reconnus. Dans l'exemple, notez les éléments suivants :

- Célébrités reconnues – `Celebrities` Liste des célébrités reconnues. Chaque objet Célébrité de la liste contient le nom de la célébrité et une liste d'URL pointant vers le contenu associé, par exemple le lien IMDB ou Wikidata de la célébrité. Amazon Rekognition ComparedFacerenvoie un objet que votre application peut utiliser pour déterminer où se trouve le visage de la célébrité sur l'image, ainsi qu'un identifiant unique pour la célébrité. Utilisez l'identifiant unique afin d'extraire par la suite les informations relatives à la célébrité à l'aide de l'opération d'API GetCelebrityInfo.
- Visages non reconnus – `UnrecognizedFaces` Liste des visages ne correspondant à aucune célébrité connue. Chaque objet ComparedFace de la liste contient un cadre de sélection (ainsi que d'autres informations) que vous pouvez utiliser pour rechercher le visage sur l'image.

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

Reconnaissance de célébrités dans une vidéo stockée

La reconnaissance de célébrités Vidéo Amazon Rekognition dans des vidéos stockées est une opération asynchrone. Pour reconnaître des célébrités dans une vidéo enregistrée, utilisez cette option StartCelebrityRecognitionpour démarrer l'analyse vidéo. Vidéo Amazon Rekognition publie l'état d'achèvement de l'opération d'analyse vidéo dans une rubrique Amazon Simple Notification Service. Si l'analyse vidéo aboutit, appelez GetCelebrityRecognition pour obtenir les résultats de cette opération. Pour plus d'informations sur le démarrage de l'analyse vidéo et l'obtention des résultats, consultez Appeler les opérations de Vidéo Amazon Rekognition.

Cette procédure s'appuie sur le code figurant dans Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK), qui utilise une file d'attente Amazon SQS pour obtenir le statut d'achèvement d'une demande d'analyse vidéo. Pour exécuter cette procédure, vous avez besoin d'un fichier vidéo qui contient un ou plusieurs visages de célébrités.

Pour détecter des célébrités dans une vidéo stockée dans un compartiment S3 (SDK)

1. Effectuez une Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK).
2. Ajoutez le code suivant à la classe `VideoDetect` que vous avez créée à l'étape 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);

       }
```

Dans la fonction `main`, remplacez la ligne :

```
       StartLabelDetection(amzn-s3-demo-bucket, video);

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

avec :

```
        StartCelebrityDetection(amzn-s3-demo-bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetCelebrityDetectionResults();
```

Java V2

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
//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
```

Dans la fonction `main`, remplacez les lignes :

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

avec :

```
    analyzer.StartCelebrityDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetCelebrityDetectionResults()
```

Node.JS

Dans l'exemple de Node.Js code suivant, remplacez la valeur de `amzn-s3-demo-bucket` par le nom du compartiment S3 contenant votre vidéo et la valeur de `videoName` par le nom du fichier vidéo. Vous devez également remplacer la valeur de `roleArn` par l'ARN associé à votre fonction du service IAM. Enfin, remplacez la valeur de `region` par le nom de la région d'exploitation associée à votre compte. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
//Copyright 2018 Amazon.com, Inc. or its 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",&TCX5-2025-waiver;
      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

Exécutez la AWS CLI commande suivante pour commencer à détecter des célébrités dans une vidéo.

```
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
```

Mettez à jour les valeurs suivantes :

- Remplacez `amzn-s3-demo-bucket` et `video-name` par le nom du compartiment Amazon S3 et le nom du fichier que vous avez spécifiés à l'étape 2.
- Remplacez `region-name` par la région AWS que vous utilisez.
- Remplacez la valeur de `profile-name` par le nom de votre profil de développeur.
- Remplacez `topic-ARN` par l'ARN de la rubrique Amazon SNS que vous avez créée à l'étape 3 de Configuration de Vidéo Amazon Rekognition.
- Remplacez `role-ARN` par l'ARN de la fonction de service IAM que vous avez créé à l'étape 7 de Configuration de Vidéo Amazon Rekognition.

Si vous accédez à la CLI sur un périphérique Windows, utilisez des guillemets doubles au lieu de guillemets simples et évitez les guillemets doubles internes par une barre oblique inverse (c'est-à-dire \) pour corriger les erreurs d'analyse que vous pourriez rencontrer. Pour un exemple, voir ci-dessous :

```
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
```

Après avoir exécuté l'exemple de code suivant, copiez l'`jobID` renvoyé et saisissez-le dans la commande suivante `GetCelebrityRecognition` pour obtenir vos résultats, en remplaçant `job-id-number` par `jobID` que vous avez reçu précédemment :

```
aws rekognition get-celebrity-recognition --job-id job-id-number --profile
 profile-name
```

> **Note**
>
> Si vous avez déjà exécuté un exemple vidéo autre que Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK), le code à remplacer peut être différent.

3. Exécutez le code. Des informations concernant les célébrités reconnues dans la vidéo s'affichent.

GetCelebrityRecognition réponse à l'opération

Voici un exemple de réponse JSON : La réponse inclut les éléments suivants :

- Célébrités reconnues : `Celebrities` est un tableau des célébrités et des moments auxquels elles ont été reconnues dans une vidéo. Un objet CelebrityRecognition existe pour chaque occurrence de reconnaissance de la célébrité dans la vidéo. Chaque `CelebrityRecognition` contient des informations sur une célébrité reconnue (CelebrityDetail) et l'heure (`Timestamp`) à laquelle elle a été reconnue dans la vidéo. `Timestamp` est mesurée en millisecondes à partir du début de la vidéo.
- CelebrityDetail— Contient des informations sur une célébrité reconnue. Cette partie comprend le nom de la célébrité (`Name`), un identifiant (`ID`) le genre connu de la célébrité (`KnownGender`), et une liste d'URL pointant vers du contenu associé (`Urls`). Cela inclut également le niveau de confiance d'Amazon Rekognition Video dans la précision de la reconnaissance, ainsi que des informations sur le visage de la célébrité. FaceDetail Si vous avez besoin d'obtenir le contenu associé ultérieurement, vous pouvez l'utiliser `ID` avec get CelebrityInfo.
- VideoMetadata— Informations sur la vidéo analysée.

```
{
    "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
    }
}
```

Obtention d'informations sur une célébrité

Dans ces procédures, vous pouvez obtenir des informations sur les célébrités à l'aide de l'opération get CelebrityInfo API. La célébrité est identifiée par son ID qui a été renvoyé par un appel précédent à RecognizeCelebrities.

Appel GetCelebrityInfo

Ces procédures exigent également l'ID de célébrité pour une célébrité connue de Amazon Rekognition. Utilisez l'ID de célébrité que vous notez à l'étape Reconnaissance de célébrités sur une image.

Pour obtenir des informations sur la célébrité (SDK)

1. Si vous ne l'avez pas déjà fait :
   a. Créez ou mettez à jour un utilisateur avec `AmazonRekognitionFullAccess` et autorisations `AmazonS3ReadOnlyAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.
   b. Installez et configurez les AWS SDK AWS CLI et. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.
2. Utilisez les exemples suivants pour appeler l'opération `GetCelebrityInfo`.

   Java

   Cet exemple affiche le nom d'une célébrité et les informations associées.

Remplacez `id` par l'un des ID de célébrités affichés dans Reconnaissance de célébrités sur une 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.)

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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
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

Cette AWS CLI commande affiche la sortie JSON pour l'opération `get-celebrity-info` CLI. Remplacez `ID` par l'un des ID de célébrités affichés dans Reconnaissance de célébrités sur une image. Remplacez la valeur de `profile-name` par le nom de votre profil de développeur.

```
aws rekognition get-celebrity-info --id celebrity-id --profile profile-name
```

Python

Cet exemple affiche le nom d'une célébrité et les informations associées.

Remplacez `id` par l'un des ID de célébrités affichés dans Reconnaissance de célébrités sur une image. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

Cet exemple affiche le nom d'une célébrité et les informations associées.

Remplacez `id` par l'un des ID de célébrités affichés dans Reconnaissance de célébrités sur une 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.)

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 demande d'opération

L'exemple ci-dessous représente l'entrée et la sortie JSON pour `GetCelebrityInfo`.

La valeur d'entrée de `GetCelebrityInfo` est l'ID de la célébrité en question.

```
{
    "Id": "nnnnnnn"
}
```

GetCelebrityInfo réponse à l'opération

`GetCelebrityInfo` renvoie un tableau (`Urls`) de liens qui donnent accès à des informations sur la célébrité demandée.

```
{
    "Name": "Celebrity Name",
    "Urls": [
        "www.imdb.com/name/nmnnnnnnn"
    ]
}
```

Modération du contenu

Vous pouvez utiliser Amazon Rekognition pour détecter le contenu inapproprié, indésirable ou offensant. Vous pouvez utiliser les API de modération Rekognition sur les réseaux sociaux, les médias audiovisuels, la publicité et le commerce électronique afin de créer une expérience utilisateur plus sûre, garantir la sécurité de la marque aux annonceurs et vous conformer aux réglementations locales et internationales.

Aujourd'hui, de nombreuses entreprises s'appuient entièrement sur des modérateurs humains pour examiner le contenu généré par des tiers ou des utilisateurs, tandis que d'autres se contentent de réagir aux plaintes des utilisateurs pour supprimer les images, publicités ou vidéos offensantes ou inappropriées. Cependant, les modérateurs humains ne peuvent à eux seuls s'adapter pour répondre à ces besoins avec une qualité ou une rapidité suffisantes, ce qui entraîne une mauvaise expérience utilisateur, des coûts élevés pour faire face au volume, voire une perte de réputation de marque. En utilisant Rekognition pour la modération des images et des vidéos, les modérateurs humains peuvent examiner un ensemble de contenu beaucoup plus restreint, généralement de 1 à 5 % du volume total, déjà signalé par la fonctionnalité machine learning. Cela leur permet de se concentrer sur des activités plus intéressantes tout en bénéficiant d'une couverture de modération complète pour une fraction de leurs coûts actuels. Pour configurer les effectifs humains et effectuer des tâches de révision humaines, vous pouvez utiliser Amazon Augmented AI, qui est déjà intégré à Rekognition.

Vous pouvez améliorer la précision du modèle deep learning de modération grâce à la fonctionnalité de modération personnalisée. Avec la modération personnalisée, vous entraînez un adaptateur de modération personnalisé en téléchargeant vos images et en les annotant. L'adaptateur entraîné peut ensuite être fourni à l'DetectModerationLabelsopération pour améliorer ses performances sur vos images. Pour plus d'informations, consultez Amélioration de la précision grâce à la modération personnalisée.

Étiquettes prises en charge par les opérations de modération du contenu de Rekognition

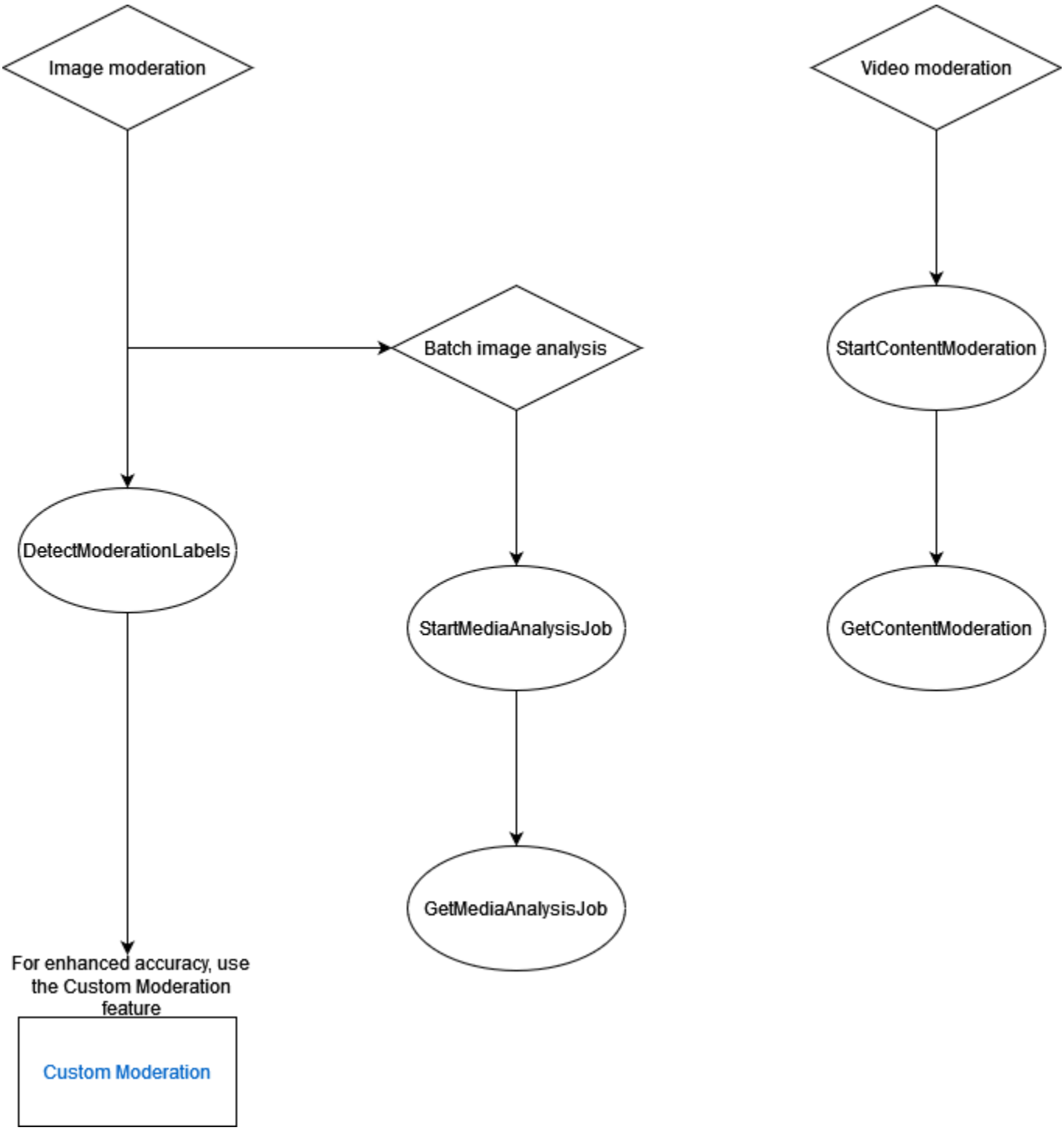
- Pour télécharger la liste des labels de modération, cliquez ici.

Rubriques

- Utilisation des API de modération d'image et de vidéo
- Test de la version 7 de Content Moderation et transformation de la réponse de l'API
- Détection d'images inappropriées

- Détection des vidéos stockées inappropriées
- Amélioration de la précision grâce à la modération personnalisée
- Vérification de contenu inappropriée avec Amazon Augmented AI

Le schéma suivant montre l'ordre des opérations d'appel, en fonction de vos objectifs d'utilisation des composants image ou vidéo de Content Moderation :



Utilisation des API de modération d'image et de vidéo

Dans l'API Amazon Rekognition Image, vous pouvez détecter le contenu inapproprié, indésirable ou offensant de manière synchrone et DetectModerationLabelsasynchrone en utilisant et en utilisant des opérations. StartMediaAnalysisJobGetMediaAnalysisJob Vous pouvez utiliser l'API Amazon

Rekognition Video pour détecter ce type de contenu de manière asynchrone à l'aide des opérations et. StartContentModerationGetContentModeration

Catégories d'étiquette

Amazon Rekognition utilise une taxonomie hiérarchique à trois niveaux pour étiqueter les catégories de contenu inapproprié, indésirable ou offensant. Chaque étiquette de niveau de taxonomie 1 (L1) possède un certain nombre d'étiquettes de niveau de taxonomie 2 (L2), et certaines étiquettes de niveau 2 peuvent avoir des étiquettes de niveau de taxonomie 3 (L3). Cela permet une classification hiérarchique du contenu.

Pour chaque étiquette de modération détectée, l'API renvoie également le`TaxonomyLevel`, qui contient le niveau (1, 2 ou 3) auquel appartient l'étiquette. Par exemple, une image peut être étiquetée conformément à la catégorisation suivante :

L1 : Non-Explicit Nudité des parties intimes et baisers, L2 : Non-Explicit Nudité, L3 : Nudité implicite.

> **Note**
>
> Nous vous recommandons d'utiliser les catégories L1 ou L2 pour modérer votre contenu et d'utiliser les catégories L3 uniquement pour supprimer les concepts spécifiques que vous ne souhaitez pas modérer (c'est-à-dire pour détecter le contenu que vous ne souhaitez peut-être pas classer comme contenu inapproprié, indésirable ou offensant en fonction de votre politique de modération).

Le tableau suivant montre les relations entre les niveaux de catégorie et les libellés possibles pour chaque niveau. Pour télécharger la liste des labels de modération, cliquez ici.

Top-Level Catégorie (L1)	Second-Level Catégorie (L2)	Third-Level Catégorie (L3)	Définitions
Explicite	Nudité explicite (Explicit Nudity)	Organes génitaux masculins exposés	Les organes génitaux masculins humains, y compris le pénis (qu'il soit en érection ou flasque), le scrotum et les poils

	pubiens visibles. Ce terme est applicable dans des contextes impliquant une activité sexuelle ou tout contenu visuel où les organes génitaux masculins sont affichés complètem ent ou partiellement.
Organes génitaux féminins exposés	Parties externes du système reproducteur féminin, comprenan t la vulve, le vagin et tout poil pubien observable. Ce terme est applicabl e dans les scénarios impliquant une activité sexuelle ou tout contenu visuel dans lequel ces aspects de l'anatomie féminine sont présentés complètement ou partiellement.

	Fesses ou anus exposés	Fesses ou anus humains, y compris les cas où les fesses sont nues ou lorsqu'el les sont visibles à travers des vêtements transparents. La définition s'appliqu e spécifiquement aux situations où les fesses ou l'anus sont directement et complètement visibles, à l'exclusi on des scénarios où toute forme de sous-vêtement ou de vêtement fournit une couverture complète ou partielle.
	Mamelon féminin exposé	Mamelons féminins humains, y compris les aéroles (zone entourant les mamelons) et les mamelons entièreme nt visibles et partielle ment visibles.

	Activité sexuelle explicite	N/A	Représentation d'actes sexuels réels ou simulés comprenant les rapports sexuels humains, le sexe oral, ainsi que la stimulation génitale masculine et la stimulation génitale féminine par d'autres parties du corps et objets. Le terme inclut également l'éjaculation ou les sécrétions vaginales sur des parties du corps ainsi que les pratiques érotiques ou les jeux de rôle impliquant le bondage, la discipline, la domination et la soumission, ainsi que le sadomasochisme.
	Jouets sexuels	N/A	Objets ou appareils utilisés pour la stimulation ou le plaisir sexuels, par exemple un gode, un vibromasseur, un plug anal, des battements, etc.

Non-Explicit Nudité des parties intimes et baisers	Non-Explicit Nudité	Dos nu	Partie postérieure humaine où la majeure partie de la peau est visible du cou à l'extrémité de la colonne vertébrale. Ce terme ne s'applique pas lorsque le dos de la personne est partiellement ou totalement occlus.
		Mamelon masculin exposé	Mamelons masculins humains, y compris les mamelons partiellement visibles.
		Fesses partiellement exposées	Fesses humaines partiellement exposées. Ce terme inclut une région partiellement visible des fesses ou des fessiers due à des vêtements courts, ou une partie supérieure partiellement visible de la fente anale. Le terme ne s'applique pas aux cas où les fesses sont entièrement nues.

Poitrine féminine partiellement exposée	Sein féminin humain partiellement exposé où une partie du sein de la femme est visible ou découverte sans révéler la totalité du sein. Ce terme s'applique lorsque la région du pli intérieur du sein est visible ou lorsque le pli inférieur du sein est visible lorsque le mamelon est entièrement recouvert ou occlus.
Nudité implicite	Personne nue, aux seins nus ou sans fond, mais dont les parties intimes, telles que les fesses, les tétons ou les organes génitaux, sont couvertes, occluses ou ne sont pas entièrement visibles.

	Parties intimes obstruées	Mamelon féminin obstrué	Représentation visuelle d'une situation dans laquelle les mamelons d'une femme sont recouverts de vêtements ou de revêtements opaques, mais leurs formes sont clairement visibles.
		Organes génitaux masculins obstrués	Représentation visuelle d'une situation dans laquelle les organes génitaux ou le pénis d'un homme sont recouverts de vêtements ou de revêtements opaques, mais dont la forme est clairement visible. Ce terme s'applique lorsque les organes génitaux obstrués sur l'image sont en gros plan.
	Baiser sur les lèvres	N/A	Représentation des lèvres d'une personne entrant en contact avec celles d'une autre personne.

Maillots de bain ou sous-vêtements	Maillots de bain ou sous-vêtements féminins	N/A	Vêtements humains pour maillots de bain féminins (maillots de bain une pièce, bikinis, tankinis, etc.) et sous-vêtements féminins (soutiens-gorge, culottes, slips, lingerie, strings, etc.)
	Maillots de bain ou sous-vêtements pour hommes	N/A	Vêtements humains pour maillots de bain masculins (par exemple, maillots de bain, shorts de bain, slips de bain, etc.) et sous-vête ments masculins (par exemple, slips, boxers, etc.)

Violence	Armes	N/A	Instruments ou dispositifs utilisés pour blesser ou endommager des êtres vivants, des structures ou des systèmes. Cela inclut les armes à feu (par exemple, les fusils, les mitrailleuses, etc.), les armes tranchantes (par exemple, les épées, les couteaux, etc.), les explosifs et les munitions (par exemple, les missiles, les bombes, les balles, etc.).
	Violence graphique	Violence avec des armes	L'utilisation d'armes pour blesser, endommager, blesser ou tuer soi-même, d'autres personnes ou des biens.

Violence physique	Le fait de porter atteinte à d'autres personnes ou à des biens (par exemple, frapper, se battre, s'arrache r les cheveux, etc.) ou tout autre acte de violence impliquant une foule ou plusieurs personnes.
Self-Harm	Le fait de se blesser soi-même, souvent en coupant des parties du corps telles que les bras ou les jambes, où les coupures sont généralement visibles.
Du sang et du sang	Représentation visuelle de la violence à l'égard d'une personne, d'un groupe d'individ us ou d'animaux , impliquant des blessures ouvertes, des effusions de sang et des parties du corps mutilées.

		Explosions et explosions	Représentation d'une explosion violente et destructrice de flammes intenses accompagnée d'une épaisse fumée ou de poussière et de fumée s'échappant du sol.
Perturbant visuellement	Mort et émaciation	Corps décharnés	Des corps humains extrêmement maigres et sous-alimentés avec une atrophie physique sévère et un épuisement des tissus musculaires et adipeux.
		Cadavres	Des cadavres humains sous forme de corps mutilés, de cadavres suspendus ou de squelettes.
	Collisions	Accident aérien	Incidents impliquant des véhicules aériens, tels que des avions, des hélicoptères ou d'autres véhicules volants, entraînant des dommages, des blessures ou la mort. Ce terme s'applique lorsque des parties des véhicules aériens sont visibles.

Drogues et tabac	Produits	Pilules	Petites tables ou capsules solides, souvent rondes ou ovales. Ce terme s'applique aux pilules présentées séparément, dans un flacon ou dans un emballage transparent et ne s'applique pas à la représentation visuelle d'une personne prenant des pilules.
	Matériel et usage des drogues et du tabac	Smoking	Le fait d'inhaler, d'expirer et d'allumer des substances brûlantes, notamment des cigarettes, des cigares, des cigarettes électroniques, du narguilé ou des joints.
Alcool	Consommation d'alcool	Boire	Le fait de boire des boissons alcoolisées dans des bouteilles ou des verres d'alcool ou d'alcool.

	Boissons alcoolisées	N/A	Gros plan d'une ou de plusieurs bouteilles d'alcool ou de spiritueux, de verres ou de mugs contenant de l'alcool ou de l'alcool, et de verres ou mugs contenant de l'alcool ou de l'alcool tenus par un individu. Ce terme ne s'applique pas à une personne qui boit dans des bouteilles ou des verres d'alcool ou d'alcool.
Gestes grossiers	Doigt d’honneur	N/A	La représentation visuelle d'un geste de la main avec le majeur est étendue vers le haut tandis que les autres doigts sont rabattus vers le bas.
Jeu	N/A	N/A	Le fait de participer à des jeux de hasard pour avoir la chance de gagner un prix dans les casinos, par exemple aux cartes à jouer, aux blackjacks, à la roulette, aux machines à sous dans les casinos, etc.

Symboles de haine	Parti nazi	N/A	Représentation visuelle de symboles, de drapeaux ou de gestes associés au parti nazi.
	Suprématie blanche	N/A	Représentation visuelle de symboles ou de vêtements associés au Ku Klux Klan (KKK) et d'images avec des drapeaux confédérés.
	Extrémiste	N/A	Images contenant des drapeaux de groupes extrémistes et terrorist es.

Toutes les étiquettes de la catégorie L2 ne disposent pas d'une étiquette compatible dans la catégorie L3. De plus, les étiquettes L3 sous « Produits » et « Accessoires et usage des drogues et du tabac » ne sont pas exhaustives. Ces étiquettes L2 couvrent des concepts autres que les étiquettes L3 mentionnées et dans de tels cas, seules les étiquettes L2 sont renvoyées dans la réponse de l'API.

Vous pouvez déterminer l'adéquation d'une image pour votre application. Par exemple, les images de nature suggestive peuvent être acceptables, ce qui n'est pas le cas de celles représentant la nudité. Pour filtrer des images, utilisez le tableau d'étiquettes ModerationLabel renvoyé par `DetectModerationLabels` (images) et par `GetContentModeration` (vidéos).

Type de contenu

L'API peut également identifier le type de contenu animé ou illustré, et le type de contenu est renvoyé dans le cadre de la réponse :

- Le contenu animé inclut les jeux vidéo et les animations (par exemple, dessins animés, bandes dessinées, mangas, anime).

- Le contenu illustré inclut le dessin, la peinture et les croquis.

Fiabilité

Vous pouvez définir le seuil de confiance utilisé par Amazon Rekognition pour détecter du contenu inapproprié en spécifiant le paramètre d'entrée `MinConfidence`. Les étiquettes de contenu inapproprié détectées avec une fiabilité plus faible que `MinConfidence` ne sont pas renvoyées.

La spécification d'une valeur inférieure à 50 % est susceptible de renvoyer un grand nombre de résultats faussement positifs (c'est-à-dire un rappel plus élevé, une précision moindre). `MinConfidence` En revanche, le fait de spécifier une valeur `MinConfidence` supérieure à 50 % est susceptible de donner un nombre inférieur de résultats faussement positifs (c'est-à-dire un taux de rappel plus faible, une précision plus élevée). Si vous ne spécifiez pas de valeur pour `MinConfidence`, Amazon Rekognition renvoie les étiquettes de contenu inapproprié détecté avec au moins 50 % de confiance.

Le tableau `ModerationLabel` contient les étiquettes des catégories précédentes et une estimation de la précision du contenu reconnu. Une étiquette de premier niveau est renvoyée avec les étiquettes de second niveau qui ont été identifiées. Par exemple, Amazon Rekognition peut retourner « Nudité explicite » avec un haut degré de fiabilité comme étiquette de premier niveau. Cela peut suffire à vos besoins de filtrage. Toutefois, le cas échéant, vous pouvez utiliser le score de confiance d'une étiquette de deuxième niveau (par exemple, « Nudité graphique masculine ») pour obtenir un filtrage plus affiné. Pour obtenir un exemple, consultez Détection d'images inappropriées.

Gestion des versions

Image Amazon Rekognition et Vidéo Amazon Rekognition renvoient tous deux la version du modèle de détection de modération utilisée pour détecter les contenus inappropriés (`ModerationModelVersion`).

Tri et agrégation

Lorsque vous récupérez des résultats avec GetContentModeration, vous pouvez les trier et les agréger.

Ordre de tri : le tableau d'étiquettes renvoyé est trié par heure. Pour trier par étiquette, spécifiez `NAME` dans le paramètre d'entrée `SortBy` pour `GetContentModeration`. Si l'étiquette apparaît plusieurs fois dans la vidéo, il existe plusieurs instances de l'élément `ModerationLabel`.

Informations sur l'étiquette : l'élément du ModerationLabels tableau contient un `ModerationLabel` objet, qui contient à son tour le nom de l'étiquette et la confiance d'Amazon Rekognition dans l'exactitude de l'étiquette détectée. L'horodatage est l'heure à laquelle le `ModerationLabel` a été détecté, défini comme le nombre de millisecondes écoulées depuis le début de la vidéo. Pour les résultats agrégés par vidéo `SEGMENTS`, les structures `StartTimestampMillis`, `EndTimestampMillis` et `DurationMillis` sont renvoyées, qui définissent respectivement l'heure de début, l'heure de fin et la durée d'un segment.

Agrégation : spécifie la manière dont les résultats sont agrégés lorsqu'ils sont renvoyés. La valeur par défaut est d'agréger par `TIMESTAMPS`. Vous pouvez également choisir d'agréger par `SEGMENTS`, ce qui permet d'agréger les résultats sur une période donnée. Seules les étiquettes détectées lors des segments sont renvoyées.

Statuts des adaptateurs de modération personnalisés

Les adaptateurs de modération personnalisés peuvent avoir l'un des statuts suivants : TRAINING_IN_PROGRESS, TRAINING_COMPLETED, TRAINING_FAILED, DELETING, DEPRECATED ou EXPIRED. Pour une explication complète de l'état de ces adaptateurs, consultez la section Gestion des adaptateurs.

> **Note**
>
> Amazon Rekognition n'est pas une autorité en matière de contenu inapproprié ou offensant, et ne prétend en aucun cas être un filtre exhaustif en la matière. De plus, les API de modération d'image et de vidéo ne détectent pas si une image inclut du contenu illégal, comme le CSAM.

Test de la version 7 de Content Moderation et transformation de la réponse de l'API

Rekognition a mis à jour le modèle d'apprentissage automatique pour les composants image vidéo de la fonctionnalité de détection d'étiquettes Content Moderation de la version 6.1 à la version 7. Cette mise à jour a amélioré la précision globale, introduit plusieurs nouvelles catégories et en a modifié d'autres.

Si vous êtes un utilisateur vidéo actuel de la version 6.1, nous vous recommandons de prendre les mesures suivantes pour effectuer une transition fluide vers la version 7 :

1. Téléchargez et utilisez un SDK privé AWS (voir lethe section called “AWS SDK et guide d'utilisation pour la modération de contenu version 7”) pour appeler l' StartContentModeration API.
2. Consultez la liste mise à jour des labels et des scores de confiance renvoyés dans la réponse ou la console de l'API. Ajustez la logique de post-traitement de votre application en conséquence si nécessaire.
3. Votre compte restera en version 6.1 jusqu'au 13 mai 2024. Si vous souhaitez utiliser la version 6.1 au-delà du 13 mai 2024, contactez l'équipe de support AWS avant le 30 avril 2024 pour demander une extension. Nous pouvons étendre votre compte pour qu'il reste sur la version 6.1 jusqu'au 10 juin 2024. Si nous n'avons pas de nouvelles de vous avant le 30 avril 2024, votre compte sera automatiquement migré vers la version 7.0 à compter du 13 mai 2024.

AWS SDK et guide d'utilisation pour la modération de contenu version 7

Téléchargez le SDK correspondant au langage de développement que vous avez choisi et consultez le guide de l'utilisateur approprié.

Lien vers le SDK	Guide d'installation et d'utilisation
Java-1.X	Manuel - Java 1.pdf
Java-2.X	Manuel - Java 2.pdf
JavaScript v2	Guide - JavaScript v2.pdf
JavaScript v3	Guide - JavaScript v3.pdf
Python	Guide - Python et AWS CLI.pdf
Ruby	Guide - RubyV3.pdf
go_v1	Guide - GO V1.pdf
go v2	Guide - GO V2.pdf
DotNet	Guide -. NET.pdf
php	Guide - PHP.pdf

Mappages d'étiquettes pour les versions 6.1 à 7

La version 7 de modération du contenu a ajouté de nouvelles catégories d'étiquettes et modifié les noms d'étiquettes existants. Référez-vous au tableau de taxonomie disponible à l'the section called “Catégories d’étiquette”adresse suivante pour décider comment mapper les étiquettes 6.1 à 7 étiquettes.

Vous trouverez des exemples de mappage d'étiquettes dans la section suivante. Nous vous recommandons de consulter ces mappages et les définitions d'étiquettes avant d'effectuer les mises à jour nécessaires en fonction de la logique de post-traitement de votre application.

Schéma de mappage L1

Si vous utilisez une logique de post-traitement qui filtre uniquement sur la catégorie de niveau supérieur (L1) (telle que`Explicit Nudity`,, `Violence` etc.)`Suggestive`, reportez-vous au tableau ci-dessous pour mettre à jour votre code.

V6.1 L1	V7 L1
Nudité explicite (Explicit Nudity)	Explicite
Suggestif (Suggestive)	Non-Explicit Nudité des parties intimes et baisers
	Maillots de bain ou sous-vêtements
Violence	Violence
Perturbant visuellement	Perturbant visuellement
Gestes grossiers	Gestes grossiers
Médicaments	Drogues et tabac
Tabac	Drogues et tabac
Alcool	Alcool
Jeu	Jeu
Symboles de haine	Symboles de haine

Schéma de mappage L2

Si vous utilisez une logique de post-traitement qui filtre à la fois sur les catégories L1 et L2 (telles que`Explicit Nudity / Nudity, Suggestive / Female Swimwear Or Underwear`, `Violence / Weapon Violence` etc.), reportez-vous au tableau ci-dessous pour mettre à jour votre code.

V6.1 L1	V6.1 L2	V7 L1	V7 L2	V7 L3	V7 ContentTypes
Nudité explicite (Explicit Nudity)	Nudité (Nudity)	Explicite	Nudité explicite (Explicit Nudity)	Mamelon féminin exposé Fesses ou anus exposés	
	Nudité masculine explicite (Graphic Male Nudity)	Explicite	Nudité explicite (Explicit Nudity)	Organes génitaux masculins exposés	
	Nudité féminine explicite (Graphic Female Nudity)	Explicite	Nudité explicite (Explicit Nudity)	Organes génitaux féminins exposés	
	Acte sexuel (Sexual Activity)	Explicite	Activité sexuelle explicite		
	Nudité explicite illustrée	Explicite	Nudité explicite (Explicit Nudity)		Carte vers « Animé » et « Illustré »

	Nudité explicite illustrée	Explicite	Activité sexuelle explicite		Carte vers « Animé » et « Illustré »
	Jouets pour adultes	Explicite	Jouets sexuels		
Suggestif (Suggestive)	Maillot de bain ou sous-vêtement féminin (Female Swimwear Or Underwear)	Maillots de bain ou sous-vêtements	Maillots de bain ou sous-vêtements féminins		
	Maillot de bain ou sous-vêtement masculin (Male Swimwear Or Underwear)	Maillots de bain ou sous-vêtements	Maillots de bain ou sous-vêtements pour hommes		
	Nudité partielle (Partial Nudity)	Non-Explicit Nudité des parties intimes et baisers	Non-Explicit Nudité	Nudité implicite	
	Homme torse nu	Non-Explicit Nudité des parties intimes et baisers	Non-Explicit Nudité	Mamelon masculin exposé	

	Vêtements révélateurs (Revealing Clothes)	Non-Explicit Nudité des parties intimes et baisers	Non-Explicit Nudité	
		Non-Explicit Nudité des parties intimes et baisers	Parties intimes obstruées	
	Situations sexuelles	Non-Explicit Nudité des parties intimes et baisers	Baiser sur les lèvres	
Violence	Violence graphique ou gore	Violence	Violence graphique	Du sang et du sang
	Violence physique	Violence	Violence graphique	Violence physique
	Violence avec des armes	Violence	Violence graphique	Violence avec des armes
	Armes	Violence	Armes	
	Auto-mutilation	Violence	Violence graphique	Self-Harm
Perturbant visuellement	Corps décharnés	Perturbant visuellement	Mort et émaciation	Corps décharnés
	Cadavres	Perturbant visuellement	Mort et émaciation	Cadavres

	Pendaison	Perturbant visuellement	Mort et émaciation	Cadavres
	Accident aérien	Perturbant visuellement	Collisions	Accident aérien
	Explosions et explosions	Violence	Violence graphique	Explosions et explosions
Gestes grossiers	Doigt d’honneur	Gestes grossiers	Doigt d’honneur	
Médicaments	Produits médicamen teux	Drogues et tabac	Produits	
	Usage de drogues	Drogues et tabac	Matériel et usage des drogues et du tabac	
	Pilules	Drogues et tabac	Produits	Pilules
	Accessoir es liés à la drogue	Drogues et tabac	Matériel et usage des drogues et du tabac	
Tabac	Produits du tabac	Drogues et tabac	Produits	
	Smoking	Drogues et tabac	Matériel et usage des drogues et du tabac	Fumer
Alcool	Boire	Alcool	Consommat ion d'alcool	Boire

	Boissons alcoolisées	Alcool	Boissons alcoolisées
Jeu	Jeu	Jeu	
Symboles de haine	Parti nazi	Symboles de haine	Parti nazi
	Suprématie blanche	Symboles de haine	Suprématie blanche
	Extrémiste	Symboles de haine	Extrémiste

Détection d'images inappropriées

Vous pouvez utiliser cette DetectModerationLabelsopération pour déterminer si une image contient un contenu inapproprié ou offensant. Pour obtenir la liste des étiquettes de modération dans Amazon Rekognition consultez Utilisation des API de modération d'images et de vidéos.

Détection d'un contenu inapproprié dans une image

L'image doit être au format .jpg ou .png. Vous pouvez fournir l'image d'entrée sous la forme d'un tableau d'octets d'image (octets d'image encodés en base64), ou spécifier un objet Amazon S3. Dans ces procédures, vous chargez une image (.jpg ou .png) dans votre compartiment S3.

Pour exécuter ces procédures, vous devez avoir installé le AWS CLI ou le AWS SDK approprié. Pour de plus amples informations, veuillez consulter Premiers pas avec Amazon Rekognition. Le compte AWS que vous utilisez doit disposer des autorisations d'accès à l'API Amazon Rekognition. Pour plus d'informations, consultez Actions définies par Amazon Rekognition.

Pour détecter des étiquettes de modération dans une image (SDK)

1. Si vous ne l'avez pas déjà fait :

    a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess` et `AmazonS3ReadOnlyAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Chargez une image dans votre compartiment S3.

   Pour en savoir plus, consultez Chargement d'objets dans Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

3. Utilisez les exemples suivants pour appeler l'opération `DetectModerationLabels`.

   Java

   Cet exemple génère les noms des étiquettes de contenu inapproprié détecté, les niveaux de fiabilité et l'étiquette parent des étiquettes de modération détectées.

   Remplacez les valeurs de `amzn-s3-demo-bucket` et de `photo` par le nom du compartiment S3 et le nom du fichier image utilisés à l'étape 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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
//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

Cette AWS CLI commande affiche la sortie JSON pour l'opération `detect-moderation-labels` CLI.

Remplacez `amzn-s3-demo-bucket` et `input.jpg` par le nom du compartiment S3 et le nom du fichier image utilisés à l'étape 2. Remplacez la valeur de `profile_name` par le nom de votre profil de développeur. Pour utiliser un adaptateur, fournissez l'ARN de la version du projet au paramètre `project-version`.

```
aws rekognition detect-moderation-labels --image "{S3Object:{Bucket:<amzn-s3-
demo-bucket>,Name:<image-name>}}" \
--profile profile-name \
--project-version "ARN"
```

Si vous accédez à la CLI sur un périphérique Windows, utilisez des guillemets doubles au lieu de guillemets simples, et évitez les guillemets doubles internes par une barre oblique inverse (c'est-à-dire \) pour corriger les erreurs d'analyse que vous pourriez rencontrer. Par exemple, consultez ce qui suit :

```
aws rekognition detect-moderation-labels --image "{\"S3Object\":{\"Bucket\":
\"amzn-s3-demo-bucket\",\"Name\":\"image-name\"}}" \
--profile profile-name
```

Python

Cet exemple génère des noms d'étiquettes de contenu inapproprié détecté, des niveaux de confiance et l'étiquette parent pour les étiquettes de contenu inapproprié détecté.

Dans la fonction `main`, remplacez les valeurs de `amzn-s3-demo-bucket` et `photo` par le nom du compartiment S3 et le nom du fichier image que vous avez utilisés à l'étape 2. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

Cet exemple génère les noms des étiquettes de contenu inapproprié détecté, les niveaux de fiabilité et l'étiquette parent des étiquettes de modération détectées.

Remplacez les valeurs de `amzn-s3-demo-bucket` et de `photo` par le nom du compartiment S3 et le nom du fichier image utilisés à l'étape 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 demande d'opération

La valeur d’entrée de `DetectModerationLabels` est une image. Dans cet exemple d’entrée JSON, l’image source est chargée à partir d’un compartiment Amazon S3. `MinConfidence` est le niveau de fiabilité minimum que doit avoir Image Amazon Rekognition en ce qui concerne la précision de l’étiquette détectée pour que celle-ci soit renvoyée dans la réponse.

```
{
```

```
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "input.jpg"
        }
    },
    "MinConfidence": 60
}
```

DetectModerationLabels réponse à l'opération

`DetectModerationLabels` permet de récupérer des images d'entrée à partir d'un compartiment S3. Vous pouvez également les fournir sous forme d'octets d'image. L'exemple suivant est la réponse d'un appel à `DetectModerationLabels`.

Dans l'exemple de réponse JSON suivant, notez les points suivants :

- Informations de détection d'image inappropriées : l'exemple montre une liste d'étiquettes indiquant le contenu inapproprié ou offensant trouvé dans l'image. La liste comprend l'étiquette de premier niveau et chaque étiquette de second niveau détectée dans l'image.

  Etiquette :chaque étiquette possède un nom, une estimation de la fiabilité attribuée par Amazon Rekognition quant à l'exactitude de l'étiquette, et le nom de son étiquette parent. Le nom parent d'une étiquette de premier niveau est `""`.

  Fiabilité de l'étiquette : chaque étiquette a une valeur de fiabilité comprise entre 0 et 100 qui indique le pourcentage de fiabilité attribuée par Amazon Rekognition à l'exactitude de l'étiquette. Vous spécifiez le niveau de fiabilité requis pour qu'une étiquette soit renvoyée dans la réponse de la demande d'opération d'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"
        }
    ]
}
```

Détection des vidéos stockées inappropriées

La détection de contenus inappropriés ou offensants dans les vidéos stockées par Vidéo Amazon Rekognition est une opération asynchrone. Pour commencer à détecter un contenu inapproprié ou offensant, appelez StartContentModeration. Vidéo Amazon Rekognition publie l'état d'achèvement de l'opération d'analyse vidéo dans une rubrique Amazon Simple Notification Service. Si l'analyse vidéo aboutit, appelez GetContentModeration pour obtenir les résultats de cette opération. Pour plus d'informations sur le démarrage de l'analyse vidéo et l'obtention des résultats, consultez Appeler les opérations de Vidéo Amazon Rekognition. Pour obtenir la liste des étiquettes de modération dans Amazon Rekognition consultez Utilisation des API de modération d'images et de vidéos.

Cette procédure s'appuie sur le code figurant dans Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK), qui utilise une file d'attente Amazon Simple Queue Service pour obtenir l'état d'achèvement d'une demande d'analyse vidéo.

Pour détecter le contenu inapproprié ou offensant d'une vidéo stockée dans un compartiment Amazon S3 (SDK)

1. Effectuez une Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK).

2. Ajoutez le code suivant à la classe `VideoDetect` que vous avez créée à l'étape 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);

        }
```

Dans la fonction `main`, remplacez les lignes :

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

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

avec :

```
        StartUnsafeContentDetection(amzn-s3-demo-bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetUnsafeContentDetectionResults();
```

Java V2

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
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
```

Dans la fonction `main`, remplacez les lignes :

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

avec :

```
    analyzer.StartUnsafeContent()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetUnsafeContentResults()
```

> **Note**
>
> Si vous avez déjà exécuté un exemple vidéo autre que Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK), le code à remplacer peut être différent.

3. Exécutez le code. Une liste d'étiquettes de contenu inapproprié détectées dans la vidéo s'affiche à l'écran.

GetContentModeration réponse à l'opération

La réponse de `GetContentModeration` consiste en un tableau `ModerationLabels` composé d'objets ContentModerationDetection. Le tableau contient un élément pour chaque fois qu'une étiquette de contenu inapproprié est détectée. Dans un `ContentModerationDetectionObject` objet, ModerationLabelcontient des informations relatives à un élément détecté présentant un contenu inapproprié ou offensant. `Timestamp`est le temps, en millisecondes à compter du début de la vidéo, auquel l'étiquette a été détectée. Les étiquettes sont hiérarchisées de la même façon que les étiquettes de contenu inapproprié détectées par l'analyse d'image. Pour de plus amples informations, veuillez consulter Modération du contenu.

Voici un exemple de réponse provenant de `GetContentModeration`, triée par `NAME` et agrégée par `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"
    }
}
```

Voici un exemple de réponse provenant de `GetContentModeration`, triée par `NAME` et agrégée par `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"
    }
}
```

Amélioration de la précision grâce à la modération personnalisée

L'DetectModerationLabelsAPI d'Amazon Rekognition vous permet de détecter le contenu inapproprié, indésirable ou offensant. La fonction de modération personnalisée de Rekognition vous permet d'améliorer la précision de en utilisant des adaptateurs. DetectModerationLabels Les adaptateurs sont des composants modulaires qui peuvent être ajoutés à un modèle de Rekognition deep learning existant, afin d'étendre ses capacités aux tâches pour lesquelles il a été entraîné. En créant un adaptateur et en le fournissant à l'DetectModerationLabelsopération, vous pouvez améliorer la précision des tâches de modération du contenu liées à votre cas d'utilisation spécifique.

Lorsque vous personnalisez le modèle de modération de contenu de Rekognition pour des étiquettes de modération spécifiques, vous devez créer un projet et former un adaptateur sur un ensemble d'images que vous fournissez. Vous pouvez ensuite vérifier de manière itérative les performances de l'adaptateur et le réentraîner au niveau de précision souhaité. Les projets sont utilisés pour contenir les différentes versions des adaptateurs.

Vous pouvez créer des projets et des adaptateurs à l'aide de la console Rekognition. Vous pouvez également utiliser un AWS SDK et les API associées pour créer un projet, former un adaptateur et gérer vos adaptateurs.

Création et utilisation d'adaptateurs

Les adaptateurs sont des composants modulaires qui peuvent être ajoutés au modèle deep learning de Rekognition existant afin d'étendre ses capacités aux tâches pour lesquelles il est entraîné. En formant un modèle de deep learning avec des adaptateurs, vous pouvez obtenir une meilleure précision pour les tâches d'analyse d'images liées à votre cas d'utilisation spécifique.

Pour créer et utiliser un adaptateur, vous devez fournir des données d'entraînement et de test à Rekognition. Vous pouvez y parvenir de deux façons :

- Analyse et vérification en bloc : vous pouvez créer un jeu de données d'entraînement en analysant en bloc des images que Rekognition analyse, et auxquelles Rekognition attribue des étiquettes. Vous pouvez ensuite consulter les annotations générées pour vos images et vérifier ou corriger les prédictions. Pour plus d'informations sur le fonctionnement de l'analyse en bloc d'images, voir Analyse en bloc.
- Annotation manuelle : avec cette approche, vous créez vos données d'entraînement en téléchargeant et en annotant des images. Vous créez vos données de test soit en téléchargeant et en annotant des images, soit en les divisant automatiquement.

Choisissez l'une des rubriques suivantes pour en savoir plus :

Rubriques

- Analyse et vérification en bloc
- Annotation manuelle

Analyse et vérification en bloc

Avec cette approche, vous téléchargez un grand nombre d'images que vous souhaitez utiliser comme données d'entraînement, puis vous utilisez Rekognition pour obtenir des prédictions pour ces images, qui leur attribue automatiquement des étiquettes. Vous pouvez utiliser ces prédictions comme point de départ pour votre adaptateur. Vous pouvez vérifier l'exactitude des prévisions, puis entraîner l'adaptateur en fonction des prévisions vérifiées. Cela peut être fait avec la AWS console.

Analyse en bloc et modération personnalisée

Téléchargez des images pour une analyse en bloc

Pour créer un jeu de données d'entraînement pour votre adaptateur, téléchargez des images en bloc pour que Rekognition puisse prévoir les étiquettes. Pour de meilleurs résultats, fournissez autant d'images que possible pour l'entraînement, dans la limite de 10 000, et assurez-vous que les images sont représentatives de tous les aspects de votre cas d'utilisation.

Lorsque vous utilisez la AWS console, vous pouvez télécharger des images directement depuis votre ordinateur ou fournir un bucket Amazon Simple Storage Service qui stocke vos images. Toutefois, lorsque vous utilisez les API de Rekognition avec un SDK, vous devez fournir un fichier manifeste qui fait référence aux images stockées dans un compartiment Amazon Simple Storage Service. Pour plus d'informations, consultez Analyse en bloc.

Vérifiez les prédictions

Une fois que vous avez chargé vos images sur la console Rekognition, Rekognition génère des étiquettes pour celles-ci. Vous pouvez ensuite vérifier les prédictions dans l'une des catégories suivantes : vrai positif, faux positif, vrai négatif, faux négatif. Après avoir vérifié les prévisions, vous pouvez former un adaptateur en fonction de vos commentaires.

Entraînez l'adaptateur

Une fois que vous avez terminé de vérifier les prédictions renvoyées par l'analyse en bloc, vous pouvez lancer le processus de formation de votre adaptateur.

Obtenez le AdapterId

Une fois que l'adaptateur a été formé, vous pouvez obtenir l'identifiant unique que votre adaptateur pourra utiliser avec les API d'analyse d'image de Rekognition.

Appelez l'opération de l'API

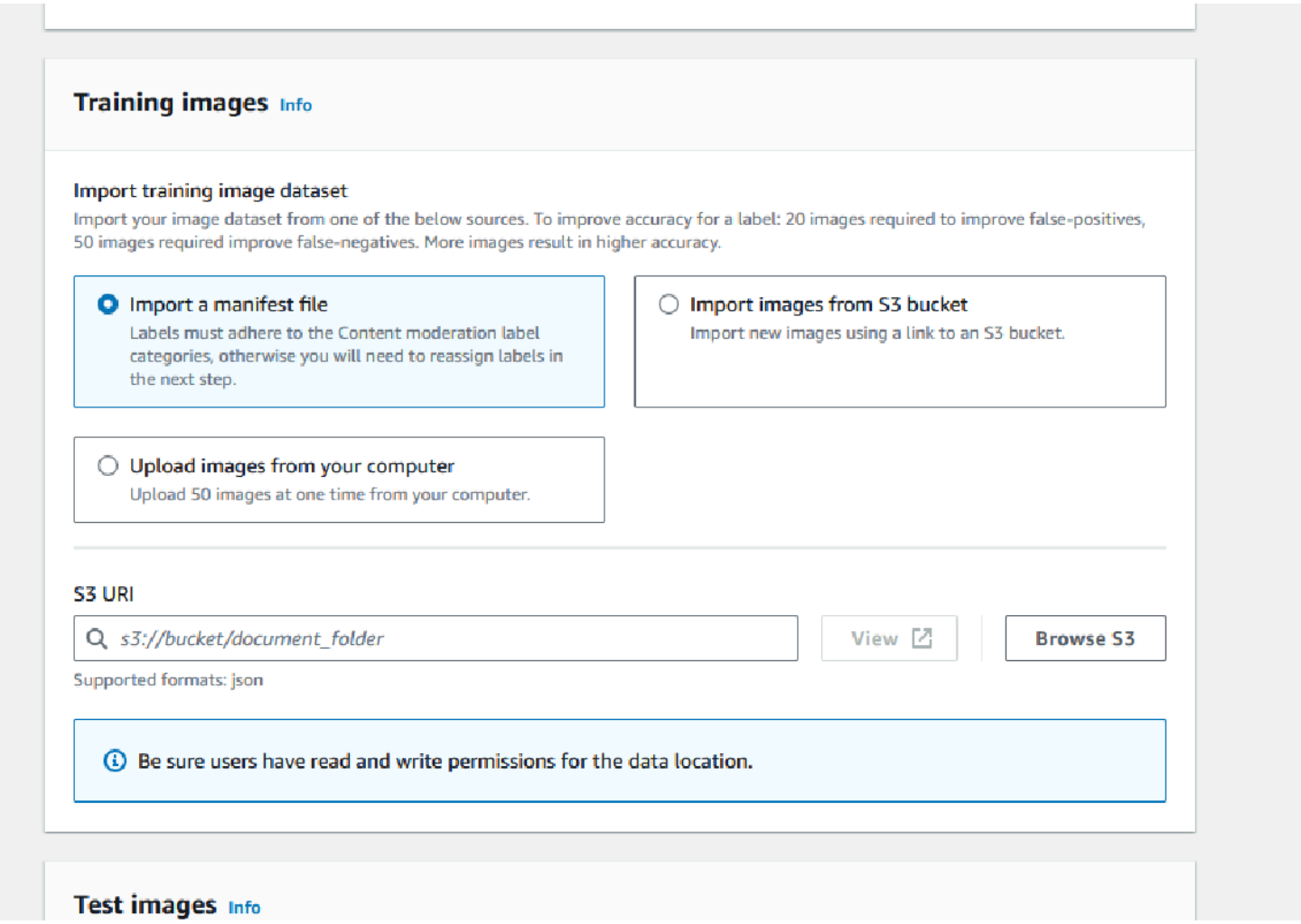
Pour appliquer votre adaptateur personnalisé, fournissez son identifiant lorsque vous appelez l'une des API d'analyse d'image compatibles avec les adaptateurs. Cela améliore la précision des prédictions pour vos images.

Annotation manuelle

Avec cette approche, vous créez vos données d'entraînement en téléchargeant et en annotant des images manuellement. Vous créez vos données de test soit en téléchargeant et en annotant les images de test, soit en les divisant automatiquement pour que Rekognition utilise automatiquement une partie de vos données d'entraînement comme images de test.

Téléchargement et annotation d'images

Pour entraîner l'adaptateur, vous devez télécharger un ensemble d'exemples d'images représentatifs de votre cas d'utilisation. Pour de meilleurs résultats, fournissez autant d'images que possible pour l'entraînement, dans la limite de 10 000, et assurez-vous que les images sont représentatives de tous les aspects de votre cas d'utilisation.



Lorsque vous utilisez la AWS console, vous pouvez télécharger des images directement depuis votre ordinateur, fournir un fichier manifeste ou fournir un compartiment Amazon S3 qui stocke vos images.

Toutefois, lorsque vous utilisez les API de Rekognition avec un SDK, vous devez fournir un fichier manifeste qui fait référence aux images stockées dans un compartiment Amazon S3.

Vous pouvez utiliser l'interface d'annotation de la console Rekognition pour annoter vos images. Annotez vos images en les repérant avec des étiquettes, cela constitue une « vérité de base » pour la formation. Vous devez également désigner des ensembles d'entraînement et de test, ou utiliser la fonctionnalité de division automatique, avant de pouvoir entraîner un adaptateur. Lorsque vous avez fini de désigner vos jeux de données et d'annoter vos images, vous pouvez créer un adaptateur basé sur les images annotées de votre ensemble de tests. Vous pouvez ensuite évaluer les performances de votre adaptateur.

Créez un ensemble de tests

Vous devrez fournir un ensemble de tests annoté ou utiliser la fonction de division automatique. Le kit d'entraînement est utilisé pour entraîner réellement l'adaptateur. L'adaptateur apprend les motifs contenus dans ces images annotées. Le kit de test est utilisé pour évaluer les performances du modèle avant de finaliser l'adaptateur.

Entraînez l'adaptateur

Une fois que vous avez terminé d'annoter les données d'entraînement, ou que vous avez fourni un fichier manifeste, vous pouvez lancer le processus de formation de votre adaptateur.

Obtenez l'ID de l'adaptateur

Une fois que l'adaptateur a été formé, vous pouvez obtenir l'identifiant unique que votre adaptateur utilisera avec les API d'analyse d'image de Rekognition.

Appelez l'opération de l'API

Pour appliquer votre adaptateur personnalisé, fournissez son identifiant lorsque vous appelez l'une des API d'analyse d'image compatibles avec les adaptateurs. Cela améliore la précision des prédictions pour vos images.

Préparation de vos jeux de données

Pour créer un adaptateur, vous devez fournir à Rekognition deux jeux de données, un jeu de données d'entraînement et un jeu de données de test. Chaque ensemble de données est composé de deux éléments : des images et annotations/labels. Les sections suivantes expliquent à quoi servent les étiquettes et les images, et comment elles sont combinées pour créer des jeux de données.

Images

Vous devrez former un adaptateur sur des échantillons représentatifs de vos images. Lorsque vous sélectionnez des images pour l'entraînement, essayez d'inclure au moins quelques images illustrant la réponse attendue pour chacune des étiquettes que vous ciblez avec votre adaptateur.

Pour créer un jeu de données d'entraînement, vous devez fournir l'un des deux types d'image suivants :

- Images avec des prédictions de faux positifs. Par exemple, lorsqu'un modèle de base prédit qu'une image contient de l'alcool, mais ce n'est pas le cas.

- Images avec des prédictions de faux négatifs. Par exemple, lorsqu'un modèle de base prédit qu'une image ne contient pas d'alcool, mais c'est le cas.

Pour créer un jeu de données équilibré, il est recommandé de fournir l'un des deux types d'image suivants :

- Images avec des prédictions de vrais positifs. Par exemple, lorsqu'un modèle de base prédit correctement qu'une image contient de l'alcool. Il est recommandé de fournir ces images si vous fournissez des images avec des faux positifs.
- Images avec des prédictions de vrais négatifs. Par exemple, lorsqu'un modèle de base prédit correctement qu'une image ne contient pas d'alcool. Il est recommandé de fournir ces images si vous fournissez des images avec des faux négatifs.

Étiquettes

Une étiquette fait référence à l'un des éléments suivants : objets, événements, concepts ou activités. Pour la modération du contenu, une étiquette est une instance de contenu inapproprié, indésirable ou offensant.

Dans le contexte de la création d'un adaptateur en entraînant le modèle de base de Rekognition, lorsqu'une étiquette est attribuée à une image, on parle d'annotation. Lorsque vous entraînez un adaptateur avec la console de Rekognition, vous utilisez la console pour ajouter des annotations à vos images en choisissant une étiquette, puis en étiquetant les images correspondant à cette étiquette. Grâce à ce processus, le modèle apprend à identifier les éléments de vos images en fonction de l'étiquette attribuée. Ce processus de liaison permet au modèle de se concentrer sur le contenu le plus pertinent lors de la création d'un adaptateur, ce qui améliore la précision de l'analyse des images.

Vous pouvez également fournir un fichier manifeste contenant des informations sur les images et les annotations qui les accompagnent.

Jeux de données d'entraînement et de test

Le jeu de données d'entraînement est la base pour affiner le modèle et créer un adaptateur personnalisé. Vous devez fournir un jeu de données d'entraînement annoté pour que le modèle puisse en tirer des leçons. Le modèle tire les leçons de ce jeu de données pour améliorer ses performances sur le type d'images que vous fournissez.

Pour améliorer la précision, vous devez créer votre jeu de données d'entraînement à l'aide d' annotation/labeling images. Vous pouvez y parvenir de deux façons :

- Attribution manuelle d'étiquettes : vous pouvez utiliser la console de Rekognition pour créer un jeu de données d'entraînement en téléchargeant les images que vous souhaitez que votre ensemble de données contienne, puis en attribuant manuellement des étiquettes à ces images.
- Fichier manifeste : vous pouvez utiliser un fichier manifeste pour entraîner votre adaptateur. Le fichier manifeste contient des informations sur les annotations fondamentales pour vos images d'entraînement et de test, ainsi que sur l'emplacement de vos images d'entraînement. Vous pouvez fournir le fichier manifeste lorsque vous entraînez un adaptateur à l'aide des API de Rekognition ou lorsque vous utilisez la console. AWS

Le jeu de données de test est utilisé pour évaluer les performances de l'adaptateur après l'entraînement. Pour garantir une évaluation fiable, le jeu de données de test est créé en utilisant une tranche du jeu de données d'apprentissage d'origine que le modèle n'a jamais vue auparavant. Ce processus garantit que les performances de l'adaptateur sont évaluées à l'aide de nouvelles données, créant ainsi des mesures et des métriques précises. Pour des améliorations de précision optimales, voir Bonnes pratiques relatives aux adaptateurs d'entraînement.

Gestion des adaptateurs à l'aide du AWS CLI et SDK

Rekognition vous permet d'utiliser de multiples fonctionnalités qui s'appuient sur des modèles de vision par ordinateur préentraînés. Avec ces modèles, vous pouvez effectuer des tâches telles que la détection des étiquettes et la modération du contenu. Vous pouvez également personnaliser certains modèles à l'aide d'un adaptateur.

Vous pouvez utiliser les API de création et de gestion de projets de Rekognition (comme et CreateProjectVersion) pour créer CreateProjectet former des adaptateurs. Les pages suivantes décrivent comment utiliser les opérations d'API pour créer, entraîner et gérer vos adaptateurs à l'aide de la AWS console, du AWS SDK de votre choix ou de la AWS CLI.

Après avoir entraîné un adaptateur, vous pouvez l'utiliser pour exécuter l'inférence avec les fonctionnalités prises en charge. Actuellement, les adaptateurs sont pris en charge lors de l'utilisation de la fonctionnalité de modération du contenu.

Lorsque vous entraînez un adaptateur à l'aide d'un AWS SDK, vous devez fournir vos étiquettes de base (annotations d'image) sous la forme d'un fichier manifeste. Vous pouvez également utiliser la console de Rekognition pour créer et entraîner un adaptateur.

> **Note**
>
> Les adaptateurs ne peuvent pas être copiés. Seules les versions du projet Rekognition Custom Labels peuvent être copiées.

Rubriques

- Statuts des adaptateurs
- Création d'un projet
- Description des projets
- Suppression d'un projet
- Création d'une version de projet
- Description d'une version de projet
- Suppression d'une version de projet

Statuts des adaptateurs

L'adaptateur de modération personnalisé (versions du projet) peut avoir l'un des statuts suivants :

- TRAINING_IN_PROGRESS - L'adaptateur est en train de s'entraîner sur les fichiers que vous avez fournis en tant que documents de formation.
- TRAINING_COMPLETED - L'adaptateur a terminé l'entraînement avec succès et est prêt à être examiné par vous pour vérifier ses performances.
- TRAINING_FAILED - L'adaptateur n'a pas réussi à terminer sa formation pour une raison quelconque. Consultez le fichier manifeste de sortie et le résumé du manifeste de sortie pour obtenir des informations sur la cause de l'échec.
- SUPPRESSION - L'adaptateur est en cours de suppression.
- OBSOLÈTE - L'adaptateur a été formé sur une ancienne version du modèle de base de modération du contenu. Il est en période de grâce et expirera dans les 60 à 90 jours suivant la sortie de la nouvelle version du modèle de base. Pendant la période de grâce, vous pouvez toujours utiliser l'adaptateur pour les inférences DetectModerationLabelsou les opérations StartMediaAnalysisJobd'API. Reportez-vous à la console de modération personnalisée pour connaître la date d'expiration de vos adaptateurs.
- EXPIRÉ - L'adaptateur a été formé sur une ancienne version du modèle de base de modération du contenu et il ne peut plus être utilisé pour obtenir des résultats personnalisés avec les opérations

de l' StartMediaAnalysisJob API DetectModerationLabels or. Si un adaptateur expiré est spécifié dans une demande d'inférence, il sera ignoré et la réponse sera renvoyée à partir de la version la plus récente du modèle de base de modération personnalisée.

Création d’un projet

Avec cette CreateProjectopération, vous pouvez créer un projet qui contiendra un adaptateur pour les opérations de détection d'étiquettes de Rekognition. Un projet est un groupe de ressources. Dans le cas d'opérations de détection d'étiquettes telles que DetectModerationLabels, un projet vous permet de stocker des adaptateurs que vous pouvez utiliser pour personnaliser le modèle de Rekognition de base. Lors de l'appel CreateProject, vous fournissez le nom du projet que vous souhaitez créer à l' ProjectName argument.

Pour créer un projet à l'aide de la AWS console :

- Connectez-vous à la console de Rekognition
- Cliquez sur Modération personnalisée
- Choisissez Créer un projet
- Sélectionnez Créer un nouveau projet ou Ajouter à un projet existant
- Ajoutez un nom de projet
- Ajoutez un nom d’adaptateur
- Ajoutez une description si vous le souhaitez
- Choisissez la manière dont vous souhaitez importer vos images d’entraînement : fichier manifeste, depuis le compartiment S3 ou depuis votre ordinateur
- Choisissez si vous souhaitez diviser automatiquement vos données d’entraînement ou importer un fichier manifeste
- Indiquez si vous souhaitez ou non que le projet soit automatiquement mis à jour
- Cliquez sur Créer un projet

Pour créer un projet à l'aide de la AWS CLI et du SDK :

1. Si ce n'est pas déjà fait, installez et configurez la AWS CLI et les AWS SDK. Pour plus d’informations, consultez Étape 2 : configurer le AWS CLI and AWS Kits SDK.
2. Utilisez le code suivant pour créer un projet :

CLI

```
# Request
# Creating Content Moderation Project
aws rekognition create-project \
    --project-name "project-name" \
    --feature CONTENT_MODERATION \
    --auto-update ENABLED
    --profile profile-name
```

Description des projets

Vous pouvez utiliser l'DescribeProjectsAPI pour obtenir des informations sur vos projets, notamment des informations sur tous les adaptateurs associés à un projet.

Pour décrire les projets avec la AWS CLI et le SDK :

1. Si ce n'est pas déjà fait, installez et configurez la AWS CLI et les AWS SDK. Pour plus d'informations, consultez Étape 2 : configurer le AWS CLI and AWS Kits SDK.
2. Utilisez le code suivant pour créer un projet :

CLI

```
# Request
# Getting CONTENT_MODERATION project details
aws rekognition describe-projects \
    --features CONTENT_MODERATION
    --profile profile-name
```

Suppression d'un projet

Vous pouvez supprimer un projet à l'aide de la console Rekognition ou en appelant l'API. DeleteProject Pour supprimer un projet, vous devez d'abord supprimer chacun des adaptateurs associés. Il est impossible de rétablir un projet ou un modèle supprimé.

Pour supprimer un projet à l'aide de la AWS console :

- Connectez-vous à la console de Rekognition.
- Cliquez sur Modération personnalisée.
- Vous devez supprimer chaque adaptateur associé à votre projet avant de pouvoir supprimer le projet lui-même. Supprimez tous les adaptateurs associés au projet en les sélectionnant, puis en sélectionnant Supprimer.
- Sélectionnez le projet, puis cliquez sur le bouton Supprimer.

Pour supprimer un projet à l'aide de la AWS CLI et du SDK :

1. Si ce n'est pas déjà fait, installez et configurez la AWS CLI et les AWS SDK. Pour plus d'informations, consultez Étape 2 : configurer le AWS CLI and AWS Kits SDK.
2. Utilisez le code suivant pour créer un projet :

CLI

```
aws rekognition delete-project
  --project-arn project_arn \
  --profile profile-name
```

Création d'une version de projet

Vous pouvez entraîner un adaptateur en vue de son déploiement à l'aide de CreateProjectVersioncette opération. CreateProjectVersion crée d'abord une nouvelle version d'un adaptateur associé à un projet, puis commence à entraîner l'adaptateur. La réponse de CreateProjectVersion est un Amazon Resource Name (ARN) pour la version du modèle. L'entraînement prend un certain temps. Vous pouvez obtenir le statut actuel en appelant DescribeProjectVersions. Lors de l'entraînement d'un modèle, Rekognition utilise les jeux de données d'entraînement et de test associés au projet. Vous créez des jeux de données à l'aide de la console. Pour de plus amples informations, consultez la section sur les jeux de données.

Pour créer une version de projet à l'aide de la console Rekognition :

- Connectez-vous à la console Rekognition AWS

- Cliquez sur Modération personnalisée
- Sélectionner un projet.
- Sur la page « Détails du projet », choisissez Créer un adaptateur
- Sur la page « Créer un projet », renseignez les informations requises pour les détails du projet, les images de formation et les images de test, puis sélectionnez Créer un projet.
- Sur la page « Attribuer des étiquettes aux images », ajoutez des étiquettes à vos images et, lorsque vous avez terminé, sélectionnez Commencer l'entraînement

Pour créer une version de projet à l'aide de la AWS CLI et du SDK :

1. Si ce n'est pas déjà fait, installez et configurez la AWS CLI et les AWS SDK. Pour plus d'informations, consultez Étape 2 : configurer le AWS CLI and AWS Kits SDK.
2. Utilisez le code suivant pour créer une version de projet :

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
```

Description d'une version de projet

Vous pouvez répertorier et décrire les adaptateurs associés à un projet à l'aide de cette DescribeProjectVersionsopération. Vous pouvez spécifier jusqu'à 10 versions de modèles dans ProjectVersionArns. Si vous ne spécifiez aucune valeur, les descriptions de toutes les versions de modèle du projet sont renvoyées.

Pour décrire une version de projet à l'aide de la AWS CLI et du SDK :

1. Si ce n'est pas déjà fait, installez et configurez la AWS CLI et les AWS SDK. Pour plus d'informations, consultez Étape 2 : configurer le AWS CLI and AWS Kits SDK.
2. Utilisez le code suivant pour décrire la version d'un projet :

CLI

```
aws rekognition describe-project-versions
  --project-arn project_arn \
  --version-names [versions]
```

Suppression d'une version de projet

Vous pouvez supprimer un adaptateur de Rekognition associé à un projet à l'aide de cette opération. DeleteProjectVersion Vous ne pouvez pas supprimer un adaptateur s'il fonctionne ou s'il est en cours d'entraînement. Pour vérifier l'état d'un adaptateur, appelez l' DescribeProjectVersions opération et vérifiez le champ Status renvoyé par celui-ci. Pour arrêter un appel d'adaptateur en cours d'exécution StopProjectVersion. Si le modèle est en cours d'entraînement, attendez la fin de l'entraînement pour le supprimer. Vous devez supprimer chaque adaptateur associé à votre projet avant de pouvoir supprimer le projet lui-même.

Pour supprimer une version de projet à l'aide de la console Rekognition :

- Connectez-vous à la console de Rekognition
- Cliquez sur Modération personnalisée
- Dans l'onglet Projets, vous pouvez voir tous vos projets et les adaptateurs associés. Sélectionnez un adaptateur, puis sélectionnez Supprimer.

Pour supprimer une version de projet à l'aide de la AWS CLI et du SDK :

1. Si ce n'est pas déjà fait, installez et configurez la AWS CLI et les AWS SDK. Pour plus d'informations, consultez Étape 2 : configurer le AWS CLI and AWS Kits SDK.
2. Utilisez le code suivant pour supprimer une version de projet :

CLI

```
# Request
aws rekognition delete-project-version
  --project-version-arn model_arn \
  --profile profile-name
```

Tutoriel sur l'adaptateur de modération personnalisé

Ce didacticiel explique comment créer, former, évaluer, utiliser et gérer des adaptateurs à l'aide de la console de Rekognition. Pour créer, utiliser et gérer des adaptateurs avec le AWS SDK, consultezGestion des adaptateurs à l'aide du AWS CLI et SDK.

Les adaptateurs vous permettent d'améliorer la précision des opérations d'API de Rekognition en personnalisant le comportement du modèle en fonction de vos besoins et de vos cas d'utilisation. Après avoir créé un adaptateur à l'aide de ce didacticiel, vous pourrez l'utiliser pour analyser vos propres images à l'aide d'opérations telles DetectModerationLabelsque le réentraînement de l'adaptateur en vue de futures améliorations.

Dans ce didacticiel, vous allez découvrir comment :

- Créer un projet à l'aide de la console de Rekognition
- Annoter vos données d'entraînement
- Former votre adaptateur sur votre jeu de données d'entraînement
- Vérifier les performances de votre adaptateur
- Utiliser votre adaptateur pour l'analyse d'images

Conditions préalables

Avant de terminer ce didacticiel, il est recommandé de le lire attentivement Création et utilisation d'adaptateurs.

Pour créer un adaptateur, vous pouvez utiliser l'outil Rekognition Console pour créer un projet, télécharger et annoter vos propres images, puis entraîner un adaptateur sur ces images. Pour commencer, consultez Création d'un projet et formation d'un adaptateur.

Vous pouvez également utiliser la console ou l'API de Rekognition pour récupérer des prédictions pour les images, puis vérifier les prédictions avant d'entraîner un adaptateur sur ces prédictions. Pour commencer, consultez Analyse en bloc, vérification des prédictions et formation d'un adaptateur.

Annotation d'image

Vous pouvez annoter vous-même les images en les étiquetant à l'aide de la console Rekognition, ou utiliser l'analyse Rekognition Bulk pour annoter des images dont vous pouvez ensuite vérifier qu'elles ont été correctement étiquetées. Choisissez l'un des sujets ci-dessous pour commencer.

Rubriques

- Création d'un projet et formation d'un adaptateur
- Analyse en bloc, vérification des prédictions et formation d'un adaptateur

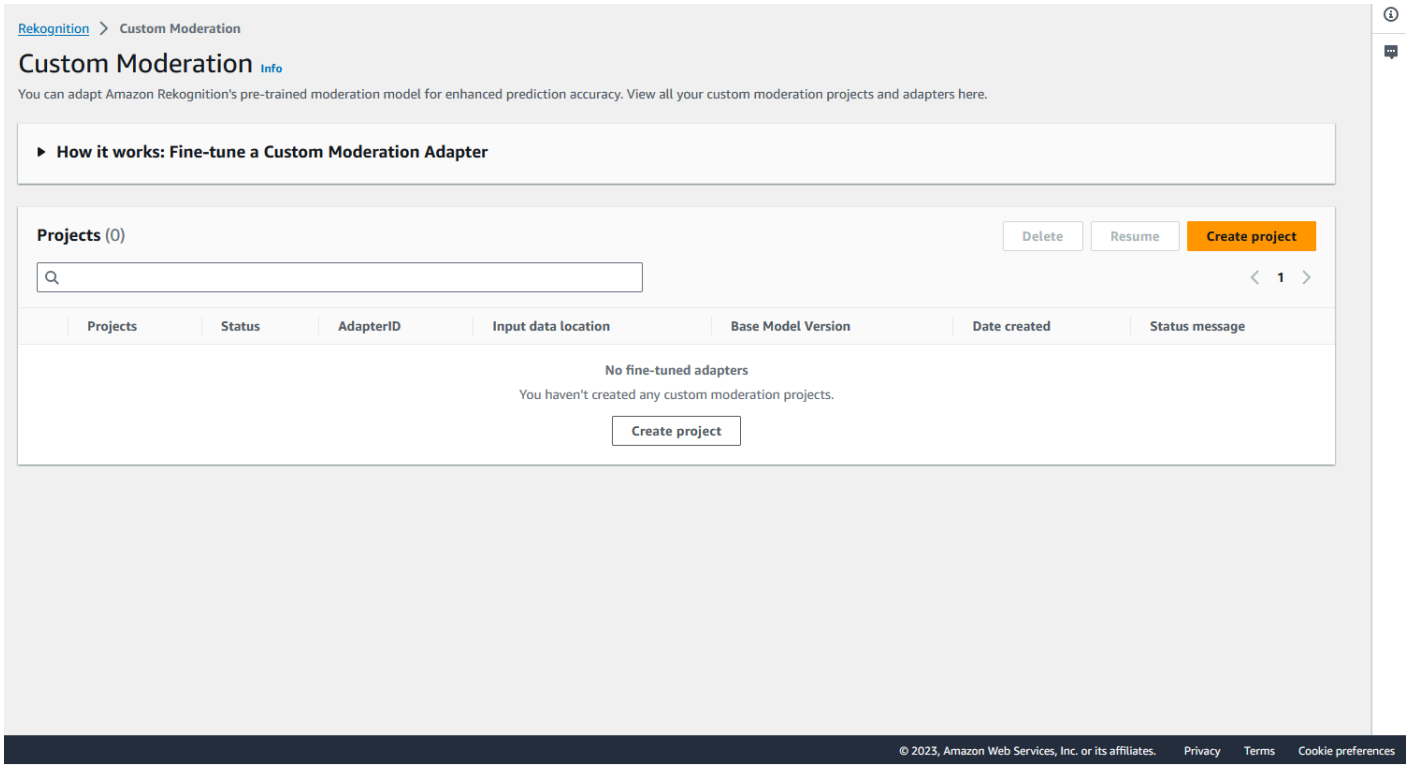
Création d'un projet et formation d'un adaptateur

Effectuez les étapes suivantes pour entraîner votre adaptateur en annotant des images à l'aide de la console Rekognition.

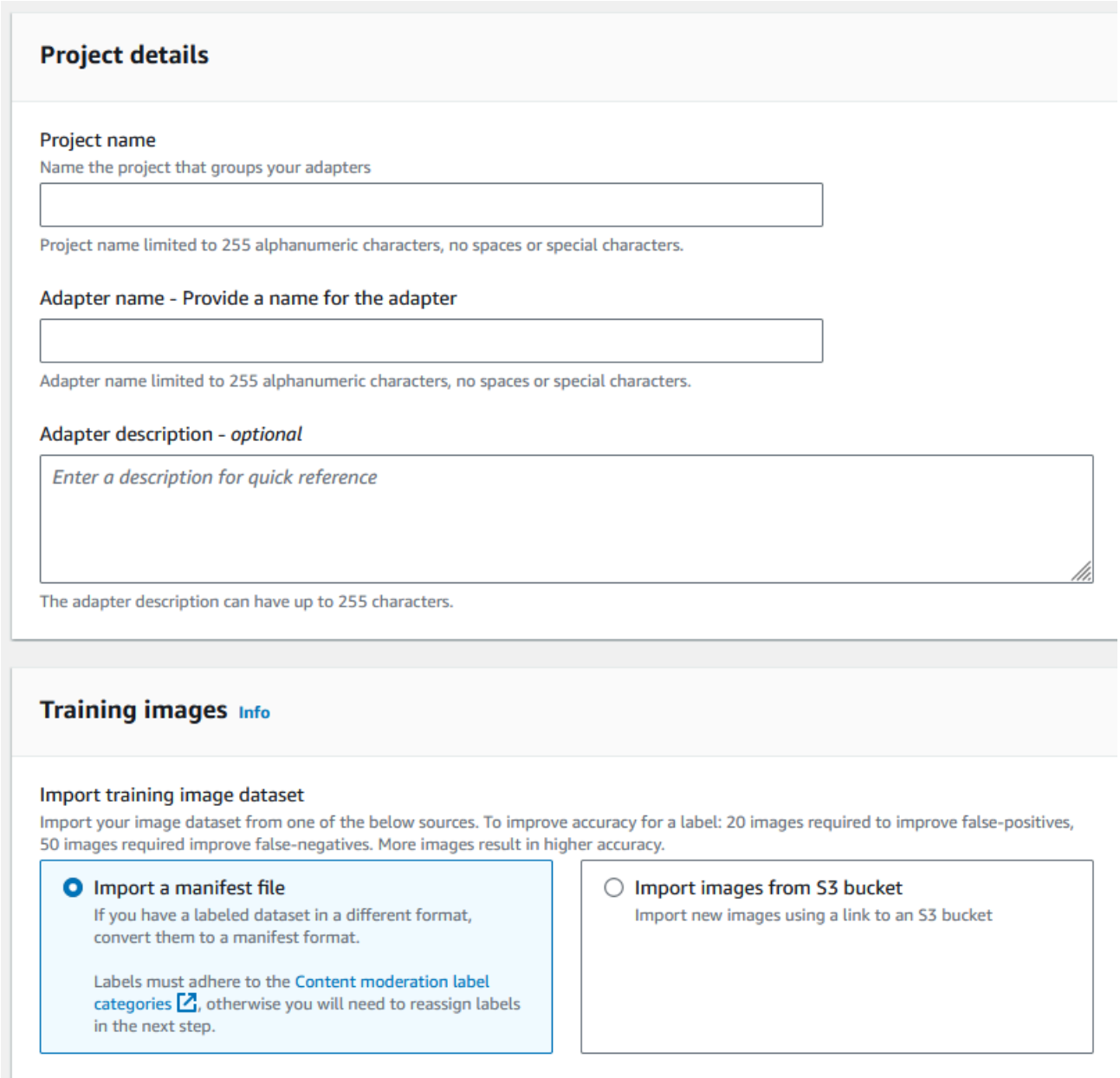
Créer un projet

Avant de pouvoir entraîner ou utiliser un adaptateur, vous devez créer le projet qui le contiendra. Vous devez également fournir les images utilisées pour entraîner votre adaptateur. Pour créer un projet, un adaptateur et vos jeux de données d'images :

1. Connectez-vous à la console de AWS gestion et ouvrez la console de Rekognition à l'adresse https://console.aws.amazon.com/rekognition/.
2. Dans le volet gauche, sélectionnez Modération personnalisée. La page d'accueil de Rekognition Custom Moderation s'affiche.



3. La page d'accueil de Modération personnalisée affiche une liste de tous vos projets et adaptateurs, ainsi qu'un bouton permettant de créer un adaptateur. Choisissez Créer un projet pour créer un nouveau projet et un nouvel adaptateur.
4. Si c'est la première fois que vous créez un adaptateur, vous serez invité à créer un compartiment Amazon S3 pour stocker les fichiers liés à votre projet et à votre adaptateur. Choisissez Créez un compartiment Amazon S3.
5. Sur la page suivante, saisissez le nom de l'adaptateur et le nom du projet. Fournissez une description de l'adaptateur si vous le souhaitez.



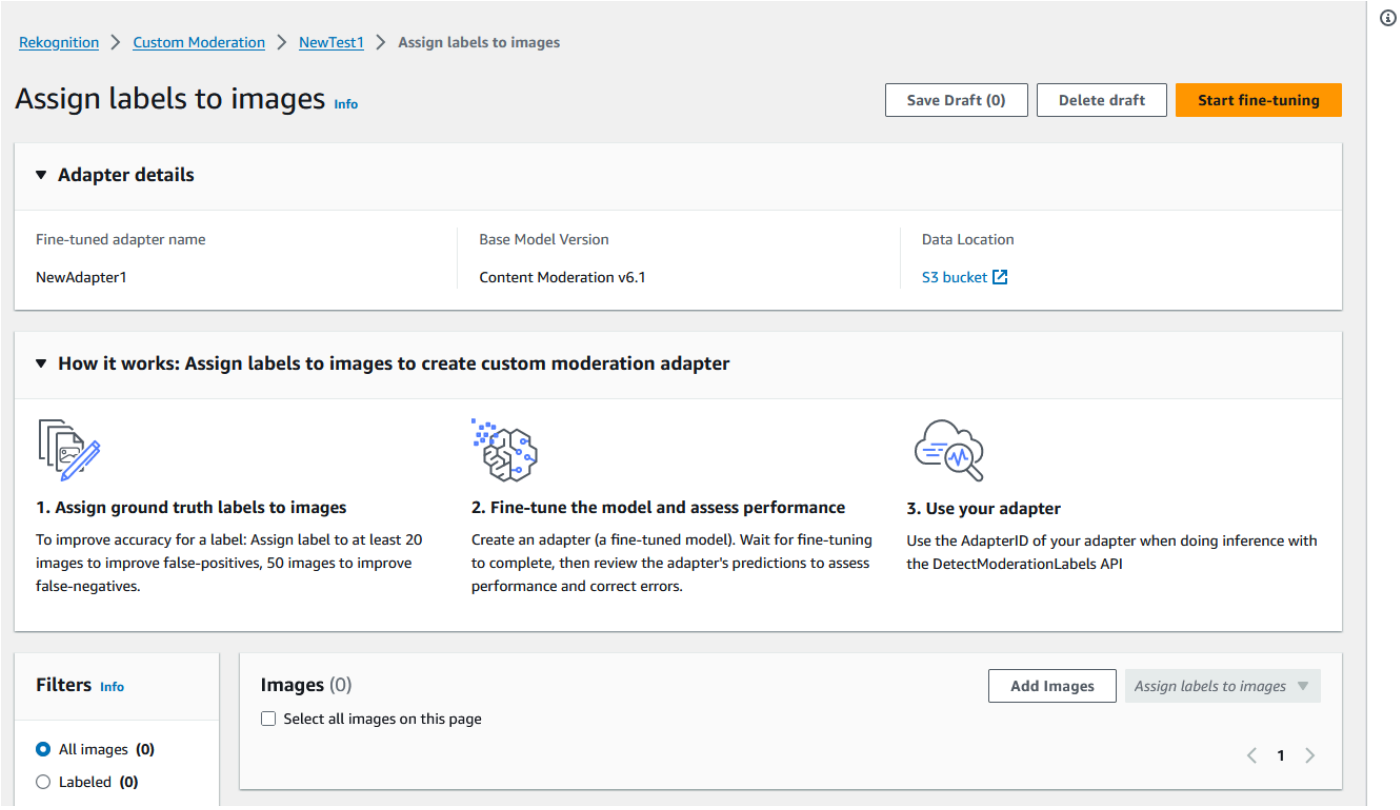
6. Dans cette étape, vous allez également fournir les images pour votre adaptateur. Vous pouvez sélectionner : Importer des images depuis votre ordinateur, importer un fichier manifeste ou importer des images depuis le compartiment Amazon S3. Si vous choisissez d'importer vos images depuis un compartiment Amazon S3, indiquez le chemin d'accès au compartiment et au dossier contenant vos images d'entraînement. Si vous téléchargez vos images directement depuis votre ordinateur, notez que vous ne pouvez télécharger que 30 images à la fois. Si vous utilisez un fichier manifeste contenant des annotations, vous pouvez ignorer les étapes répertoriées ci-dessous concernant les annotations d'image et passer à la section sur Vérification des performances de l'adaptateur.

7. Dans la section Détails du jeu de données de test, choisissez Autosplit pour que Rekognition sélectionne automatiquement le pourcentage approprié de vos images comme données de test, ou vous pouvez choisir Importer manuellement le fichier manifeste.
8. Après avoir renseigné ces informations, sélectionnez Créer un projet.

Entraînez l'adaptateur

Pour entraîner un adaptateur sur vos propres images non annotées :

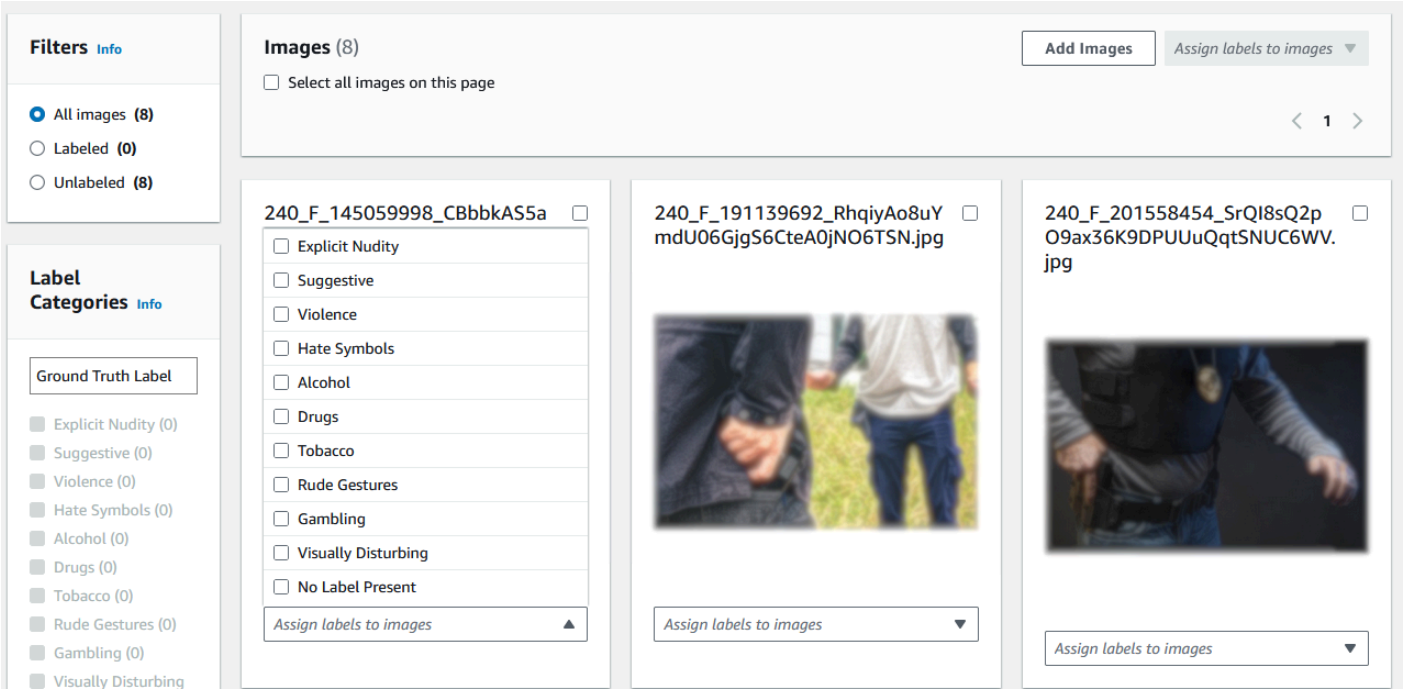
1. Sélectionnez le projet qui contient votre adaptateur, puis choisissez l'option Attribuer une étiquette aux images.
2. Sur la page Attribuer une étiquette aux images, vous pouvez voir toutes les images qui ont été téléchargées sous forme d'images d'entraînement. Vous pouvez filtrer ces images à la fois par labeled/unlabeled statut et par catégorie d'étiquette à l'aide des deux panneaux de sélection d'attributs sur la gauche. Vous pouvez ajouter des images supplémentaires à votre jeu de données d'entraînement en sélectionnant le bouton Ajouter des images.



3. Après avoir ajouté des images au jeu de données d'entraînement, vous devez annoter vos images à l'aide d'étiquettes. Après avoir chargé vos images, la page « Attribuer des étiquettes aux images » sera mise à jour pour afficher les images que vous avez téléchargées. Vous êtes invité à

sélectionner l'étiquette appropriée pour vos images dans une liste déroulante d'étiquettes prises en charge par Rekognition Moderation. Vous pouvez sélectionner plusieurs étiquettes.

4. Continuez ce processus jusqu'à ce que vous ayez ajouté des étiquettes à chacune des images de vos données d'entraînement.
5. Après avoir étiqueté toutes vos données, sélectionnez Démarrer l'entraînement pour commencer à entraîner le modèle, qui crée votre adaptateur.



6. Avant de commencer le processus de formation, vous pouvez ajouter les balises de votre choix à l'adaptateur. Vous pouvez également fournir à l'adaptateur une clé de chiffrement personnalisée ou utiliser une clé AWS KMS. Une fois que vous avez fini d'ajouter les balises de votre choix et de personnaliser le cryptage à votre guise, sélectionnez Entraîner l'adaptateur pour démarrer le processus de formation de votre adaptateur.
7. Attendez que votre adaptateur ait terminé l'entraînement. Une fois la formation terminée, vous recevrez une notification indiquant que la création de votre adaptateur est terminée.

Une fois que le statut de votre adaptateur est « Entraînement terminé », vous pouvez consulter les métriques de votre adaptateur

Analyse en bloc, vérification des prédictions et formation d'un adaptateur

Procédez comme suit pour entraîner votre adaptateur en vérifiant les prévisions d'analyse en bloc issues du modèle de modération du contenu de Rekognition.

Pour entraîner un adaptateur en vérifiant les prédictions issues du modèle de modération du contenu de Rekognition, vous devez :

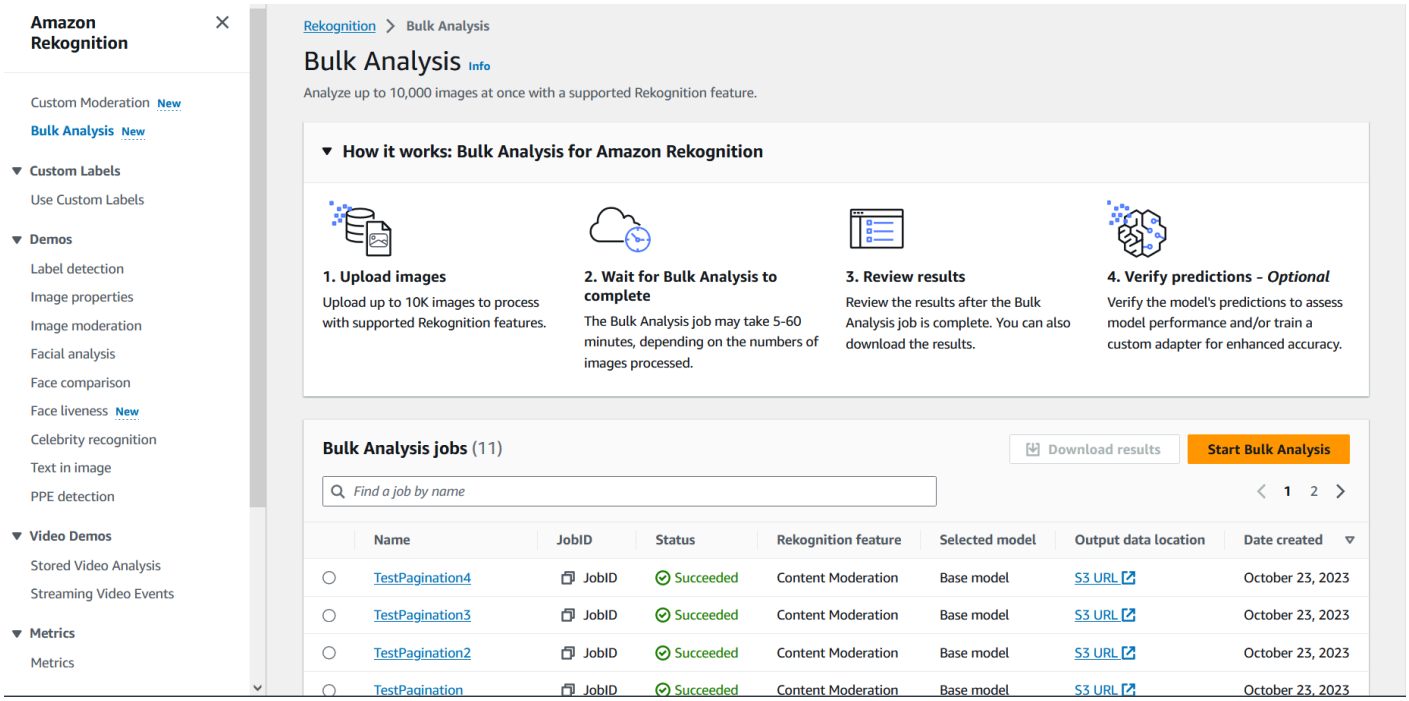
1. Effectuer une analyse en bloc de vos images
2. Vérifier les prédictions renvoyées pour vos images

Vous pouvez obtenir des prédictions pour les images en effectuant une analyse en bloc avec le modèle de base de Rekognition ou un adaptateur que vous avez déjà créé.

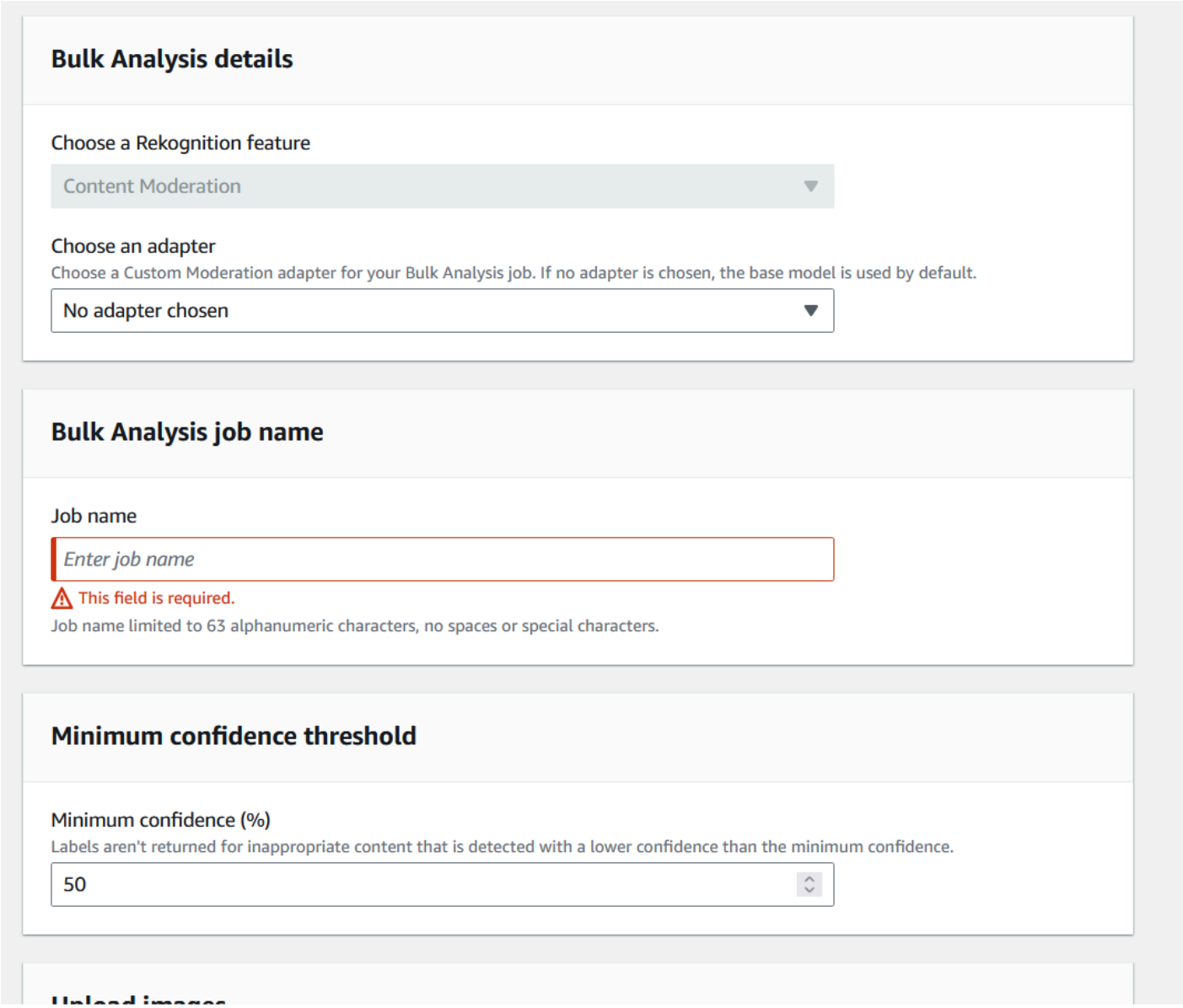
Effectuer une analyse en bloc de vos images

Pour entraîner un adaptateur sur les prédictions que vous avez vérifiées, vous devez d'abord démarrer une tâche d'analyse en bloc pour analyser un lot d'images à l'aide du modèle de base de Rekognition ou d'un adaptateur de votre choix. Pour exécuter une tâche d'analyse en bloc, procédez comme suit :

1. Connectez-vous à la console Amazon Rekognition Console de gestion AWS et ouvrez-la à l'adresse. https://console.aws.amazon.com/rekognition/
2. Dans le volet gauche, sélectionnez Analyse en bloc. La page d'accueil de l'analyse en bloc s'affiche. Choisissez Démarrer l'analyse en bloc. Vue d'ensemble de la fonctionnalité d'analyse en bloc indiquant les étapes à suivre pour télécharger des images, attendre l'analyse, examiner les résultats et éventuellement vérifier les prédictions du modèle. Répertorie les tâches d'analyse en bloc récentes pour la modération du contenu à l'aide du modèle de base.



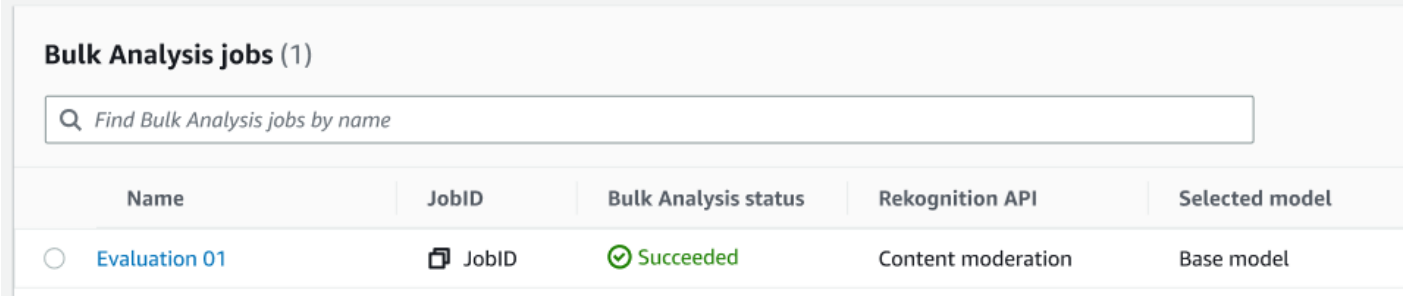
3. Si c’est la première fois que vous créez un adaptateur, vous serez invité à créer un compartiment Amazon Simple Storage Service pour stocker les fichiers liés à votre projet et à votre adaptateur. Choisissez Créez un compartiment Amazon S3.
4. Sélectionnez l’adaptateur que vous souhaitez utiliser pour l’analyse globale à l’aide du menu déroulant Choisir un adaptateur. Si aucun adaptateur n’est choisi, le modèle de base sera utilisé par défaut. Dans le cadre de ce didacticiel, ne choisissez pas d’adaptateur.



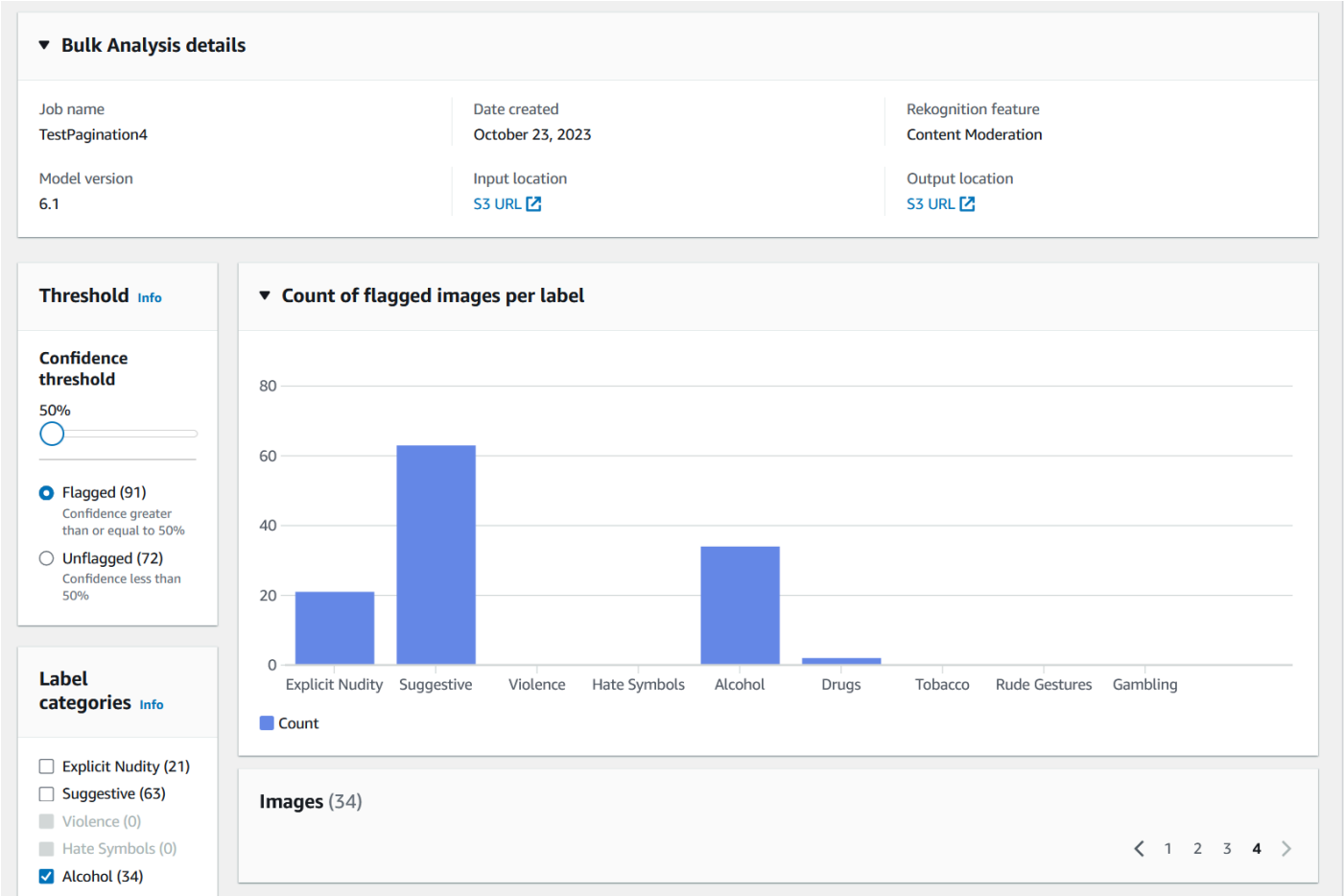
5. Dans le champ Nom de la tâche d'analyse en bloc, saisissez le nom de la tâche d'analyse en bloc.
6. Choisissez une valeur pour le seuil de confiance minimum. Les prédictions d'étiquettes dont le seuil de confiance est inférieur au seuil de confiance que vous avez choisi ne seront pas renvoyées. Notez que lorsque vous évaluerez les performances du modèle ultérieurement, vous ne pourrez pas ajuster le seuil de confiance en dessous du seuil de confiance minimum que vous avez choisi.
7. Dans cette étape, vous allez également fournir les images que vous souhaitez analyser avec l'analyse en bloc. Ces images peuvent également être utilisées pour entraîner votre adaptateur. Vous pouvez choisir Charger des images depuis votre ordinateur ou Importer des images depuis le compartiment Amazon S3. Si vous choisissez d'importer vos documents depuis un compartiment Amazon S3, indiquez le chemin d'accès au compartiment et au dossier contenant vos images

d'entraînement. Si vous téléchargez vos documents directement depuis votre ordinateur, notez que vous ne pouvez télécharger que 50 images à la fois.

8. Après avoir renseigné ces informations, choisissez Démarrer l'analyse. Cela lancera le processus d'analyse à l'aide du modèle de base de Rekognition.
9. Vous pouvez vérifier le statut de votre tâche d'analyse en bloc en vérifiant le statut d'analyse en bloc de la tâche sur la page principale d'analyse en bloc. Lorsque le statut de l'analyse en bloc devient « Réussi », les résultats de l'analyse sont prêts à être examinés.



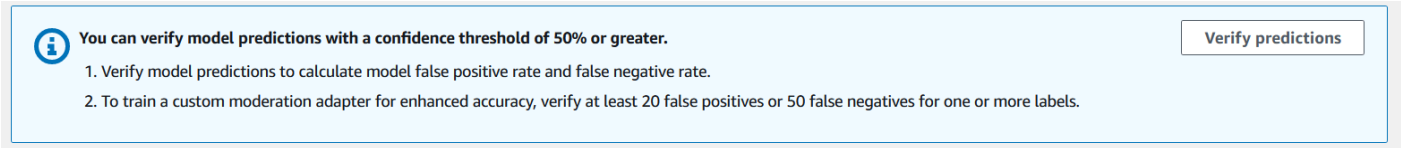
10. Choisissez l'analyse que vous avez créée dans la liste des tâches d'analyse en bloc.
11. Sur la page de détails de l'analyse en bloc, vous pouvez voir les prédictions que le modèle de base de Rekognition a établies pour les images que vous avez téléchargées.
12. Passez en revue les performances du modèle de base. Vous pouvez modifier le seuil de confiance que doit respecter votre adaptateur pour attribuer une étiquette à une image à l'aide du curseur du seuil de confiance. Le nombre d'instances signalées et non signalées change au fur et à mesure que vous ajustez le seuil de confiance. Le volet Catégories d'étiquettes affiche les catégories de haut niveau reconnues par Rekognition, et vous pouvez sélectionner une catégorie dans cette liste pour afficher toutes les images auxquelles cette étiquette a été attribuée.



Vérifiez les prédictions

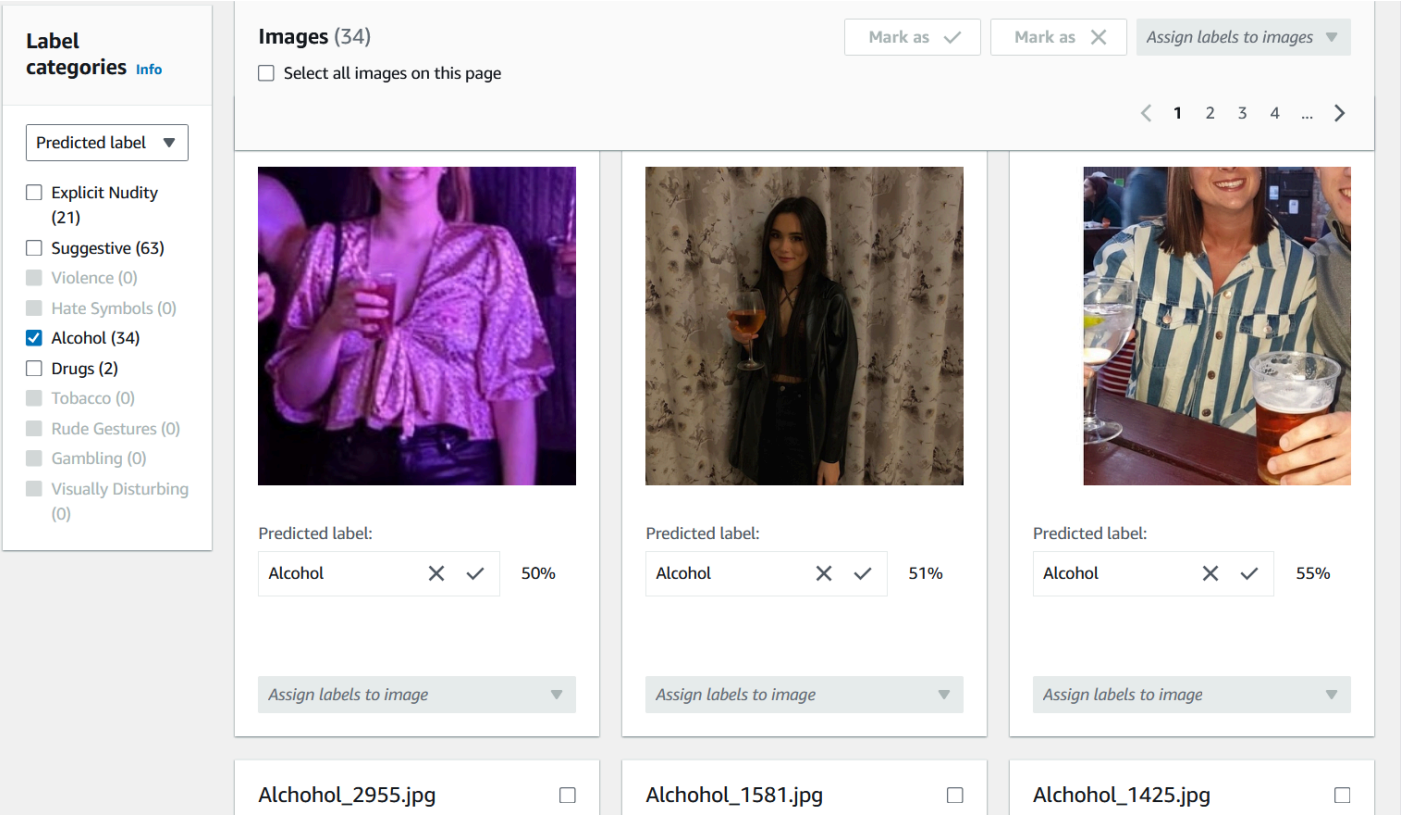
Si vous avez vérifié la précision du modèle de base de Rekognition ou d'un adaptateur choisi, et que vous souhaitez améliorer cette précision, vous pouvez utiliser le flux de travail de vérification :

1. Une fois que vous avez terminé d'examiner les performances du modèle de base, vous avez besoin de vérifier les prévisions. La correction des prédictions vous permet de former un adaptateur. Choisissez Vérifier les prédictions en haut de la page d'analyse en bloc.

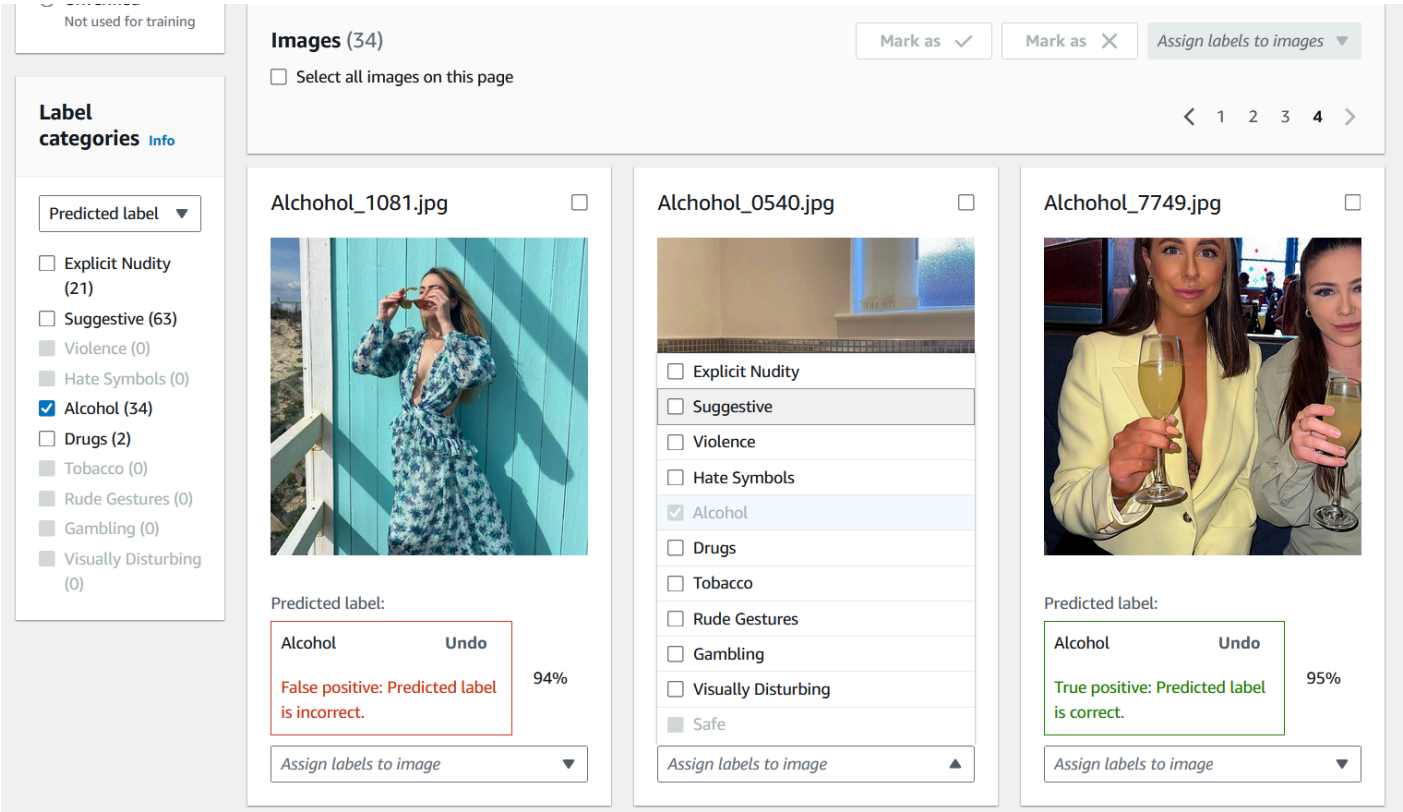


2. Sur la page Vérifier les prédictions, vous pouvez voir toutes les images que vous avez fournies au modèle de base de Rekognition, ou à un adaptateur choisi, ainsi que l'étiquette prévue pour chaque image. Vous devez vérifier que chaque prédiction est correcte ou incorrecte à l'aide des boutons situés sous l'image. Utilisez le bouton « X » pour marquer une prédiction comme

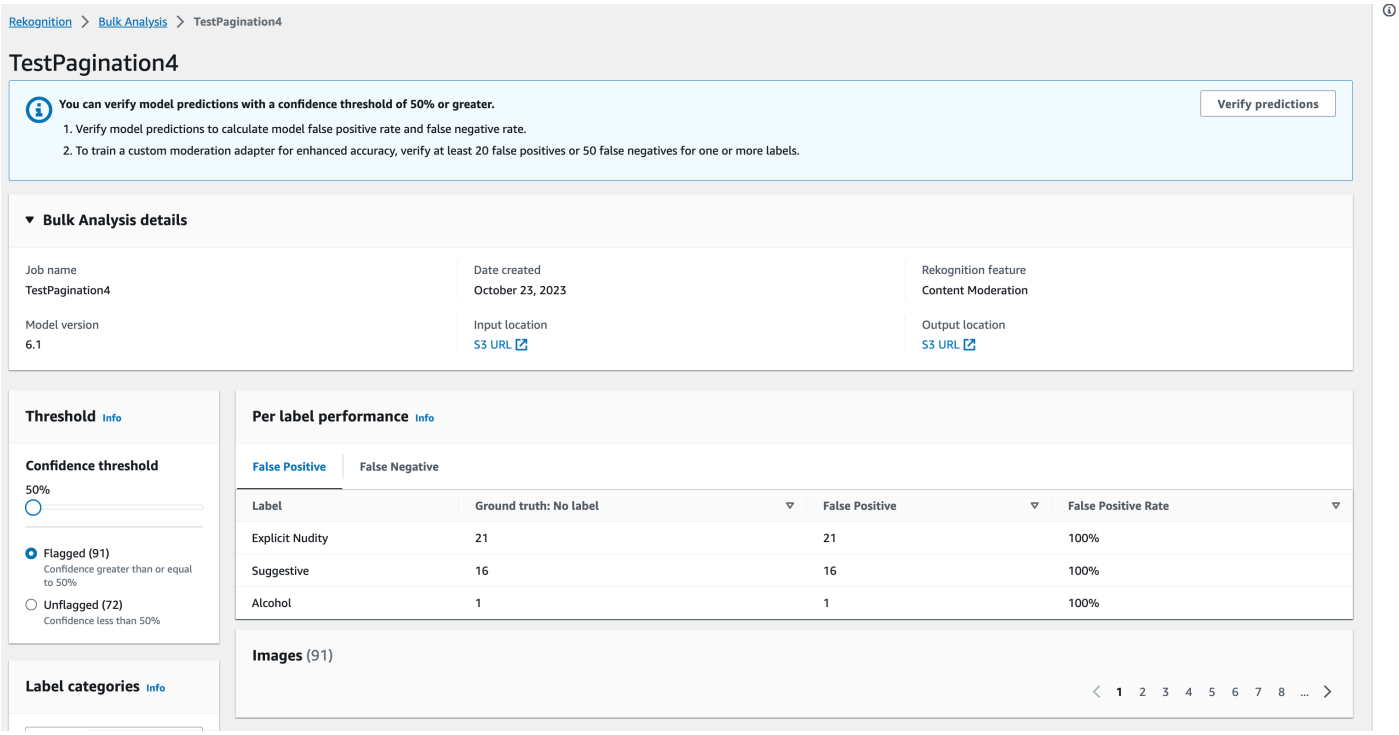
incorrecte et le bouton de coche pour marquer une prédiction comme correcte. Pour entraîner un adaptateur, vous devez vérifier au moins 20 prédictions représentant des faux positifs, et 50 prédictions représentant des faux négatifs pour une étiquette donnée. Plus vous vérifiez de prédictions, meilleures seront les performances de l'adaptateur.



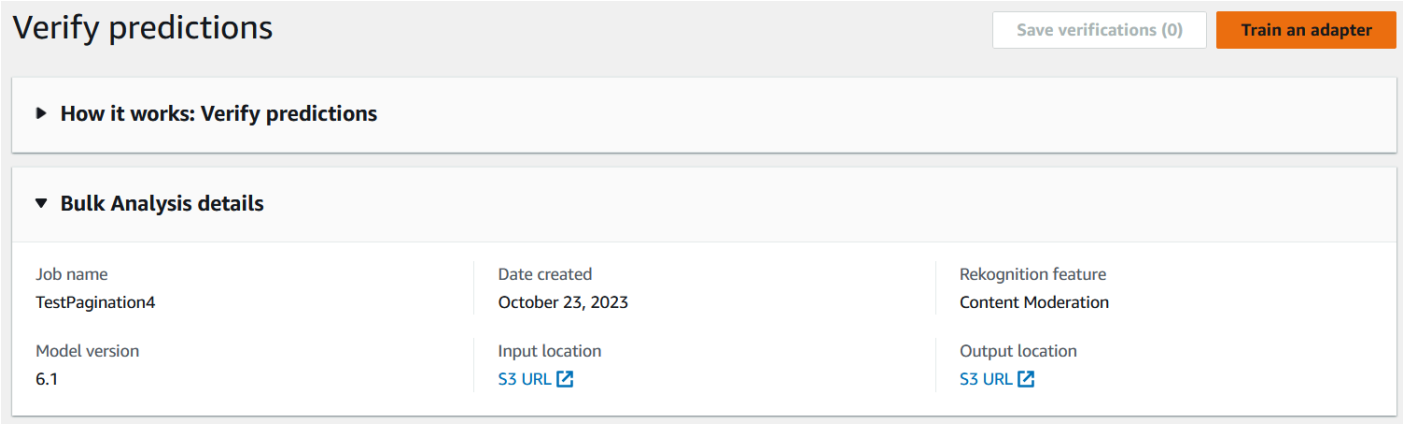
Une fois que vous avez vérifié une prédiction, le texte sous l'image change pour indiquer le type de prédiction que vous avez vérifié. Une fois que vous avez vérifié une image, vous pouvez également ajouter des étiquettes supplémentaires à l'image à l'aide du menu Attribuer des étiquettes à l'image. Vous pouvez voir quelles images sont signalées ou non par le modèle pour le seuil de confiance que vous avez choisi, ou filtrer les images par catégorie.



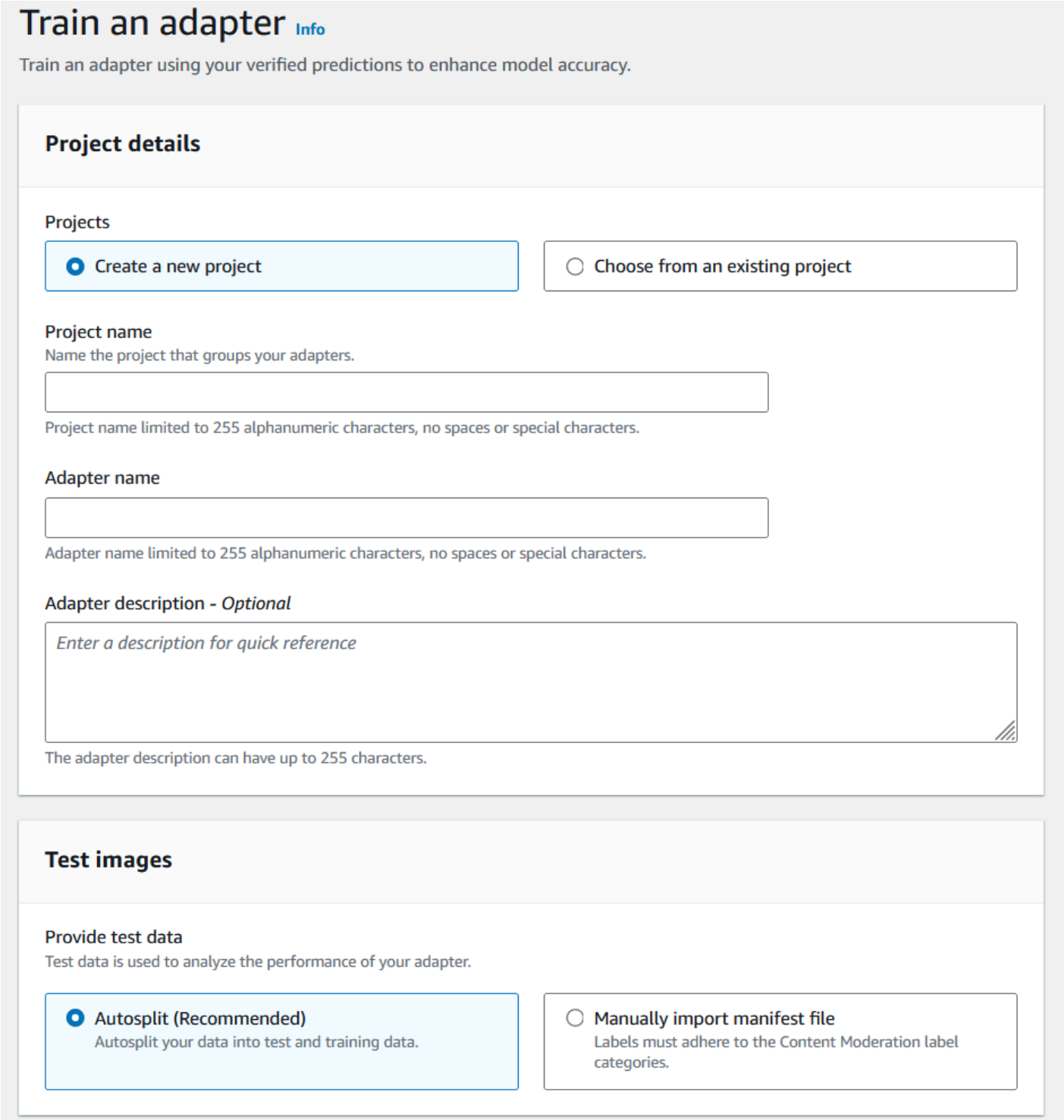
3. Une fois que vous avez terminé de vérifier toutes les prédictions que vous souhaitez vérifier, vous pouvez consulter les statistiques concernant vos prévisions vérifiées dans la section Performances par étiquette de la page de vérification. Vous pouvez également revenir à la page des détails de l'analyse en bloc pour consulter ces statistiques.



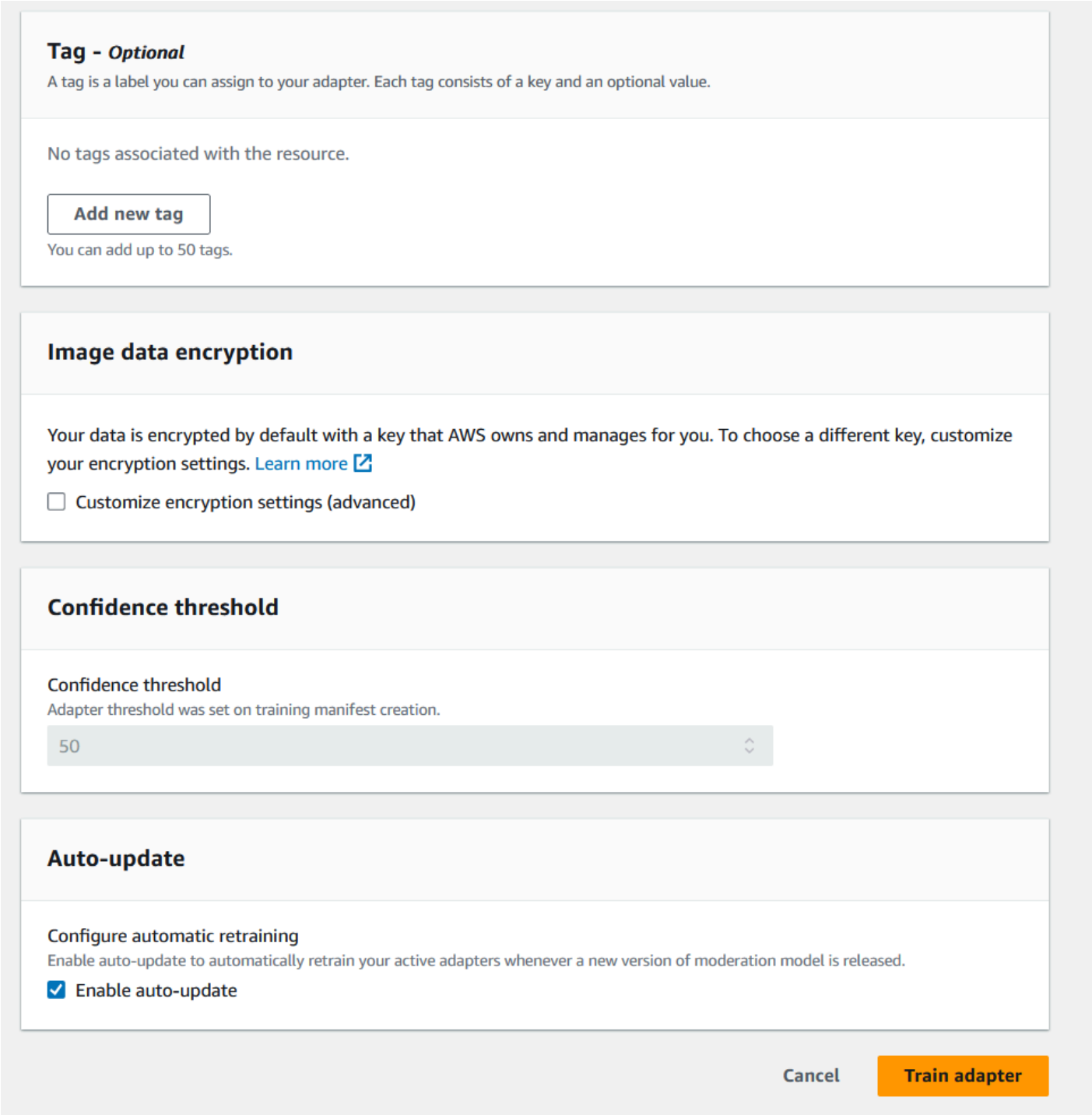
4. Lorsque vous êtes satisfait des statistiques concernant les performances par étiquette, revenez sur la page Vérifier les prévisions, puis sélectionnez le bouton Entraîner un adaptateur pour commencer à entraîner votre adaptateur.



5. Sur la page Former un adaptateur, vous êtes invité à créer un projet ou à choisir un projet existant. Nommez le projet et l'adaptateur qui sera inclus dans le projet. Vous devez également spécifier la source de vos images de test. Lorsque vous spécifiez les images, vous pouvez choisir Autosplit pour que Rekognition utilise automatiquement une partie de vos données d'entraînement comme images de test, ou vous pouvez spécifier manuellement un fichier manifeste. Il est recommandé de choisir Autosplit.



6. Spécifiez les balises que vous souhaitez, ainsi qu'une AWS KMS clé si vous ne souhaitez pas utiliser la AWS clé par défaut. Il est recommandé de laisser Auto-update cette option activée.
7. Choisissez Former l’adaptateur.

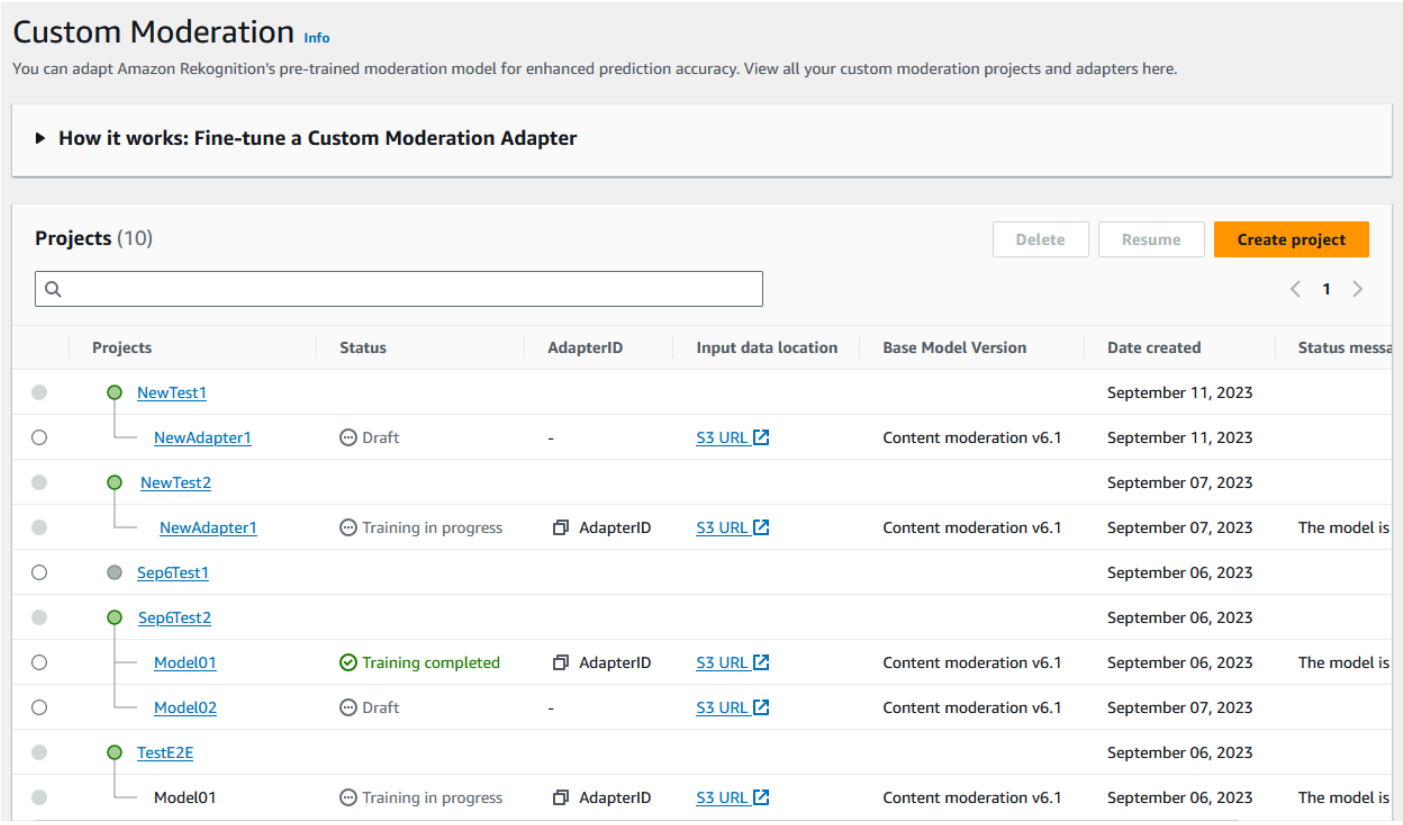


8. Une fois que le statut de votre adaptateur sur la page d'accueil de la modération personnalisée est passé à « Entraînement terminé », vous pouvez vérifier les performances de votre adaptateur. Pour plus d'informations, consultez Vérification des performances de l'adaptateur.

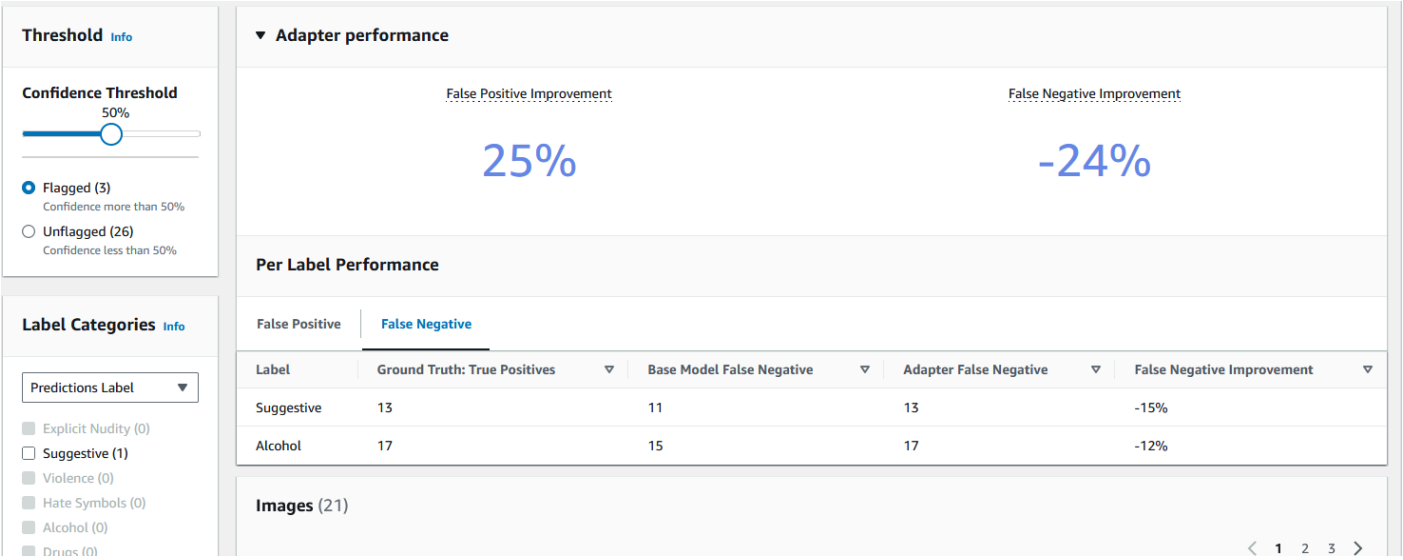
Vérification des performances de l'adaptateur

Pour vérifier les performances de votre adaptateur :

1. Lorsque vous utilisez la console, vous pouvez voir l'état de tous les adaptateurs associés à un projet sous l'onglet Projets de la page d'accueil de la modération personnalisée. Accédez à la page d'accueil de la modération personnalisée.



2. Sélectionnez l'adaptateur que vous souhaitez vérifier dans cette liste. Sur la page de détails de l'adaptateur suivante, vous pouvez voir diverses mesures relatives à l'adaptateur.

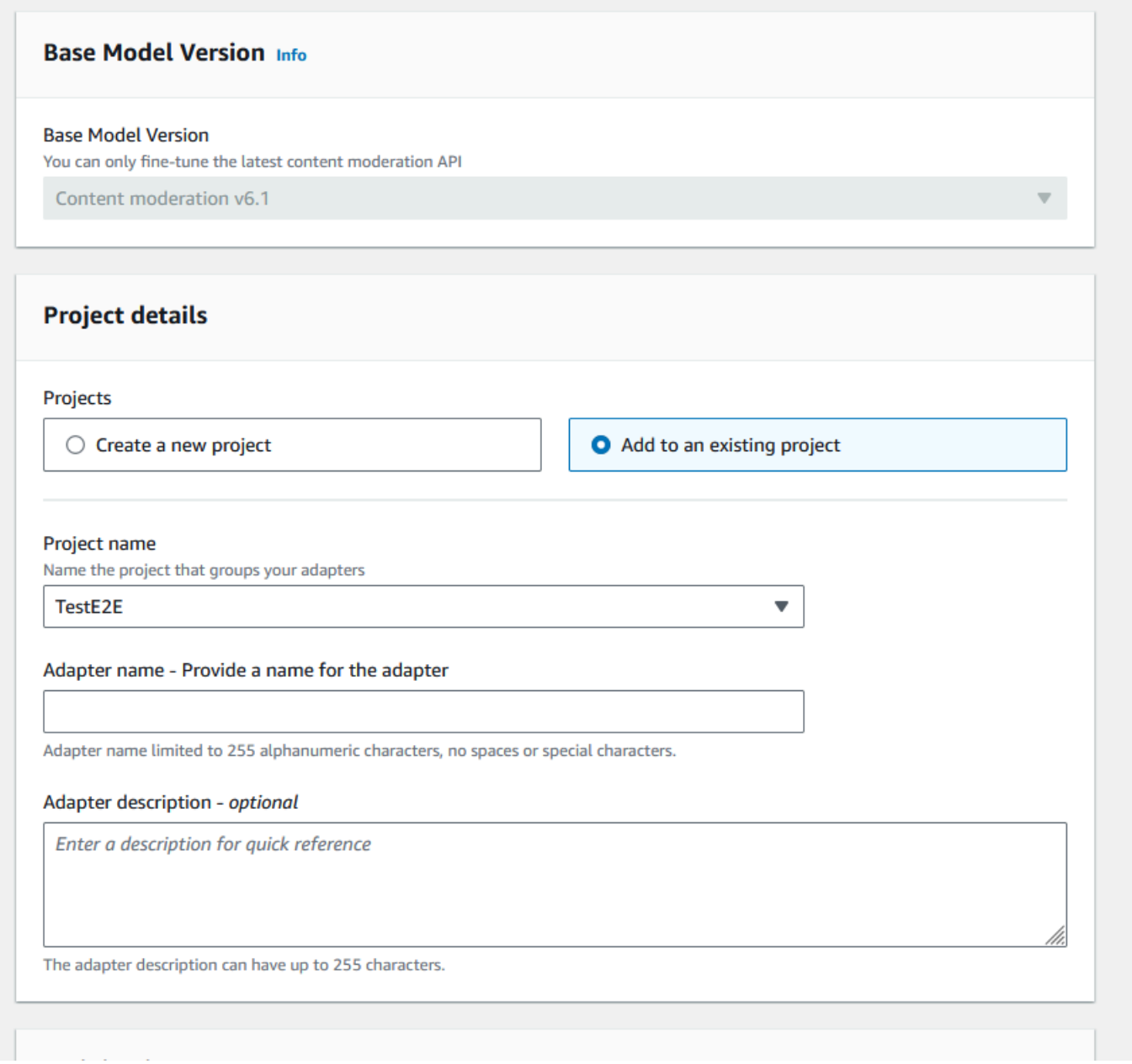


3. Le volet Seuil vous permet de modifier le seuil de confiance minimum que doit respecter votre adaptateur pour attribuer une étiquette à une image. Le nombre d'instances signalées et non signalées changera au fur et à mesure que vous ajusterez le seuil de confiance. Vous pouvez également filtrer par catégorie d'étiquette pour voir les statistiques des catégories que vous avez sélectionnées. Définissez le seuil que vous avez choisi.
4. Vous pouvez évaluer les performances de votre adaptateur sur la base de vos données de test en examinant les mesures du panneau Performances de l'adaptateur. Ces mesures sont calculées en comparant les extractions de l'adaptateur aux annotations « réalistes » figurant sur le kit de test.

Le volet des performances de l'adaptateur indique les taux d'amélioration de faux positifs et de faux négatifs de l'adaptateur que vous avez créé. L'onglet Performances par étiquette peut être utilisé pour comparer les performances de l'adaptateur et du modèle de base pour chaque catégorie d'étiquette. Il indique le nombre de prédictions représentant des faux positifs et des faux négatifs établies à la fois par le modèle de base et par l'adaptateur, stratifiées par catégorie d'étiquette. En consultant ces métriques, vous pouvez déterminer les domaines dans lesquels l'adaptateur doit être amélioré. Pour obtenir plus d'informations sur ces métriques, consultez Évaluation et amélioration de votre adaptateur.

Pour améliorer les performances, vous pouvez collecter davantage d'images d'entraînement, puis créer un nouvel adaptateur intégré au projet. Il vous suffit de retourner à la page d'accueil de la modération personnalisée et de créer un nouvel adaptateur dans votre projet, en fournissant davantage d'images d'entraînement sur lesquelles l'adaptateur sera entraîné. Cette fois, choisissez l'option Ajouter à un projet existant au lieu de Créer un nouveau projet, puis sélectionnez le projet dans lequel vous souhaitez créer le nouvel adaptateur dans le menu déroulant Nom du projet.

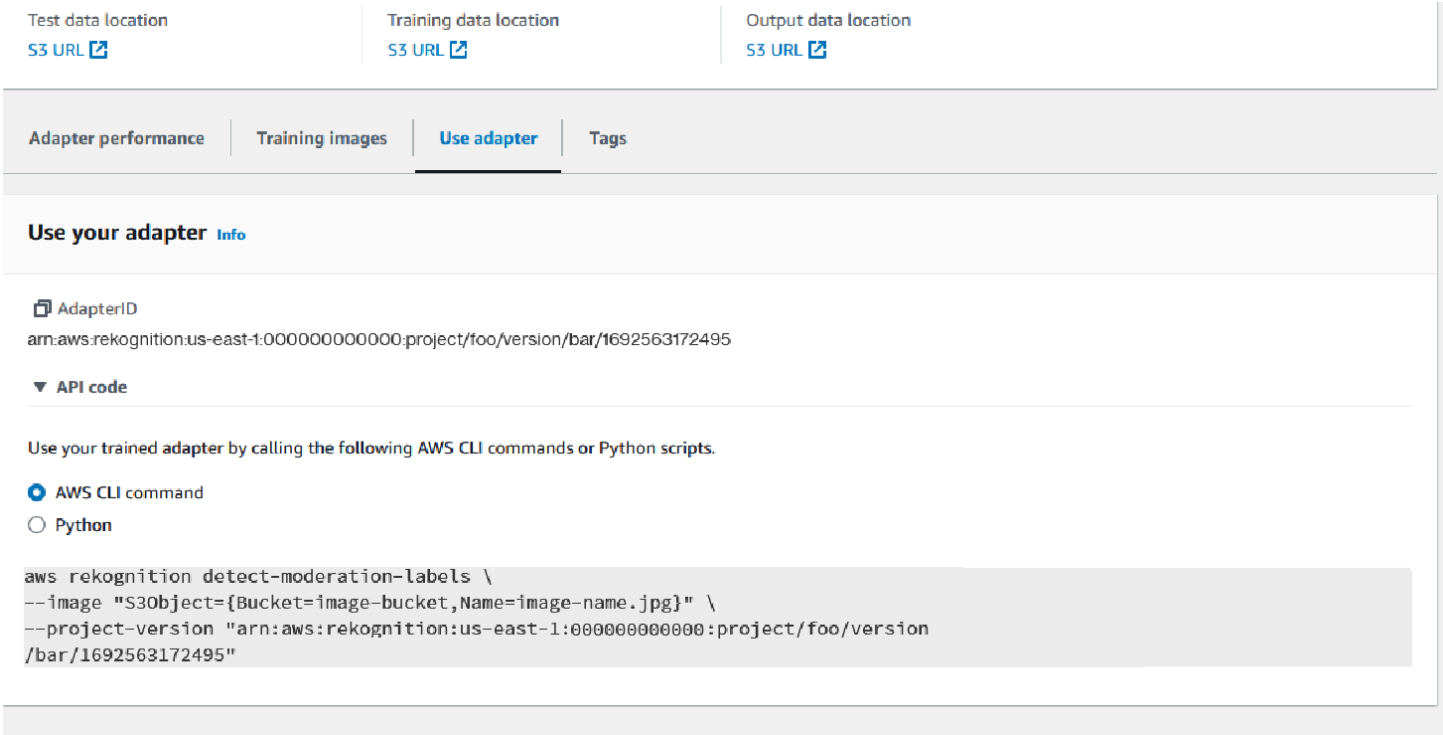
Comme précédemment, annotez vos images ou fournissez un fichier manifeste contenant des annotations.



Utilisation de votre adaptateur

Une fois que vous avez créé votre adaptateur, vous pouvez le fournir à une opération de Rekognition prise en charge telle que. DetectModerationLabels Pour voir des exemples de code que vous pouvez utiliser pour effectuer des inférences avec votre adaptateur, sélectionnez l'onglet « Utiliser l'adaptateur », où vous pouvez voir des exemples de code pour la AWS CLI et pour Python. Vous pouvez également consulter la section correspondante de la documentation relative à l'opération

pour laquelle vous avez créé un adaptateur pour voir d'autres exemples de code, des instructions de configuration et un exemple de code JSON.



Suppression de votre adaptateur et de votre projet

Vous pouvez supprimer des adaptateurs individuels ou supprimer votre projet. Vous devez supprimer chaque adaptateur associé à votre projet avant de pouvoir supprimer le projet lui-même.

1. Pour supprimer un adaptateur associé au projet, choisissez-le, puis choisissez Supprimer.
2. Pour supprimer un projet, sélectionnez le projet que vous souhaitez supprimer, puis choisissez Supprimer.

Évaluation et amélioration de votre adaptateur

Après chaque cycle de formation sur l'adaptateur, vous devez examiner les indicateurs de performance de l'outil de console de Rekognition afin de déterminer à quel niveau l'adaptateur se situe par rapport au niveau de performance souhaité. Vous pouvez ensuite améliorer encore la précision de votre adaptateur pour vos images en téléchargeant un nouveau lot d'images d'entraînement, et en entraînant un nouvel adaptateur dans votre projet. Une fois que vous avez créé une version améliorée de l'adaptateur, vous pouvez utiliser la console pour supprimer les anciennes versions de l'adaptateur dont vous n'avez plus besoin.

Vous pouvez également récupérer des métriques à l'aide de l'opération DescribeProjectVersionsAPI.

Métriques de performances

Une fois que vous avez terminé le processus de formation et créé votre adaptateur, il est important d'évaluer dans quelle mesure l'adaptateur extrait les informations de vos images.

Deux métriques sont fournis dans la console de Rekognition pour vous aider à analyser les performances de votre adaptateur : amélioration des faux positifs et amélioration des faux négatifs.

Vous pouvez consulter ces mesures pour n'importe quel adaptateur en sélectionnant l'onglet « Performances de l'adaptateur » dans la partie adaptateur de la console. Le volet des performances de l'adaptateur indique les taux d'amélioration des faux positifs et des faux négatifs de l'adaptateur que vous avez créé.

L'amélioration des faux positifs mesure en quoi la reconnaissance des faux positifs par l'adaptateur s'est améliorée par rapport au modèle de base. Si la valeur d'amélioration des faux positifs est de 25 %, cela signifie que l'adaptateur a amélioré sa reconnaissance des faux positifs de 25 % sur le jeu de données de test.

L'amélioration des faux négatifs mesure en quoi la reconnaissance des faux négatifs par l'adaptateur s'est améliorée par rapport au modèle de base. Si la valeur d'amélioration des faux négatifs est de 25 %, cela signifie que l'adaptateur a amélioré sa reconnaissance des faux négatifs de 25 % sur le jeu de données de test.

L'onglet Performances par étiquette peut être utilisé pour comparer les performances de l'adaptateur et du modèle de base pour chaque catégorie d'étiquette. Il indique le nombre de prédictions représentant des faux positifs et des faux négatifs établies à la fois par le modèle de base et par l'adaptateur, stratifiées par catégorie d'étiquette. En consultant ces métriques, vous pouvez déterminer les domaines dans lesquels l'adaptateur doit être amélioré.

Par exemple, si le taux de faux négatifs du modèle de base pour la catégorie des étiquettes alcoolisées est de 15 alors que le taux de faux négatifs de l'adaptateur est de 15 ou plus, vous savez que vous devez vous concentrer sur l'ajout d'images supplémentaires contenant l'étiquette d'alcool lors de la création d'un nouvel adaptateur.

Lorsque vous utilisez les opérations de l'API Rekognition, la métrique est renvoyée lors de l'appel F1-Score de l'opération. DescribeProjectVersions

Amélioration de votre modèle

Le déploiement d'un adaptateur est un processus itératif, car vous devrez probablement entraîner un adaptateur plusieurs fois pour atteindre votre niveau de précision cible. Après avoir créé et entraîné votre adaptateur, vous devez tester et évaluer ses performances sur différents types d'étiquettes.

Si la précision de votre adaptateur fait défaut dans un domaine, ajoutez de nouveaux exemples de ces images afin d'améliorer les performances de l'adaptateur pour ces étiquettes. Essayez de fournir à l'adaptateur des exemples supplémentaires et variés qui reflètent les cas où il rencontre des difficultés. En fournissant à votre adaptateur des images représentatives et variées, vous lui permettez de traiter divers exemples concrets.

Après avoir ajouté de nouvelles images à votre kit d'entraînement, réentraînez l'adaptateur, puis réévaluez-le sur votre ensemble de test et sur les étiquettes. Répétez ce processus jusqu'à ce que l'adaptateur atteigne le niveau de performance souhaité. Si vous fournissez des images et des annotations plus représentatives, les scores de faux positifs et de faux négatifs s'amélioreront progressivement au fil des itérations d'entraînement successives.

Formats de fichiers manifestes

Les sections suivantes présentent des exemples de formats de fichier manifeste pour les fichiers d'entrée, de sortie et d'évaluation.

Manifeste d'entrée

Un fichier manifeste est un fichier délimité par json-line, chaque ligne contenant un fichier JSON contenant des informations sur une seule image.

Chaque entrée du manifeste d'entrée doit contenir le champ `source-ref` indiquant le chemin d'accès à l'image dans le compartiment Amazon S3 et, pour une modération personnalisée, le champ `content-moderation-groundtruth` contenant des annotations de base. Toutes les images d'un jeu de données doivent se trouver dans le même compartiment. La structure est commune aux fichiers manifestes de formation et de test.

L'opération de modération personnalisée `CreateProjectVersion` utilise les informations fournies dans le manifeste d'entrée pour entraîner un adaptateur.

L'exemple suivant est une ligne d'un fichier manifeste pour une seule image contenant une seule classe dangereuse :

```
{
```

```
    "source-ref": "s3://foo/bar/1.jpg",
    "content-moderation-groundtruth": {
         "ModerationLabels": [
             {
                 "Name": "Rude Gesture"
             }
         ]
    }
}
```

L'exemple suivant est une ligne d'un fichier manifeste pour une seule image dangereuse contenant plusieurs classes dangereuses, en particulier Nudité et Gestes grossiers.

```
{
    "source-ref": "s3://foo/bar/1.jpg",
    "content-moderation-groundtruth": {
         "ModerationLabels": [
             {
                 "Name": "Rude Gesture"
             },
             {
                 "Name": "Nudity"
             }
         ]
    }
}
```

L'exemple suivant est une ligne d'un fichier manifeste pour une seule image ne contenant aucune classe dangereuses :

```
{
    "source-ref": "s3://foo/bar/1.jpg",
    "content-moderation-groundtruth": {
         "ModerationLabels": []
    }
}
```

Pour obtenir la liste complète des étiquettes prises en charge, reportez-vous à la section Modération du contenu.

Manifeste de sortie

À la fin d'une tâche de formation, un fichier manifeste de sortie est renvoyé. Le fichier manifeste de sortie est un fichier JSON-line délimité dont chaque ligne contient un code JSON contenant les informations relatives à une seule image. Le chemin d'accès à Amazon S3 OutputManifest peut être obtenu à partir de `DescribeProjectVersion` la réponse :

- `TrainingDataResult.Output.Assets[0].GroundTruthManifest.S3Object` pour jeux de données d'entraînement
- `TestingDataResult.Output.Assets[0].GroundTruthManifest.S3Object` pour jeux de données de test

Les informations suivantes sont renvoyées pour chaque entrée du manifeste de sortie :

Nom de clé	Description
`source-ref`	Référence à une image dans s3 qui a été fournie dans le manifeste d'entrée
`content-moderation-groundtruth`	Annotations fondées sur la vérité fournies dans le manifeste d'entrée
`detect-moderation-labels`	Prédictions de l'adaptateur, uniquement incluses dans l'ensemble de données de test
`detect-moderation-labels-base-model`	Prédictions du modèle de base, faisant partie de l'ensemble de données de test uniquement

Les prédictions de l'adaptateur et du modèle de base sont renvoyées à la ConfidenceTrehsold version 5.0 dans un format similaire à celui de la DetectModerationLabelsréponse.

L'exemple suivant montre la structure des prédictions de l'adaptateur et du modèle de base :

```
{
   "ModerationLabels": [
      {
         "Confidence": number,
         "Name": "string",
         "ParentName": "string"
```

```
      }
   ],
   "ModerationModelVersion": "string",
   "ProjectVersion": "string"
}
```

Pour obtenir la liste complète des étiquettes renvoyées, reportez-vous à la section Modération du contenu.

Manifeste des résultats d'évaluation

À la fin d'une tâche de formation, un fichier manifeste des résultats d'évaluation est renvoyé. Le manifeste des résultats de l'évaluation est un fichier de sortie JSON généré par la tâche de formation. Il contient des informations concernant les performances de l'adaptateur sur les données de test.

Amazon S3 Le chemin d'accès au manifeste des résultats de l'évaluation peut être obtenu à partir du `EvaluationResult.Summary.S3Object` champ de la DescribeProejctVersion réponse.

La structure du manifeste des résultats d'évaluation est présentée dans l'exemple suivant :

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

Le fichier manifeste d’évaluation contient :

- Résultats agrégés tels que définis par `F1Score`
- Détails du travail d'évaluation ProjectVersionArn, notamment le nombre d'images d'entraînement, le nombre d'images de test et les étiquettes sur lesquelles l'adaptateur a été formé.
- Agrégé TruePositive TrueNegative, FalsePositive, et FalseNegative résultats pour les performances du modèle de base et de l'adaptateur.
- Par étiquette TruePositive, TrueNegative FalsePositive, et FalseNegative résultats pour les performances du modèle de base et de l'adaptateur, calculés au seuil de confiance en entrée.
- Agrégés et par étiquette TruePositive, TrueNegative FalsePositive, et FalseNegative résultats pour les performances du modèle de base et de l'adaptateur à différents seuils de confiance. Le seuil de confiance varie de 5 à 100 par paliers de 5.

Bonnes pratiques relatives aux adaptateurs d’entraînement

Il est recommandé de respecter les meilleures pratiques suivantes lors de la création, la formation et l’utilisation de vos adaptateurs :

1. Les exemples de données d’image doivent capturer les erreurs représentatives que les clients ont l’intention de supprimer. Si le modèle commet des erreurs répétées sur des images visuellement similaires, assurez-vous d’apporter un grand nombre de ces images pour l’entraînement.
2. Au lieu de n’ajouter que des images indiquant que le modèle commet des erreurs sur une étiquette de modération en particulier, assurez-vous également d’inclure des images indiquant que le modèle ne commet pas d’erreurs sur cette étiquette de modération.
3. Fournir un minimum de 50 échantillons faussement négatifs OU 20 échantillons faussement positifs pour la formation et un minimum de 20 échantillons pour les tests. Toutefois, fournissez autant d’images annotées que possible pour améliorer les performances de l’adaptateur.
4. Annotez toutes les étiquettes qui vous intéressent pour toutes les images : si vous décidez que vous devez annoter l’occurrence d’une étiquette sur une image, veillez à annoter l’occurrence de cette étiquette sur toutes les autres images.
5. Les exemples de données d’image doivent contenir autant de variations que possible sur l’étiquette, en se concentrant sur les instances représentatives des images qui seront analysées dans un environnement de production.

Configuration des AutoUpdate autorisations

Rekognition prend en charge AutoUpdate cette fonctionnalité pour les adaptateurs personnalisés. Cela signifie que le recyclage automatique fait de son mieux lorsque le AutoUpdate drapeau est ACTIVÉ sur un projet. Ces mises à jour automatiques nécessitent l'autorisation d'accéder à vos Training/Testing ensembles de données et la AWS KMS clé avec laquelle vous formerez votre adaptateur client. Vous pouvez fournir ces autorisations en suivant les étapes ci-dessous.

Autorisations pour le compartiment Amazon S3

Par défaut, tous les objets et les compartiments Amazon S3 sont privés. Seul le propriétaire de la ressource, le AWS compte qui a créé le compartiment, peut accéder au compartiment et aux objets qu'il contient. Le propriétaire de la ressource peut toutefois accorder des autorisations d’accès à d’autres ressources et à d’autres utilisateurs en créant une politique de compartiment.

Si vous souhaitez créer ou modifier un compartiment Amazon S3 à utiliser comme source de jeux de données d'entrée et comme destination des résultats d'entraînement dans le cadre d'une formation d'adaptation personnalisée, vous devez modifier davantage la politique de compartiment. Pour lire ou écrire dans un compartiment Amazon S3, Rekognition doit disposer des autorisations suivantes.

Politique Amazon S3 requise pour Rekognition

La Rekognition nécessite une politique d'autorisation comportant les attributs suivants :

- ID d'instruction (Sid)
- Nom du compartiment
- Nom du principal de service pour Rekognition.
- Ressources requises pour Rekognition, compartiment et ensemble de son contenu
- Actions requises devant être effectuées par Rekognition.

La politique suivante autorise Rekognition à accéder à un compartiment Amazon S3 lors du recyclage automatique.

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

Vous pouvez suivre ce guide pour ajouter la politique de compartiment ci-dessus à votre compartiment S3.

Pour plus d'informations sur les politiques relatives aux compartiments, cliquez ici.

AWS KMS Autorisations clés

La Rekognition vous permet de fournir un adaptateur personnalisé en KmsKeyId option lors de l'entraînement. Lorsqu'elle est fournie, Rekognition utilise cette clé pour chiffrer les images d'entraînement et de test copiées dans le service pour l'entraînement des modèles. La clé est également utilisée pour chiffrer les résultats d'entraînement et les fichiers manifestes écrits dans le compartiment Amazon S3 de sortie (OutputConfig).

Si vous choisissez de fournir une clé KMS comme entrée pour votre formation d'adaptateur personnalisée (c'est-à-dire `Rekognition:CreateProjectVersion`), vous devez modifier davantage la stratégie de clé KMS afin de permettre au principal du service de Rekognition d'utiliser cette clé pour un recyclage automatique à l'avenir. Rekognition doit disposer des autorisations suivantes.

Politique clé en matière de AWS KMS rekognition

Amazon Rekognition nécessite une politique d'autorisation comportant les attributs suivants :

- ID d'instruction (Sid)
- Nom du principal de service pour Amazon Rekognition.
- Actions requises qu'Amazon Rekognition doit effectuer.

La stratégie de clé suivante permet à Amazon Rekognition d'accéder à une clé Amazon KMS lors du recyclage automatique :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "KeyPermissions",
            "Effect": "Allow",
            "Principal": {
                "Service": "rekognition.amazonaws.com"
```

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

Vous pouvez suivre ce guide pour ajouter la AWS KMS politique ci-dessus à votre AWS KMS clé.

Pour plus d'informations sur les AWS KMS politiques, cliquez ici.

AWS Notification du Health Dashboard pour Rekognition

Votre AWS Health Dashboard fournit une assistance pour les notifications provenant de Rekognition. Ces notifications fournissent des conseils de sensibilisation et de correction concernant les modifications prévues dans les modèles de Rekognition susceptibles d'affecter vos applications. Seuls les événements spécifiques à la fonction de modération du contenu de Rekognition sont actuellement disponibles.

Le AWS Health Dashboard fait partie du service AWS de santé. Il ne nécessite aucune configuration et peut être affiché par n'importe quel utilisateur authentifié dans votre compte. Pour de plus amples informations, veuillez consulter Démarrez avec le tableau de bord AWS Health.

Si vous recevez un avis similaire aux messages suivants, il doit être traité comme une alarme incitant à une action.

Exemple d'avis : une nouvelle version du modèle est disponible pour la modération du contenu Rekognition.

Rekognition publie l'`AWS_MODERATION_MODEL_VERSION_UPDATE_NOTIFICATION`événement sur le AWS Health Dashboard pour indiquer qu'une nouvelle version du modèle de modération a été publiée. Cet événement est important si vous utilisez l' DetectModerationLabels API et les adaptateurs avec cette API. Les nouveaux modèles peuvent avoir un impact sur la qualité en fonction de votre cas d'utilisation, et remplacent éventuellement les versions précédentes. Il est recommandé de valider la qualité de votre modèle et de connaître les délais de mise à jour du modèle lorsque vous recevez cette alerte.

Si vous recevez une notification de mise à jour de la version du modèle, vous devez la traiter comme une alarme vous demandant de prendre des mesures. Si vous n'utilisez pas d'adaptateurs, vous devez évaluer la qualité du modèle mis à jour en fonction de votre cas d'utilisation actuel. Si vous utilisez des adaptateurs, vous devez former de nouveaux adaptateurs avec le modèle mis à jour et évaluer leur qualité. Si vous avez configuré l'entraînement automatique, les nouveaux adaptateurs sont automatiquement entraînés, vous pouvez ensuite évaluer leur qualité.

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

Consultez la section Surveillance des événements AWS Health avec Amazon EventBridge pour détecter les événements de AWS santé et y réagir à l'aide de EventBridge.

Vérification de contenu inappropriée avec Amazon Augmented AI

Amazon Augmented AI (Amazon A2I) vous permet de créer les workflows requis pour la vérification humaine des prédictions de Machine Learning.

Amazon Rekognition est directement intégré à Amazon A2I afin que vous puissiez facilement implémenter une vérification humaine pour la détection d'images dangereuses. Amazon A2I fournit un workflow de vérification humaine pour la modération des images. Cela vous permet de consulter facilement les prédictions d'Amazon Rekognition. Vous pouvez définir des seuils de confiance pour votre cas d'utilisation et les ajuster au fil du temps. Avec Amazon A2I, vous pouvez utiliser un pool d'évaluateurs au sein de votre propre organisation ou Amazon Mechanical Turk. Vous pouvez également recourir à des vendeurs de main d'œuvre présélectionnés par AWS pour garantir la qualité et la conformité aux procédures de sécurité.

Les étapes suivantes vous expliquent comment configurer Amazon A2I avec Amazon Rekognition. Tout d'abord, vous créez une définition de flux avec Amazon A2I avec les conditions qui déclenchent l'examen réalisé par un humain. Ensuite, vous transmettez l'Amazon Resource Name (ARN) de la définition de flux à l'opération `DetectModerationLabel` Amazon Rekognition. Dans la réponse `DetectModerationLabel`, vous pouvez voir si une vérification humaine est nécessaire. Les résultats de la vérification humaine sont disponibles dans un compartiment Amazon S3 défini par la définition du flux.

Pour visionner une démonstration complète de l'utilisation d'Amazon A2I avec Amazon Rekognition, consultez l'un des didacticiels suivants dans le manuel Amazon AI Developer Guide. SageMaker

- Démonstration : démarrer dans la console Amazon A2I
- Démonstration : démarrer à l'aide de l'API Amazon A2I

  Pour commencer à utiliser l'API, vous pouvez également exécuter un exemple de bloc-notes Jupyter. Consultez Utiliser une instance de SageMaker bloc-notes avec Amazon A2I Jupyter Notebook pour utiliser l'intégration du bloc-notes Amazon Augmented AI (Amazon A2I) à Amazon Rekognition [Exemple] dans une instance de bloc-notes AI. SageMaker

Exécution DetectModerationLabels avec Amazon A2I

> **Note**
>
> Créez toutes vos ressources Amazon A2I et Amazon Rekognition dans la même région AWS.

1. Remplissez les conditions requises répertoriées dans Getting Started with Amazon Augmented AI dans la documentation sur l'SageMaker IA.

   N'oubliez pas non plus de configurer vos autorisations IAM comme indiqué sur la page Autorisations et sécurité dans Amazon Augmented AI de la documentation sur l'SageMaker IA.

2. Suivez les instructions de création d'un flux de travail de révision humain dans la documentation de l'SageMaker IA.

   Un workflow de vérification humaine gère le traitement d'une image. Il contient les conditions qui déclenchent une vérification humaine, l'équipe de travail à laquelle l'image est envoyée, le modèle d'interface utilisateur utilisé par l'équipe de travail et le compartiment Amazon S3 vers lequel les résultats de l'équipe de travail sont envoyés.

   Dans le cadre de votre `CreateFlowDefinition` appel, vous devez `HumanLoopRequestSource` régler le « AWS/Rekognition/DetectModerationLabels/Image/V3 ». Après cela, vous devez décider comment vous souhaitez configurer vos conditions qui déclenchent la vérification humaine.

   Avec Amazon Rekognition, vous disposez de deux options pour `ConditionType` : `ModerationLabelConfidenceCheck`, et `Sampling`.

   `ModerationLabelConfidenceCheck` crée une boucle humaine lorsque la confiance d'une étiquette de modération se situe dans une plage. Enfin, `Sampling` envoie un pourcentage aléatoire des documents traités pour vérification humaine. Chaque `ConditionType` utilise un ensemble différent de `ConditionParameters` pour définir les résultats de la vérification humaine.

   `ModerationLabelConfidenceCheck` a `ConditionParameters` `ModerationLableName` qui définit la clé qui doit être vérifiée par les opérateurs. De plus, il a confiance, qui définit la fourchette de pourcentage pour l'envoi à une évaluation humaine avec LessThan, GreaterThan, et Equals. `Samplinga` `RandomSamplingPercentage` défini un pourcentage de documents qui seront soumis à un examen humain.

   L'exemple de code suivant est un appel partiel de `CreateFlowDefinition`. Il envoie une image pour vérification humaine si elle est évaluée à moins de 98% sur l'étiquette « Suggestif », et à plus de 95% sur l'étiquette « Maillots de bain ou sous-vêtements féminins ». Cela signifie que si l'image n'est pas considérée comme suggestive, mais comporte une femme en sous-vêtements ou maillot de bain, vous pouvez vérifier l'image en utilisant la vérification humaine.

```
def create_flow_definition():
'''
Creates a Flow Definition resource

Returns:
struct: FlowDefinitionArn
'''
humanLoopActivationConditions = json.dumps(
    {
        "Conditions": [
            {
              "And": [
                {
                    "ConditionType": "ModerationLabelConfidenceCheck",
                    "ConditionParameters": {
                        "ModerationLabelName": "Suggestive",
                        "ConfidenceLessThan": 98
                    }
                },
                {
                    "ConditionType": "ModerationLabelConfidenceCheck",
                    "ConditionParameters": {
                        "ModerationLabelName": "Female Swimwear Or Underwear",
                        "ConfidenceGreaterThan": 95
                    }
                }
              ]
           }
        ]
    }
)
```

`CreateFlowDefinition` renvoie un `FlowDefinitionArn`, que vous utilisez à l'étape suivante lorsque vous appelez `DetectModerationLabels`.

Pour plus d'informations, consultez CreateFlowDefinitionle Guide de référence des API d' SageMaker IA.

3. Définissez le paramètre `HumanLoopConfig` lorsque vous appelez `DetectModerationLabels`, comme dans Détection d'images inappropriées. Consultez l'étape 4 pour des exemples d'un appel `DetectModerationLabels` avec définition `HumanLoopConfig`.

   a. Dans le paramètre `HumanLoopConfig`, définissez `FlowDefinitionArn` sur l'ARN de la définition de flux que vous avez créée à l'étape 2.

   b. Définissez votre `HumanLoopName`. Cela doit être unique dans une région et en minuscules.

   c. (Facultatif) Vous pouvez utiliser `DataAttributes` pour définir si l'image que vous avez transmise à Amazon Rekognition est exempte ou non de données d'identification personnelle (PII). Vous devez définir ce paramètre pour envoyer des informations à Amazon Mechanical Turk.

4. Exécutez `DetectModerationLabels`.

   Les exemples suivants montrent comment utiliser le `HumanLoopConfig` set AWS CLI et AWS SDK pour Python (Boto3) pour l'exécuter `DetectModerationLabels` avec.

   AWS SDK pour Python (Boto3)

   L'exemple de demande suivant utilise le SDK for Python (Boto3). Pour plus d'informations, consultez detect_moderation_labels dans la Référence des API du kit AWS SDK pour 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

L'exemple de demande suivant utilise la AWS CLI. Pour plus d'informations, consultez detect-moderation-labels dans la Référence des commandes de l'.

```
$ 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"]}}'
```

Lorsque vous exécutez la `DetectModerationLabels` commande `HumanLoopConfig` Activé, Amazon Rekognition SageMaker appelle l'opération d'API AI. `StartHumanLoop` Cette commande prend la réponse auprès de `DetectModerationLabels` et la vérifie par rapport aux conditions de la définition de flux dans l'exemple. Si elle remplit les conditions d'examen, elle renvoie un `HumanLoopArn`. Cela signifie que les membres de l'équipe de travail que vous définissez dans votre définition de flux peuvent maintenant vérifier l'image. L'appel de l'opération d'exécution Amazon Augmented AI `DescribeHumanLoop` fournit des informations sur le résultat de la boucle. Pour plus d'informations, consultez DescribeHumanLoopla documentation de référence de l'API Amazon Augmented AI.

Une fois l'image révisée, vous pouvez voir les résultats dans le compartiment spécifié dans le chemin de sortie de votre définition de flux. Amazon A2I vous informera également par le biais d'Amazon CloudWatch Events lorsque la révision sera terminée. Pour connaître les événements à rechercher, consultez la section CloudWatch Événements de la documentation sur l'SageMaker IA.

Pour plus d'informations, consultez Getting Started with Amazon Augmented AI dans la documentation sur l'SageMaker IA.

Détection de texte

Amazon Rekognition peut détecter du texte dans les images et les vidéos. Elle peut ensuite convertir le texte détecté en texte lisible par une machine. Vous pouvez utiliser la détection de texte lisible par une machine dans les images pour implémenter des solutions telles que :

- Recherche visuelle. Par exemple, récupérer et afficher des images qui contiennent le même texte.
- Informations des contenus. Par exemple : informations sur des thèmes qui reviennent dans le texte reconnu à l'intérieur des images vidéo extraites. Votre application peut rechercher des contenus pertinents dans le texte reconnu, par exemple des actualités, des résultats sportifs, des numéros d'athlètes et des sous-titres.
- Navigation. Par exemple, le développement d'une application mobile vocale pour les personnes malvoyantes qui reconnaît les noms des restaurants et des magasins, ou les plaques de rue.
- Prise en charge de la sécurité et des transports publics. Par exemple, détection des numéros de plaques minéralogiques sur des images de caméras de surveillance de la circulation.
- Filtrage en cours. Par exemple, filtrage des données d'identification personnelle (PII) sur des images.

Pour la détection de texte dans les vidéos, vous pouvez implémenter des solutions telles que :

- Recherche dans des vidéos de clips avec des mots clés spécifiques, tels que le nom de l'invité sur un graphique d'une émission d'actualités.
- Modération du contenu pour garantir la conformité aux normes de l'organisation en détectant les textes inopinés, les grossièretés ou spams.
- Recherche de toutes les superpositions de texte sur la chronologie vidéo pour un traitement ultérieur, comme le remplacement par du texte dans une autre langue pour l'internationalisation du contenu.
- Recherche d'emplacements de texte pour que les autres graphiques puissent être alignés en conséquence.

Pour détecter du texte dans des images au format JPEG ou PNG, utilisez l'DetectTextopération. Pour détecter du texte de manière asynchrone dans une vidéo, utilisez les opérations StartTextDetectionet GetTextDetection. Les opérations de détection de texte d'image et de vidéo prennent en charge la plupart des polices, y compris les polices hautement stylisées. Après avoir détecté le texte, Amazon

Rekognition crée une représentation des mots et des lignes de texte détectés, affiche la relation entre eux et vous indique où le texte se trouve sur une image ou un cadre vidéo.

Les opérations `DetectText` et `GetTextDetection` détectent les mots et les lignes. Un mot est un ou plusieurs caractères de script qui ne sont pas séparés par des espaces. `DetectText` peut détecter jusqu'à 100 mots dans une image. `GetTextDetection` peut également détecter jusqu'à 100 mots par image de vidéo.

Un mot est constitué d'un ou de plusieurs caractères qui ne sont pas séparés par des espaces. Amazon Rekognition est conçu pour détecter les mots en anglais, arabe, russe, allemand, français, italien, portugais et espagnol.

Une ligne est une chaîne de mots séparés par des espaces égaux. Une ligne n'est pas nécessairement une phrase complète (les points n'indiquent pas la fin d'une ligne). Par exemple, Amazon Rekognition détecte un numéro de permis de conduire comme étant une ligne. Une ligne se termine lorsqu'aucun texte n'est aligné après elle, ou lorsqu'il y a un grand écart entre les mots par rapport à la longueur des mots. En fonction de l'intervalle entre les mots, Amazon Rekognition peut détecter plusieurs lignes dans un texte aligné dans la même direction. Si une phrase s'étend sur plusieurs lignes, l'opération renvoie plusieurs lignes.

Examinez l'image suivante.

Les cases bleues représentent des informations sur le texte détecté et son emplacement que l'opération `DetectText` renvoie. Dans cet exemple, Amazon Rekognition détecte « IT's », « MONDAY », « but », « keep », et « Smiling » comme étant des mots. Amazon Rekognition détecte les phrases « IT's », « MONDAY », « but », « keep », et « Smiling » sous forme de lignes. Pour être détecté, le texte doit avoir une orientation maximale de +/- 90 degrés par rapport à l'axe horizontal.

Pour obtenir un exemple, consultez Détection de texte sur une image.

Rubriques

- Détection de texte sur une image
- Détection de texte dans une vidéo stockée

Détection de texte sur une image

Vous pouvez fournir une image d'entrée sous la forme d'un tableau d'octets d'image (octets d'image encodés en base64) ou en tant qu'objet Amazon S3. Dans cette procédure, vous chargez une image JPEG ou PNG dans votre compartiment S3 et vous spécifiez le nom du fichier.

Pour détecter du texte sur une image (API)

1. Remplissez les conditions préalables requises suivantes, si vous ne l'avez pas déjà fait.

   a. Créez ou mettez à jour un utilisateur avec les autorisations `AmazonRekognitionFullAccess` et `AmazonS3ReadOnlyAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS Command Line Interface les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Chargez l'image qui contient du texte dans votre compartiment S3.

   Pour en savoir plus, consultez Chargement d'objets dans Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

3. Utilisez les exemples suivants pour appeler l'opération `DetectText`.

   Java

   L'exemple de code suivant affiche dans une image les lignes et les mots qui ont été détectés.

Remplacez la valeur de `amzn-s3-demo-bucket` et de `photo` par le nom du compartiment S3 et celui de l'image utilisés à l'étape 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

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
/**
*  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

Cette AWS CLI commande affiche la sortie JSON pour l'opération `detect-text` CLI.

Remplacez la valeur de `amzn-s3-demo-bucket` et de `Name` par le nom du compartiment S3 et le nom de l'image utilisés à l'étape 2.

Remplacez la valeur de `profile_name` par le nom de votre profil de développeur.

```
aws rekognition detect-text  --image "{"S3Object":{"Bucket":"amzn-s3-demo-
bucket","Name":"image-name"}}" --profile default
```

Si vous accédez à la CLI sur un périphérique Windows, utilisez des guillemets doubles au lieu de guillemets simples et évitez les guillemets doubles internes par une barre oblique inverse (c'est-à-dire \) pour corriger les erreurs d'analyse que vous pourriez rencontrer. Pour un exemple, consultez ce qui suit :

```
aws rekognition detect-text  --image "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --profile default
```

Python

L'exemple de code suivant affiche les lignes et les mots détectés dans une image.

Remplacez la valeur de `amzn-s3-demo-bucket` et de `photo` par le nom du compartiment S3; et le nom de l'image utilisés à l'étape 2. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

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

L'exemple de code suivant affiche les lignes et les mots détectés dans une image.

Remplacez la valeur de `amzn-s3-demo-bucket` et de `photo` par le nom du compartiment S3; et le nom de l'image utilisés à l'étape 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

L’exemple de code suivant affiche les lignes et les mots détectés dans une image.

Remplacez la valeur de `amzn-s3-demo-bucket` et de `photo` par le nom du compartiment S3; et le nom de l’image utilisés à l’étape 2. Remplacez la valeur de `region` par la région figurant dans vos informations d’identification .aws. Remplacez la valeur de `profile_name` dans la ligne qui crée la session de Rekognition par le nom de votre profil de développeur.

```
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 demande d'opération

Dans le cadre de l'opération `DetectText`, vous fournissez une image d'entrée sous la forme d'un tableau d'octets encodé en base64 ou en tant qu'image stockée dans un compartiment Amazon S3. L'exemple de demande JSON suivant présente l'image chargée à partir d'un compartiment Amazon S3.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "inputtext.jpg"
        }
    }
}
```

Filtres

Le filtrage par région, taille et score de fiabilité du texte vous offre encore plus de souplesse pour contrôler ce que donne la détection de votre texte. En utilisant les régions intéressantes, vous pouvez facilement limiter la détection de texte aux régions qui vous intéressent, par exemple en haut à droite de la photo de profil ou à un emplacement fixe par rapport à un point de référence lors de la lecture des numéros de pièces à partir d'une image d'une machine. Il est possible d'utiliser un filtre pour la taille du cadre de délimitation d'un mot, afin d'éviter l'apparition de texte de petite taille gênant ou non pertinent en arrière-plan. Le filtre de fiabilité des mots vous permet de supprimer les résultats non fiables car flous ou illisibles.

Pour plus d'informations sur les valeurs des filtres, consultez `DetectTextFilters`.

Vous pouvez utiliser les filtres suivants :

- MinConfidence: définit le niveau de confiance de la détection des mots. Les mots dont la confiance de détection est inférieure à ce niveau sont exclus du résultat. Les valeurs doivent être comprises entre 0 et 100.
- MinBoundingBoxWidth— Définit la largeur minimale du cadre de délimitation des mots. Les mots dont les zones de délimitation sont plus petites que cette valeur sont exclus du résultat. La valeur est relative à la largeur du cadre de l'image.

- MinBoundingBoxHeight— Définit la hauteur minimale du cadre de délimitation des mots. Les mots dont les hauteurs de zone de délimitation sont inférieures à cette valeur sont exclus du résultat. La valeur est relative à la hauteur du cadre de l'image.
- RegionsOfInterest— Limite la détection à une zone spécifique du cadre de l'image. Les valeurs sont relatives aux dimensions du cadre. Pour du texte se trouvant seulement partiellement dans une région, la réponse n'est pas définie.

DetectText réponse à l'opération

L'`DetectText`opération analyse l'image et renvoie un tableau TextDetections, où chaque élément (`TextDetection`) représente une ligne ou un mot détecté dans l'image. Pour chaque élément, `DetectText` renvoie les informations suivantes :

- Le texte détecté (`DetectedText`)
- Les relations entre les mots et les lignes (`Id` et `ParentId`)
- L'emplacement du texte sur l'image (`Geometry`)
- La confiance d'Amazon Rekognition dans l'exactitude du texte détecté et du cadre de délimitation (`Confidence`)
- Le type de texte détecté (`Type`)

Texte détecté

Chaque élément `TextDetection` contient le texte reconnu (mots ou lignes) dans le champ `DetectedText`. Un mot est constitué d'un ou plusieurs caractères non séparés par des espaces. `DetectText` peut détecter jusqu'à 100 mots dans une image. Le texte renvoyé peut inclure des caractères qui rendent un mot méconnaissable. Par exemple, Ch@t au lieu de Chat. Pour déterminer si un `TextDetection` élément représente une ligne de texte ou un mot, utilisez le champ `Type`.

Chaque élément `TextDetection` comprend une valeur de pourcentage qui représente le degré de fiabilité déterminé par Amazon Rekognition en ce qui concerne la précision du texte détecté et du cadre de délimitation qui entoure le texte.

Relations entre les mots et les lignes

Chaque élément `TextDetection` possède un champ d'identifiant, `Id`. L'`Id` montre la position du mot dans une ligne. Si l'élément est un mot, le champ de l'identifiant parent, `ParentId`, identifie la

ligne où le mot a été détecté. Le `ParentId` d'une ligne est null. Par exemple, la ligne « but keep » de l'exemple d'image a l'`Id` et les valeurs `ParentId` suivants :

Texte	ID	ID du parent
but keep	3	
but	8	3
keep	9	3

Emplacement du texte sur une image

Pour déterminer l'emplacement du texte reconnu dans une image, utilisez les informations du cadre de délimitation (Géometrie) renvoyées par `DetectText`. L'objet `Geometry` contient deux types d'informations sur les cadres de délimitation pour les lignes et les mots détectés :

- Un contour rectangulaire épais aligné sur l'axe dans un objet BoundingBox
- Un polygone plus fin composé de plusieurs coordonnées X et Y dans un tableau Point

Les coordonnées du cadre de sélection et du polygone montrent où le texte est situé sur l'image source. Les valeurs des coordonnées sont un ratio de la taille globale de l'image. Pour de plus amples informations, veuillez consulter BoundingBox.

La réponse JSON suivante de l'opération `DetectText` indique les mots et les lignes qui ont été détectés dans l'image suivante.

```
{
 'TextDetections': [{'Confidence': 99.35693359375,
                     'DetectedText': "IT'S",
                     'Geometry': {'BoundingBox': {'Height': 0.09988046437501907,
                                                  'Left': 0.6684935688972473,
                                                  'Top': 0.18226495385169983,
                                                  'Width': 0.1461552083492279},
                                  'Polygon': [{'X': 0.6684935688972473,
                                               'Y': 0.1838926374912262},
                                              {'X': 0.8141663074493408,
                                               'Y': 0.18226495385169983},
                                              {'X': 0.8146487474441528,
                                               'Y': 0.28051772713661194},
                                              {'X': 0.6689760088920593,
                                               'Y': 0.2821454107761383}]},
                     'Id': 0,
                     'Type': 'LINE'},
                    {'Confidence': 99.6207275390625,
                     'DetectedText': 'MONDAY',
                     'Geometry': {'BoundingBox': {'Height': 0.11442459374666214,
                                                  'Left': 0.5566731691360474,
```

```
                                        'Top': 0.3525116443634033,
                                        'Width': 0.39574965834617615},
                            'Polygon': [{'X': 0.5566731691360474,
                                         'Y': 0.353712260723114},
                                        {'X': 0.9522717595100403,
                                         'Y': 0.3525116443634033},
                                        {'X': 0.9524227976799011,
                                         'Y': 0.4657355844974518},
                                        {'X': 0.5568241477012634,
                                         'Y': 0.46693623065948486}]},
                 'Id': 1,
                 'Type': 'LINE'},
                {'Confidence': 99.6160888671875,
                 'DetectedText': 'but keep',
                 'Geometry': {'BoundingBox': {'Height': 0.08314694464206696,
                                        'Left': 0.6398131847381592,
                                        'Top': 0.5267938375473022,
                                        'Width': 0.2021435648202896},
                            'Polygon': [{'X': 0.640289306640625,
                                         'Y': 0.5267938375473022},
                                        {'X': 0.8419567942619324,
                                         'Y': 0.5295097827911377},
                                        {'X': 0.8414806723594666,
                                         'Y': 0.609940767288208},
                                        {'X': 0.6398131847381592,
                                         'Y': 0.6072247624397278}]},
                 'Id': 2,
                 'Type': 'LINE'},
                {'Confidence': 88.95134735107422,
                 'DetectedText': 'Smiling',
                 'Geometry': {'BoundingBox': {'Height': 0.4326171875,
                                        'Left': 0.46289217472076416,
                                        'Top': 0.5634765625,
                                        'Width': 0.5371078252792358},
                            'Polygon': [{'X': 0.46289217472076416,
                                         'Y': 0.5634765625},
                                        {'X': 1.0, 'Y': 0.5634765625},
                                        {'X': 1.0, 'Y': 0.99609375},
                                        {'X': 0.46289217472076416,
                                         'Y': 0.99609375}]},
                 'Id': 3,
                 'Type': 'LINE'},
                {'Confidence': 99.35693359375,
                 'DetectedText': "IT'S",
```

```
                    'Geometry': {'BoundingBox': {'Height': 0.09988046437501907,
                                                 'Left': 0.6684935688972473,
                                                 'Top': 0.18226495385169983,
                                                 'Width': 0.1461552083492279},
                                 'Polygon': [{'X': 0.6684935688972473,
                                              'Y': 0.1838926374912262},
                                             {'X': 0.8141663074493408,
                                              'Y': 0.18226495385169983},
                                             {'X': 0.8146487474441528,
                                              'Y': 0.28051772713661194},
                                             {'X': 0.6689760088920593,
                                              'Y': 0.2821454107761383}]},
                    'Id': 4,
                    'ParentId': 0,
                    'Type': 'WORD'},
                   {'Confidence': 99.6207275390625,
                    'DetectedText': 'MONDAY',
                    'Geometry': {'BoundingBox': {'Height': 0.11442466825246811,
                                                 'Left': 0.5566731691360474,
                                                 'Top': 0.35251158475875854,
                                                 'Width': 0.39574965834617615},
                                 'Polygon': [{'X': 0.5566731691360474,
                                              'Y': 0.3537122905254364},
                                             {'X': 0.9522718787193298,
                                              'Y': 0.35251158475875854},
                                             {'X': 0.9524227976799011,
                                              'Y': 0.4657355546951294},
                                             {'X': 0.5568241477012634,
                                              'Y': 0.46693626046180725}]},
                    'Id': 5,
                    'ParentId': 1,
                    'Type': 'WORD'},
                   {'Confidence': 99.96778869628906,
                    'DetectedText': 'but',
                    'Geometry': {'BoundingBox': {'Height': 0.0625,
                                                 'Left': 0.6402802467346191,
                                                 'Top': 0.5283203125,
                                                 'Width': 0.08027780801057816},
                                 'Polygon': [{'X': 0.6402802467346191,
                                              'Y': 0.5283203125},
                                             {'X': 0.7205580472946167,
                                              'Y': 0.5283203125},
                                             {'X': 0.7205580472946167,
                                              'Y': 0.5908203125},
```

```
                                        {'X': 0.6402802467346191,
                                         'Y': 0.5908203125}]},
                     'Id': 6,
                     'ParentId': 2,
                     'Type': 'WORD'},
                    {'Confidence': 99.26438903808594,
                     'DetectedText': 'keep',
                     'Geometry': {'BoundingBox': {'Height': 0.0818721204996109,
                                                  'Left': 0.7344760298728943,
                                                  'Top': 0.5280686020851135,
                                                  'Width': 0.10748066753149033},
                                  'Polygon': [{'X': 0.7349520921707153,
                                               'Y': 0.5280686020851135},
                                              {'X': 0.8419566750526428,
                                               'Y': 0.5295097827911377},
                                              {'X': 0.8414806127548218,
                                               'Y': 0.6099407076835632},
                                              {'X': 0.7344760298728943,
                                               'Y': 0.6084995269775391}]},
                     'Id': 7,
                     'ParentId': 2,
                     'Type': 'WORD'},
                    {'Confidence': 88.95134735107422,
                     'DetectedText': 'Smiling',
                     'Geometry': {'BoundingBox': {'Height': 0.4326171875,
                                                  'Left': 0.46289217472076416,
                                                  'Top': 0.5634765625,
                                                  'Width': 0.5371078252792358},
                                  'Polygon': [{'X': 0.46289217472076416,
                                               'Y': 0.5634765625},
                                              {'X': 1.0, 'Y': 0.5634765625},
                                              {'X': 1.0, 'Y': 0.99609375},
                                              {'X': 0.46289217472076416,
                                               'Y': 0.99609375}]},
                     'Id': 8,
                     'ParentId': 3,
                     'Type': 'WORD'}],
 'TextModelVersion': '3.0'}
```

Détection de texte dans une vidéo stockée

La détection de textes par Vidéo Amazon Rekognition dans les vidéos stockées est une opération asynchrone. Pour commencer à détecter le texte, appelez StartTextDetection. Vidéo Amazon

Rekognition publie le statut d'achèvement de l'analyse des vidéos dans une rubrique Amazon SNS. Si l'analyse vidéo aboutit, appelez GetTextDetection pour obtenir les résultats de cette opération. Pour plus d'informations sur le démarrage de l'analyse vidéo et l'obtention des résultats, consultez Appeler les opérations de Vidéo Amazon Rekognition.

Cette procédure se développe sur le code dans Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK). Il utilise une file d'attente Amazon SQS pour obtenir l'état d'achèvement d'une demande d'analyse vidéo.

Pour détecter des visages dans une vidéo stockée dans un compartiment Amazon S3 (SDK)

1. Effectuez les étapes décrites dans la section Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK).
2. Ajoutez le code suivant à la classe `VideoDetect` de l'étape 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);


}
```

Dans la fonction `main`, remplacez les lignes :

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

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

avec :

```
        StartTextDetection(amzn-s3-demo-bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetTextDetectionResults();
```

Java V2

Ce code est extrait du GitHub référentiel d'exemples du SDK de AWS documentation. Voir l'exemple complet ici.

```
//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
```

Dans la fonction `main`, remplacez les lignes :

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

avec :

```
    analyzer.StartTextDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetTextDetectionResults()
```

CLI

Exécutez la AWS CLI commande suivante pour commencer à détecter du texte dans une vidéo.

```
 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
```

Mettez à jour les valeurs suivantes :

- Remplacez `amzn-s3-demo-bucket` et `video-name` par le nom du compartiment Amazon S3 et le nom du fichier que vous avez spécifiés à l'étape 2.
- Remplacez `region-name` par la région AWS que vous utilisez.
- Remplacez la valeur de `profile-name` par le nom de votre profil de développeur.
- Remplacez `topic-ARN` par l'ARN de la rubrique Amazon SNS que vous avez créée à l'étape 3 de Configuration de Vidéo Amazon Rekognition.
- Remplacez `role-ARN` par l'ARN de la fonction de service IAM que vous avez créé à l'étape 7 de Configuration de Vidéo Amazon Rekognition.

Si vous accédez à la CLI sur un périphérique Windows, utilisez des guillemets doubles au lieu de guillemets simples et évitez les guillemets doubles internes par une barre oblique inverse (c'est-à-dire \) pour corriger les erreurs d'analyse que vous pourriez rencontrer. Pour un exemple, voir ci-dessous :

```
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
```

Après avoir exécuté l'exemple de code suivant, copiez l'`jobID` renvoyé et saisissez-le dans la commande suivante `GetTextDetection` pour obtenir vos résultats, en remplaçant `job-id-number` par `jobID` que vous avez reçu précédemment :

```
aws rekognition get-text-detection --job-id job-id-number --profile profile-name
```

> **Note**
>
> Si vous avez déjà exécuté un exemple vidéo autre que Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK), le code à remplacer peut être différent.

3. Exécutez le code. Le texte détecté dans la vidéo est affiché dans une liste.

Filtres

Les filtres sont des paramètres de requête facultatifs qui peuvent être utilisés lorsque vous appelez `StartTextDetection`. Le filtrage par région, taille et score de fiabilité du texte vous offre encore plus de souplesse pour contrôler ce que donne la détection de votre texte. Grâce aux régions d'intérêt, vous pouvez aisément délimiter la détection de texte aux seules régions pertinentes, comme une région tierce pour les graphiques ou un panneau dans le coin supérieur gauche indiquant le score pour un match de foot, par exemple. Il est possible d'utiliser un filtre pour la taille du cadre de délimitation d'un mot, afin d'éviter l'apparition de texte de petite taille gênant ou non pertinent en

arrière-plan. Pour finir, le filtre de fiabilité des mots vous permet de supprimer les résultats non fiables car flous ou illisibles.

Pour plus d'informations sur les valeurs des filtres, consultez `DetectTextFilters`.

Vous pouvez utiliser les filtres suivants :

- MinConfidence: définit le niveau de confiance de la détection des mots. Les mots dont la confiance de détection est inférieure à ce niveau sont exclus du résultat. Les valeurs doivent être comprises entre 0 et 100.
- MinBoundingBoxWidth— Définit la largeur minimale du cadre de délimitation des mots. Les mots dont les zones de délimitation sont plus petites que cette valeur sont exclus du résultat. La valeur est relative à la largeur de l'image vidéo.
- MinBoundingBoxHeight— Définit la hauteur minimale du cadre de délimitation des mots. Les mots dont les hauteurs de zone de délimitation sont inférieures à cette valeur sont exclus du résultat. La valeur est relative à la hauteur de l'image vidéo.
- RegionsOfInterest— Limite la détection à une zone spécifique du cadre. Les valeurs sont relatives aux dimensions du cadre. Pour les objets partiellement situés dans les régions, la réponse n'est pas définie.

GetTextDetection réponse

`GetTextDetection` renvoie un tableau (`TextDetectionResults`) qui contient des informations sur le texte détecté dans la vidéo. Un élément de liste, TextDetection, existe pour chaque occurrence de détection d'un mot ou d'une ligne dans la vidéo. Les éléments de la liste sont triés par heure, en millisecondes depuis le début de la vidéo.

Voici une réponse JSON partielle renvoyée par `GetTextDetection`. Dans la réponse, notez les points suivants :

- Informations textuelles — L'élément du `TextDetectionResult` tableau contient des informations sur le texte détecté (TextDetection) et l'heure à laquelle le texte a été détecté dans la vidéo (`Timestamp`).
- Informations sur la pagination : l'exemple montre une page d'informations sur la détection de texte. Vous pouvez spécifier le nombre d'éléments de texte à renvoyer dans le paramètre d'entrée `MaxResults` pour `GetTextDetection`. Si le nombre de résultats est supérieur à `MaxResults`, ou s'il y a plus de résultats que le maximum par défaut, `GetTextDetection` renvoie un jeton

(`NextToken`) qui est utilisé pour obtenir la page suivante de résultats. Pour de plus amples informations, veuillez consulter Obtenir les résultats de l'analyse de Vidéo Amazon Rekognition.

- Informations sur la vidéo : la réponse comprend des informations sur le format vidéo (`VideoMetadata`) dans chaque page d'informations renvoyée par `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"
}
```

Détection de segments vidéo dans une vidéo stockée

Vidéo Amazon Rekognition fournit une API qui identifie les segments utiles de la vidéo, tels que les images noires et le générique de fin.

Les utilisateurs n’ont jamais regardé autant de contenus. En particulier, les plateformes Over-The-Top (OTT) et Video-On-Demand (VOD) offrent une vaste sélection de choix de contenus à tout moment, n'importe où et sur n'importe quel écran. Avec la prolifération des volumes de contenus, les entreprises de multimédia sont confrontées à des défis dans la préparation et la gestion de leurs contenus. Ceci est crucial pour offrir une expérience de visualisation de haute qualité et une meilleure monétisation du contenu. Aujourd’hui, les entreprises font appel à de grandes équipes de main-d’œuvre qualifiée, par exemple pour accomplir les tâches suivantes.

- Trouver où sont les génériques de début et de fin dans un élément de contenu
- Choisir les bons endroits pour insérer des publicités, par exemple dans des séquences d’images noires silencieuses
- Décomposer les vidéos en petits clips pour une meilleure indexation

Ces processus manuels sont coûteux, lents et incapables de s’adapter au volume de contenus produits, accrédités et récupérés quotidiennement dans les archives.

Vous pouvez utiliser Vidéo Amazon Rekognition pour automatiser les tâches opérationnelles d’analyse multimédia à l’aide d’API de détection de segments vidéo entièrement gérées et spécialement conçues, basées sur le machine learning (ML). En utilisant les API de segmentation Vidéo Amazon Rekognition, vous pouvez facilement analyser de grands volumes de vidéos et détecter des marqueurs tels que des images noires ou des changements de plans. Pour chaque détection, vous obtenez des codes temporels, des horodatages et des numéros de trames SMPTE (Society of Motion Picture and Television Engineers). Aucune expérience de machine learning n’est requise.

Vidéo Amazon Rekognition analyse les vidéos stockées dans un compartiment Amazon Simple Storage Service (Amazon S3). Les codes temporels SMPTE renvoyés sont précis à l’image près. Vidéo Amazon Rekognition fournit le numéro de trame exact d’un segment vidéo détecté et gère divers formats de fréquence d’images vidéo. Vous pouvez utiliser les métadonnées précises de Vidéo Amazon Rekognition pour automatiser complètement certaines tâches ou réduire considérablement la charge de travail de révision des opérateurs humains formés, afin qu’ils puissent se concentrer sur les tâches plus créatives. Ceci vous permet d’effectuer des tâches telles que la préparation des

contenus, l'insertion de publicités et l'ajout de « marqueurs en rafale » dans les contenus à grande échelle dans le cloud.

Pour en savoir plus sur la tarification, veuillez consulter Tarification Amazon Rekognition.

La détection de segments Vidéo Amazon Rekognition prend en charge deux types de tâches de segmentation : la détection Repères techniques et Détection des plans.

Rubriques

- Repères techniques
- Détection des plans
- À propos de l'API de détection des segments Vidéo Amazon Rekognition
- Utilisation de l'API de segmentation Amazon Rekognition
- Exemple : Détection de segments dans une vidéo stockée

Repères techniques

Un repère technique permet d'identifier les images noires, les barres de couleur, les génériques de début et de fin, les logos des studios et le contenu principal du programme d'une vidéo.

Images noires

Les vidéos contiennent souvent des images noires vides, sans aucun son, pour délimiter les emplacements d'insertion des publicités ou la fin d'un segment de programme comme une scène ou le générique de début. Avec Vidéo Amazon Rekognition, vous pouvez détecter les séquences d'images noires pour automatiser l'insertion de publicités, conditionner le contenu pour la vidéo à la demande et délimiter divers segments ou scènes de programme. Les images noires avec audio (tels que les fondus ou les voix off) sont considérées comme du contenu et ne sont pas restituées.

Génériques

Vidéo Amazon Rekognition vous permet d'identifier automatiquement les images exactes où commence et se termine le générique de clôture pour un film ou une série TV. Grâce à ces informations, vous pouvez générer des « marqueurs en rafale » ou des instructions interactives, telles que « Épisode suivant » ou « Sauter l'intro », dans les applications de vidéo à la demande (VOD). Vous pouvez également détecter la première et la dernière image du contenu d'un programme

dans une vidéo. Vidéo Amazon Rekognition est formé pour gérer une grande variété de styles de génériques de début et de fin, allant de simples génériques progressifs à des génériques plus complexes associés au contenu.

Barres de couleur

Vidéo Amazon Rekognition vous permet de détecter les sections de vidéo qui affichent des barres de couleurs SMPTE, à savoir un ensemble de couleurs affichées selon des motifs spécifiques pour s'assurer que la couleur est correctement calibrée sur les moniteurs de diffusion, les programmes et les caméras. Pour de plus amples informations sur les barres de couleur SMPTE, veuillez consulter Barre de couleurs SMPTE. Ces métadonnées sont utiles pour préparer les contenus pour les applications de vidéo à la demande en supprimant les segments de barres de couleur du contenu, ou pour détecter des problèmes tels que la perte de signaux de diffusion dans un enregistrement, lorsque les barres de couleur sont affichées en continu comme signal par défaut au lieu du contenu.

Ardoises

Les ardoises sont des sections de la vidéo, généralement situées au début, qui contiennent des métadonnées textuelles sur l'épisode, le studio, le format vidéo, les chaînes audio, etc. Vidéo Amazon Rekognition peut identifier le début et la fin des listes, ce qui facilite l'utilisation des métadonnées du texte ou la suppression de l'ardoise lors de la préparation du contenu pour le visionnage final.

Logos de studio

Les logos de studio sont des séquences qui montrent les logos ou les emblèmes du studio de production impliqué dans la création de l'émission. Vidéo Amazon Rekognition peut détecter ces séquences afin que les utilisateurs puissent les consulter afin d'identifier les studios.

Contenu

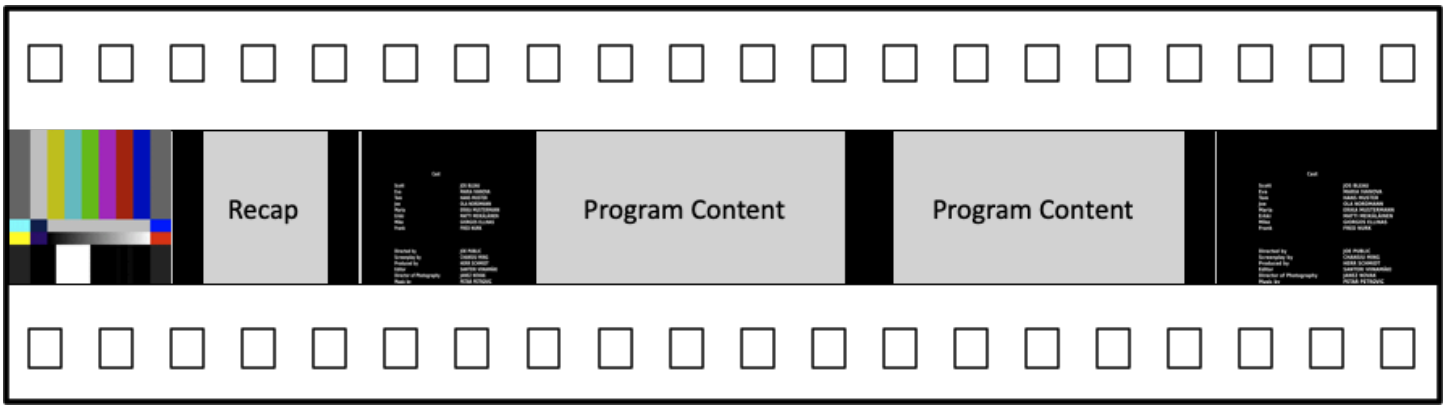
Le contenu correspond aux parties de l'émission de télévision ou du film qui contiennent le programme ou des éléments connexes. Les images noires, les génériques, les barres de couleur, les ardoises et les logos de studio ne sont pas considérés comme du contenu. Vidéo Amazon Rekognition peut détecter le début et la fin de chaque segment de contenu de la vidéo, afin que vous puissiez connaître la durée d'exécution du programme ou des segments spécifiques.

Ces segments de contenu comprennent les attributs suivants, sans s'y limiter :

- Scènes de programmation entre deux pauses publicitaires
- Un bref résumé de l'épisode précédent au début de la vidéo
- Contenu bonus post-générique
- Contenu « sans texte », tel qu'un ensemble de scènes de programme contenant à l'origine du texte superposé, mais dont le texte a été supprimé pour permettre la traduction dans d'autres langues.

Une fois que Vidéo Amazon Rekognition a fini de détecter tous les segments de contenu, vous pouvez appliquer les connaissances du domaine ou les envoyer pour examen par un humain afin de mieux classer chaque segment. Par exemple, si vous utilisez des vidéos qui commencent toujours par un récapitulatif, vous pouvez classer le premier segment de contenu dans la catégorie récapitulatif.

Le diagramme suivant montre les segments de repères techniques sur la chronologie d'une émission ou d'un film. Notez les barres de couleur et le générique d'ouverture, les segments de contenu tels que le récapitulatif et le programme principal, les images noires de la vidéo et le générique de fin.



Détection des plans

Un plan est une série d'images consécutives et interdépendantes, prises de façon contiguë par une seule caméra et représentant une action continue dans le temps et l'espace. Avec Vidéo Amazon Rekognition, vous pouvez détecter le début, la fin et la durée de chaque plan, ainsi que le nombre total de plans d'un contenu. Vous pouvez utiliser les métadonnées des plans par exemple pour les tâches suivantes.

- Création de vidéos promotionnelles à l'aide de plans sélectionnés.
- Insertion de publicités à des endroits qui ne perturbent pas l'expérience du spectateur, par exemple au milieu d'un plan lorsque quelqu'un parle.

- Génération d'un ensemble de miniatures d'aperçu qui évitent le contenu de transition entre les plans.

Une détection de plans est marquée à l'image exacte où il y a un passage à une autre caméra. S'il y a une transition douce d'une caméra à l'autre, Vidéo Amazon Rekognition omet la transition. Ceci garantit que les débuts et fins de plan n'incluent pas de sections sans contenu réel.

Le diagramme suivant illustre les segments de détection de plans sur une piste de film. Notez que chaque plan est identifié par une coupure d'un angle de caméra ou d'un emplacement vers un autre.

À propos de l'API de détection des segments Vidéo Amazon Rekognition

Pour segmenter une vidéo stockée, vous utilisez les opérations asynchrones StartSegmentDetectionet d'GetSegmentDetectionAPI pour démarrer une tâche de segmentation et récupérer les résultats. La détection de segments accepte les vidéos stockées dans un compartiment Amazon S3 et renvoie une sortie JSON. Vous pouvez choisir de détecter uniquement les repères techniques, uniquement les changements de plans, ou les deux ensemble en configurant la demande d'API `StartSegmentdetection`. Vous pouvez également filtrer les segments détectés en définissant des seuils pour une confiance de prédiction minimale. Pour de plus amples informations, veuillez consulter Utilisation de l'API de segmentation Amazon Rekognition. Pour obtenir un exemple de code, consultez Exemple : Détection de segments dans une vidéo stockée.

Utilisation de l'API de segmentation Amazon Rekognition

La détection de segments Vidéo Amazon Rekognition dans les vidéos stockées est une opération asynchrone de Vidéo Amazon Rekognition. L'API de segmentation Amazon Rekognition est une API

composite dans laquelle vous choisissez le type d'analyse (repères techniques ou détection de plans) à partir d'un seul appel d'API. Pour de plus amples informations sur l'appel d'opérations asynchrones, veuillez consulter Appeler les opérations de Vidéo Amazon Rekognition.

Rubriques

- Démarrage de l'analyse de segments
- Obtention des résultats de l'analyse des segments

Démarrage de l'analyse de segments

Pour démarrer la détection de segments dans un appel vidéo stocké StartSegmentDetection. Les paramètres d'entrée sont les mêmes que pour les autres opérations Vidéo Amazon Rekognition avec l'ajout de la sélection du type de segment et du filtrage des résultats. Pour de plus amples informations, veuillez consulter Démarrage d'une analyse vidéo.

Voici un exemple de JSON transmis par `StartSegmentDetection`. La demande spécifie que les segments de repères techniques et les segments de détection de plans sont tous détectés. Différents filtres pour la confiance de détection minimale sont demandés pour les segments de repères techniques (90 %) et les segments de détection de plans (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
      }
  }
}
```

Choix d'un type de segment

Utilisez le paramètre d'entrée de la `SegmentTypes` matrice pour détecter les segments techniques de détection des and/or coups de repère dans la vidéo d'entrée.

- TECHNICAL_CUE : identifie les horodatages précis à l'image près pour le début, la fin et la durée des repères techniques (images noires, barres de couleur, génériques de début et de fin, logos studio et contenu de programme primaire) détectés dans une vidéo. Par exemple, vous pouvez utiliser des repères techniques pour trouver le début du générique de fin. Pour de plus amples informations, veuillez consulter Repères techniques.
- SHOT : identifie le début, la fin et la durée d'un plan. Par exemple, vous pouvez utiliser la détection de plans afin d'identifier les plans candidats pour un montage final d'une vidéo. Pour de plus amples informations, veuillez consulter Détection des plans.

Filtrage des résultats de l'analyse

Vous pouvez utiliser le paramètre d'entrée `Filters` (StartSegmentDetectionFilters) pour spécifier la confiance minimale de détection renvoyée dans la réponse. Dans `Filters`, utilisez `ShotFilter` (StartShotDetectionFilter) pour filtrer les plans détectés. Utilisez `TechnicalCueFilter` (StartTechnicalCueDetectionFilter) pour filtrer les informations techniques.

Pour obtenir un exemple de code, consultez Exemple : Détection de segments dans une vidéo stockée.

Obtention des résultats de l'analyse des segments

Vidéo Amazon Rekognition publie l'état d'achèvement de l'opération d'analyse vidéo dans une rubrique Amazon Simple Notification Service. Si l'analyse vidéo est réussie, appelez GetSegmentDetectionpour obtenir les résultats de l'analyse vidéo.

Vous trouverez ci-après un exemple de demande `GetSegmentDetection`. Le `JobId` est l'identifiant de tâche renvoyé par l'appel à `StartSegmentDetection`. Pour de plus amples informations sur les autres paramètres d'entrée, veuillez consulter Obtenir les résultats de l'analyse de Vidéo Amazon Rekognition.

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

```
    "NextToken": "XfXnZKiyMOGDhzBzYUhS5puM+g1IgezqFeYpv/H/+5noP/LmM57FitUAwSQ5D6G4AB/
PNwolrw=="
}
```

`GetSegmentDetection` renvoie les résultats de l'analyse demandée et des informations générales sur la vidéo stockée.

Informations générales

`GetSegmentDection` renvoie les informations générales suivantes.

- Informations audio — La réponse inclut des métadonnées audio dans un tableau d'AudioMetadataobjets. `AudioMetadata` Il peut y avoir plusieurs flux audio. Chaque objet `AudioMetadata` contient des métadonnées pour un flux audio unique. Les informations audio d'un objet `AudioMetadata` incluent le codec audio, le nombre de canaux audio, la durée du flux audio et la fréquence d'échantillonnage. Les métadonnées audio sont renvoyées dans chaque page d'informations renvoyée par `GetSegmentDetection`.
- Informations sur la vidéo — Actuellement, Amazon Rekognition Video renvoie un VideoMetadataseul objet dans le tableau. `VideoMetadata` L'objet contient des informations sur le flux vidéo dans le fichier d'entrée que Vidéo Amazon Rekognition a choisi d'analyser. L'objet `VideoMetadata` inclut le codec vidéo, le format vidéo et d'autres informations. Les métadonnées vidéo sont renvoyées dans chaque page d'informations renvoyée par `GetSegmentDetection`.
- Informations sur la pagination : l'exemple montre une page d'informations de segments. Vous pouvez spécifier le nombre d'éléments à renvoyer dans le paramètre d'entrée `MaxResults` pour `GetSegmentDetection`. Si le nombre de résultats est supérieur à `MaxResults`, `GetSegmentDetection` renvoie un jeton (`NextToken`) utilisé pour obtenir la page de résultats suivante. Pour de plus amples informations, veuillez consulter Obtenir les résultats de l'analyse de Vidéo Amazon Rekognition.
- Informations sur la demande : le type d'analyse demandée dans l'appel à `StartSegmentDetection` est renvoyé dans le champ `SelectedSegmentTypes`.

Segments

Les repères techniques et les informations sur les plans détectés dans une vidéo sont renvoyés dans un tableau, `Segments`, d'objets SegmentDetection. Le tableau est trié en fonction des types de segments (TECHNICAL_CUE ou SHOT) spécifiés dans le paramètre d'entrée `SegmentTypes` de `StartSegmentDetection`. Dans chaque type de segment, le tableau est trié par valeurs

d'horodatage. Chaque objet `SegmentDetection` inclut des informations sur le type de segment détecté (repère technique ou détection de plans) et des informations générales, telles que l'heure de début, l'heure de fin et la durée du segment.

Les informations temporelles sont renvoyées dans trois formats.

- Millisecondes

  Nombre de millisecondes depuis le début de la vidéo. Les champs `DurationMillis`, `StartTimestampMillis` et `EndTimestampMillis` sont présentés en millisecondes.

- Code temporel

  Les codes temporels Vidéo Amazon Rekognition sont au format SMPTE, où chaque image de la vidéo a une valeur de code temporel unique. Le format est hh:mm:ss:frame. Par exemple, une valeur de code temporel de 01:05:40:07 serait lue comme une heure, cinq minutes, quarante secondes et sept images. Les cas d'utilisation de Drop frame sont pris en charge par Vidéo Amazon Rekognition. Le format de code temporel drop rate est hh:mm:ss;frame. Les champs `DurationSMPTE`, `StartTimecodeSMPTE` et `EndTimecodeSMPTE` sont au format de code temporel.

- Compteurs à cadres

  La durée de chaque segment vidéo est également exprimée par le nombre d'images. Le champ `StartFrameNumber` indique le numéro d'image au début d'un segment vidéo et `EndFrameNumber` donne le numéro d'image à la fin d'un segment vidéo. `DurationFrames` indique le nombre total d'images d'un segment vidéo. Ces valeurs sont calculées à l'aide d'un indice de trame commençant par 0.

Vous pouvez utiliser le champ `SegmentType` pour déterminer le type d'un segment renvoyé par Vidéo Amazon Rekognition.

- Indices techniques : le `TechnicalCueSegment` champ est un TechnicalCueSegmentobjet qui contient la fiabilité de détection et le type d'un indice technique. Les types de repères techniques sont `ColorBars`, `EndCredits`, `BlackFrames`, `OpeningCredits`, `StudioLogo`, `Slate`, et `Content`.
- Plan : le `ShotSegment` champ est un ShotSegmentobjet qui contient le niveau de confiance de détection et un identifiant pour le segment de prise de vue dans la vidéo.

L'exemple suivant constitue la réponse JSON de `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
        }
    ]
}
```

Pour obtenir un exemple de code, consultez Exemple : Détection de segments dans une vidéo stockée.

Exemple : Détection de segments dans une vidéo stockée

La procédure suivante montre comment détecter des segments de repères techniques et des segments de détection de plans dans une vidéo stockée dans un compartiment Amazon S3. La procédure montre également comment filtrer les segments détectés en fonction de la confiance de Vidéo Amazon Rekognition dans la précision de la détection.

L'exemple s'appuie sur le code figurant dans Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK), qui utilise une file d'attente Amazon Simple Queue Service pour obtenir le statut d'achèvement d'une demande d'analyse vidéo.

Pour détecter des segments dans une vidéo stockée dans un compartiment Amazon S3 (SDK)

1. Effectuez une Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK).
2. Ajoutez ce qui suit au code que vous avez utilisé à l'étape 1.

   Java

   1. Ajoutez les importations suivantes.

      ```
      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. Ajoutez le code suivant à la 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. Dans la fonction `main`, remplacez les lignes :

```
       StartLabelDetection(amzn-s3-demo-bucket, video);

       if (GetSQSMessageSuccess()==true)
```

```
         GetLabelDetectionResults();
```

avec :

```
        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. Ajoutez le code suivant à la classe `VideoDetect` que vous avez créée à l'étape 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. Dans la fonction `main`, remplacez les lignes :

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

avec :

```
analyzer.StartSegmentDetection()
if analyzer.GetSQSMessageSuccess()==True:
    analyzer.GetSegmentDetectionResults()
```

> **Note**
>
> Si vous avez déjà exécuté un exemple vidéo autre que Analyse d'une vidéo stockée dans un compartiment Amazon S3 avec Java or Python (SDK), le code à remplacer peut être différent.

3. Exécutez le code. Les informations sur les segments détectés dans la vidéo d'entrée sont affichées.

Détection de la vivacité faciale

Amazon Rekognition Face Liveness vous aide à vérifier qu'un utilisateur soumis à une vérification faciale est physiquement présent devant une caméra. Cette fonctionnalité détecte également les attaques frauduleuses présentées à une caméra ou qui tentent de contourner une caméra. Les utilisateurs peuvent effectuer un test Face Liveness en prenant un court selfie vidéo dans lequel ils suivent une série d'instructions destinées à vérifier leur présence.

La vivacité faciale d'un visage est déterminée à l'aide d'un calcul probabiliste, puis un score de confiance (compris entre 0 et 100) est renvoyé après le contrôle. Plus le score est élevé, plus on est sûr que la personne qui reçoit le chèque est réelle. Face Liveness renvoie également un cadre, appelé image de référence, qui peut être utilisé pour la comparaison et la recherche de visages. Comme tout système basé sur les probabilités, Face Liveness ne peut garantir des résultats parfaits. Utilisez-le avec d'autres facteurs pour prendre une décision fondée sur le risque concernant l'identité personnelle des utilisateurs.

Face Liveness utilise plusieurs composants :

- AWS Amplify SDK (React, Swift (iOS) et Android) avec composant FaceLivenessDetector
- AWS SDK
- AWS API cloud

Lorsque vous configurez votre application pour qu'elle s'intègre à la fonctionnalité Face Liveness, elle utilise les opérations d'API suivantes :

- CreateFaceLivenessSession- Démarre une session Face Liveness, permettant d'utiliser le modèle de détection Face Liveness dans votre application. Renvoie une valeur SessionId pour la session créée. Vous permet également de définir votre ChallengePrefrence, afin que vous puissiez utiliser l' FaceMovementChallenge option.
- StartFaceLivenessSession- Appelé par l' AWS Amplify FaceLivenessDetector. Démarre un flux d'événements contenant des informations sur les événements et les attributs pertinents de la session en cours.
- GetFaceLivenessSessionResults- Récupère les résultats d'une session Face Liveness spécifique, y compris un score de confiance Face Liveness, une image de référence et des images d'audit.

Vous utiliserez le SDK AWS Amplify pour intégrer la fonctionnalité Face Liveness à vos flux de travail de vérification basés sur le visage pour les applications Web. Lorsque les utilisateurs intègrent ou s'authentifient via votre application, envoyez-les vers le flux de travail de vérification de Face Liveness dans le SDK Amplify. Le SDK Amplify gère l'interface utilisateur et les commentaires en temps réel des utilisateurs lorsqu'ils capturent leur selfie vidéo.

Lorsque vous utilisez FaceMovementAndLightChallenge le visage de l'utilisateur qui se déplace dans l'ovale affiché sur son appareil, le SDK Amplify affiche une séquence de lumières colorées à l'écran. Il diffuse ensuite en toute sécurité la vidéo du selfie vers les API cloud. Sinon, lors de l'utilisation FaceMovementChallenge, le visage de l'utilisateur se déplace dans l'ovale affiché sur son appareil, mais il n'y a aucune séquence de lumières colorées. Bien que « FaceMovementAndLightChallenge » reste le meilleur réglage pour optimiser la précision, « FaceMovementChallenge » permet aux clients de donner la priorité à la rapidité des contrôles de vivacité plutôt qu'à la précision. Lorsqu'ils choisissent l'un de ces paramètres, les clients doivent prendre en compte les exigences de leur cas d'utilisation, notamment les types d'attaques attendus, les taux de fausses acceptations et de faux rejets souhaités, ainsi que des contrôles supplémentaires tels que la géolocalisation (par exemple en fonction de l'adresse IP), les codes One Time Pass (OTP), etc. Les clients doivent prendre cette décision après avoir testé les performances de Liveness avec différents seuils de confiance pour leur contenu spécifique. En outre, avec les deux types de contrôle de vivacité, les propriétaires de l'application doivent mettre en œuvre des contrôles pour sécuriser l'appareil à partir duquel le flux vidéo est envoyé. Une fois l'analyse terminée, vous recevez les informations suivantes sur le backend :

- Un score de confiance de Face Liveness (entre 0 et 100)
- Une image de haute qualité appelée image de référence qui peut être utilisée pour Face Match ou Face Search
- Un ensemble de quatre images maximum, appelées images d'audit, sélectionnées à partir de la vidéo du selfie

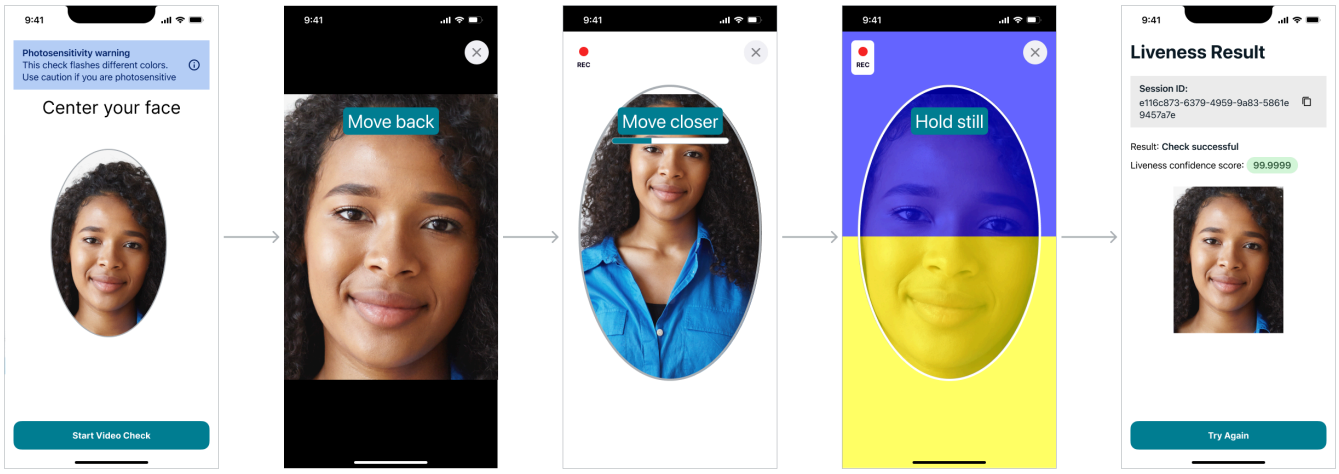
Face Liveness peut être utilisé pour toute une série cas d'utilisation. Par exemple, Face Liveness peut être utilisé en association avec la reconnaissance faciale (avec CompareFaces et SearchFacesByImage) pour la vérification de l'identité, pour l'estimation de l'âge sur les plateformes avec restriction d'accès basée sur l'âge et pour détecter de vrais utilisateurs humains tout en dissuadant les robots.

La carte de service IA Rekognition Face Liveness vous fournit de plus amples informations sur les cas d'utilisation auxquels le service est destiné, sur la manière dont la fonctionnalité machine learning

(ML) est utilisée par le service, et sur les principales considérations relatives à la conception et à l'utilisation responsables du service.

Vous pouvez définir des seuils pour Face Liveness et des scores de confiance pour Face Match. Les seuils que vous avez choisis doivent refléter votre cas d'utilisation. Vous envoyez ensuite une vérification d'identité approval/denial à l'utilisateur en fonction du score supérieur ou inférieur aux seuils. En cas de refus, demandez à l'utilisateur de réessayer, ou envoyez-le vers une autre méthode.

Le graphique suivant illustre le flux utilisateur, depuis les instructions jusqu'au résultat renvoyé, en passant par le contrôle de l'état de fonctionnement :



User-Side Exigences relatives à la vivacité du visage

Amazon Rekognition Face Liveness nécessite les spécifications minimales suivantes :

Appareils :

- L'appareil doit être équipé d'une caméra frontale
- Fréquence de rafraîchissement minimale de l'écran de l'appareil : 60 Hz
- Taille minimale d'affichage ou d'écran : 4 pouces
- L'appareil ne doit pas être jailbreaké ou rooté

Spécifications de la caméra :

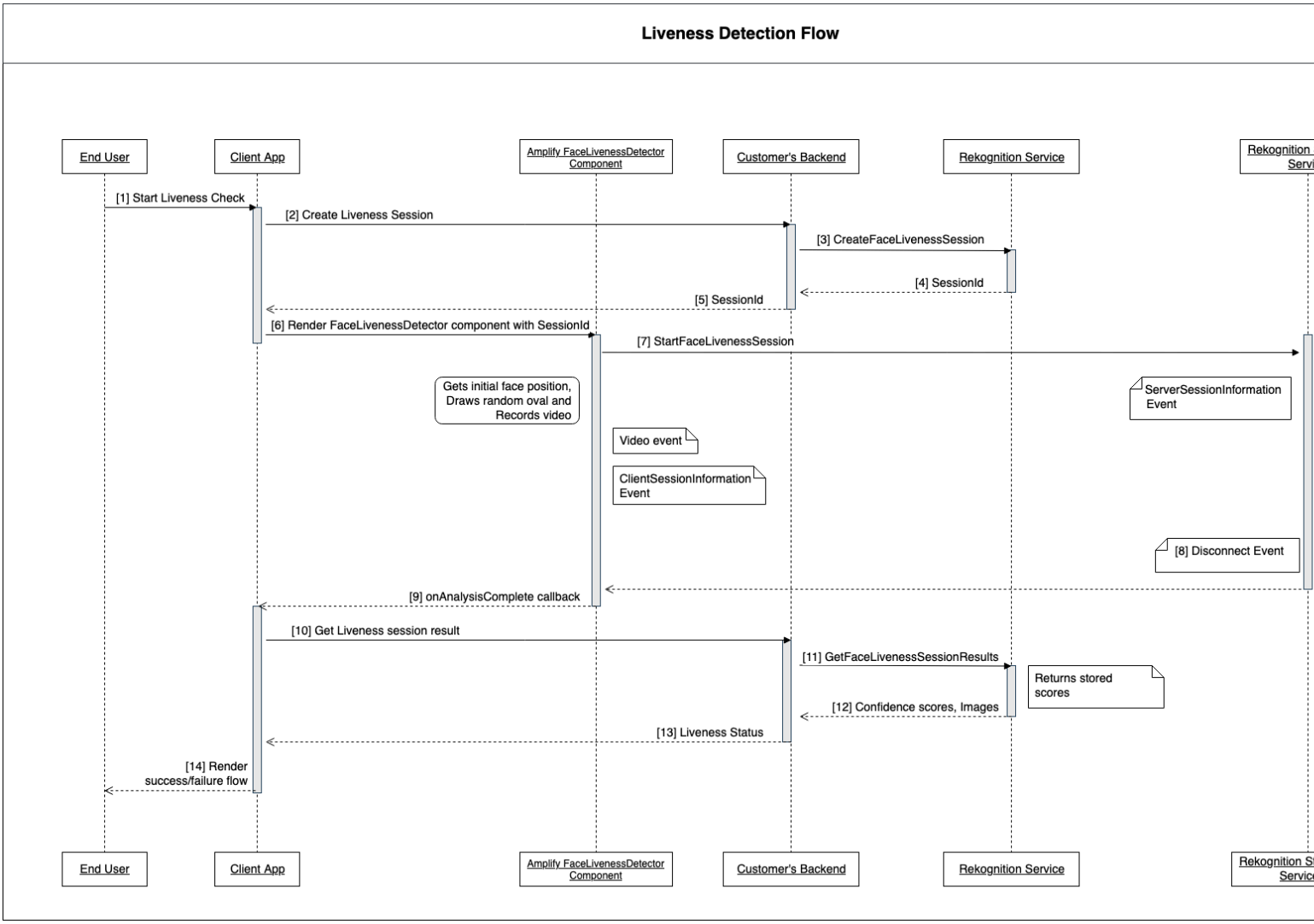
- Caméra couleur : la caméra frontale doit être capable d'enregistrer les couleurs.
- Pas de caméra virtuelle ni de logiciel de caméra.
- Capacité d'enregistrement minimale : 15 images par seconde.
- Résolution d'enregistrement vidéo minimale : 480 x 640 pixels.
- Lorsque les utilisateurs se servent d'une webcam avec un ordinateur de bureau pour effectuer un test Face Liveness, il est important de monter la webcam au-dessus de l'écran où le contrôle Face Liveness commence.

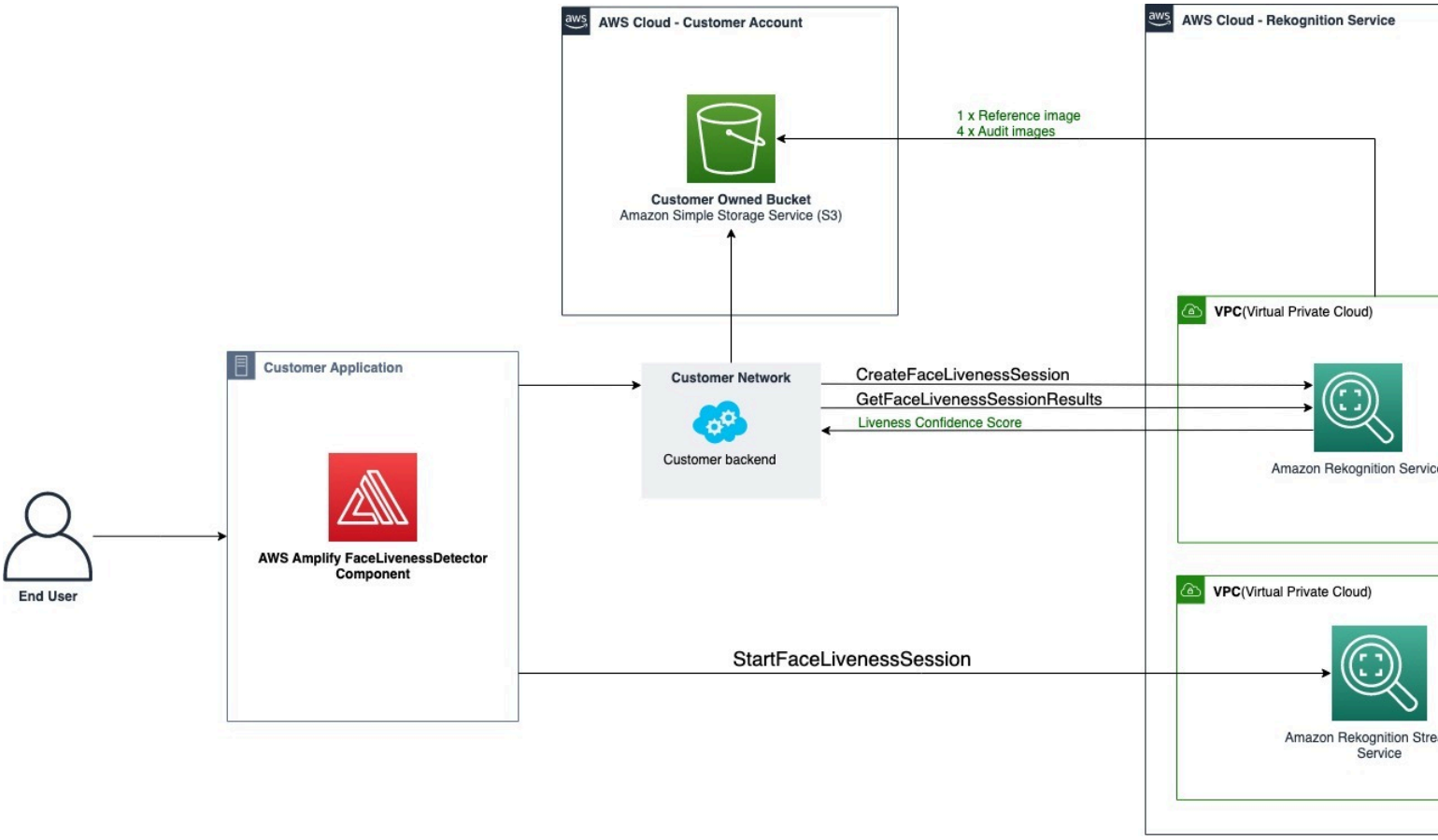
Bande passante minimale requise : 100 kbps

Navigateurs pris en charge : les trois dernières versions des principaux navigateurs, tels que Google Chrome, Mozilla Firefox, Apple Safari et Microsoft Edge. Pour obtenir des informations sur les versions supportées des navigateurs Chrome et Firefox, veuillez consulter Quels navigateurs sont supportés et peuvent être utilisés avec la console de gestion AWS ?

Architecture et diagrammes de séquence

Les diagrammes suivants décrivent le fonctionnement d'Amazon Rekognition Face Liveness en ce qui concerne l'architecture de la fonctionnalité et la séquence des opérations :


Liveness Detection Flow
End User
Client App
Amplify FaceLivenessDetector Component
Customer's Backend
Rekognition Service
[1] Start Liveness Check
[2] Create Liveness Session
[3] CreateFaceLivenessSession
[4] SessionId
[5] SessionId
[6] Render FaceLivenessDetector component with SessionId
[7] StartFaceLivenessSession
Gets initial face position, Draws random oval and Records video
ServerSessionInformation Event
Video event
ClientSessionInformation Event
[8] Disconnect Event
[9] onAnalysisComplete callback
[10] Get Liveness session result
[11] GetFaceLivenessSessionResults
Returns stored scores
[12] Confidence scores, Images
[13] Liveness Status
[14] Render success/failure flow



Le processus de vérification de la vivacité faciale comporte plusieurs étapes, comme indiqué ci-dessous :

1. L’utilisateur lance un test Face Liveness dans l’application cliente.
2. L’application cliente appelle le backend du client, qui à son tour appelle le service Amazon Rekognition. Le service crée une session Face Liveness et renvoie une session unique SessionId. Remarque : Une SessionId fois envoyé, il expire au bout de 3 minutes. Il ne reste donc que 3 minutes pour terminer les étapes 3 à 7 ci-dessous. Un nouveau SessionID doit être utilisé pour chaque vérification de Face Liveness. Si un identifiant de session donné est utilisé pour les vérifications ultérieures de Face Liveness, les vérifications échoueront. En outre, a SessionId expire 3 minutes après son envoi, rendant toutes les données Liveness associées à la session (par exemple, SessionID, image de référence, images d'audit, etc.) indisponibles.
3. L'application client affiche le composant FaceLivenessDetector Amplify à l'aide des rappels obtenus SessionId et appropriés.
4. Le FaceLivenessDetector composant établit une connexion au service de streaming Amazon Rekognition, affiche un ovale sur l'écran de l'utilisateur et affiche une séquence de lumières

colorées. FaceLivenessDetector enregistre et diffuse des vidéos en temps réel sur le service de streaming Amazon Rekognition.

5. Le service de streaming Amazon Rekognition traite la vidéo en temps réel, stocke les résultats et renvoie un DisconnectEvent au FaceLivenessDetector composant une fois la diffusion terminée.
6. Le FaceLivenessDetector composant appelle le `onAnalysisComplete` rappel pour signaler à l'application cliente que le streaming est terminé et que les scores sont prêts à être récupérés.
7. L'application cliente appelle le backend du client pour obtenir un drapeau booléen indiquant si l'utilisateur était actif ou non. Le backend du client fait la demande au service Amazon Rekognition pour obtenir le score de confiance, les images de référence et les images d'audit. Le backend client utilise ces attributs pour déterminer si l'utilisateur est en ligne et renvoie une réponse appropriée à l'application cliente.
8. Enfin, l'application client transmet la réponse au FaceLivenessDetector composant, qui affiche correctement le success/failure message pour terminer le flux.

Conditions préalables

Les conditions préalables à l'utilisation d'Amazon Rekognition Face Liveness sont les suivantes :

1. Créer un AWS compte
2. Configurer les SDK Face Liveness AWS
3. Configurer les ressources AWS Amplify

Étape 1 : Configurez un AWS compte

Si vous n'avez pas encore de AWS compte, suivez les étapes décrites à la section Créez un AWS Compte et utilisateur pour en créer un.

Étape 2 : configurer le Face Liveness AWS Kits SDK

Si ce n'est pas déjà fait, installez et configurez le AWS CLI et les AWS kits de développement logiciel. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

Il existe plusieurs méthodes pour authentifier les appels du AWS SDK. Les exemples de ce guide supposent que vous utilisez un profil d'informations d'identification par défaut pour appeler les commandes AWS CLI et les opérations d'API du AWS SDK.

Consultez la page Octroi d'un accès programmatique pour plus d'informations sur l'octroi de l'accès à votre compte utilisateur au AWS SDK que vous avez choisi. La page explique également comment utiliser un profil sur votre ordinateur local et comment exécuter l'exemple de code dans AWS des environnements.

Assurez-vous que l'utilisateur qui appelle les opérations Face Liveness dispose des autorisations appropriées pour appeler les opérations, telles que les autorisations `AmazonRekognitionFullAccess` et `AmazonS3FullAccess`.

Étape 3 : Configurer AWS Amplifier les ressources

Pour intégrer Amazon Rekognition Face Liveness à votre application, vous devez configurer le SDK Amplify pour utiliser le AWS composant Amplify. FaceLivenessDetector

Si ce n'est pas déjà fait, suivez les instructions pour configurer l'Interface de la ligne de commande AWS (AWS CLI) à la section Mise en route avec AWS CLI. Une fois la CLI installée, suivez les étapes de l'étape Configurer l'authentification présentées sur le site de documentation de l'interface utilisateur Amplify pour configurer vos ressources AWS Amplify.

Bonnes pratiques pour détecter la vivacité faciale

Les bonnes pratiques suivantes sont recommandées pour l'utilisation d'Amazon Rekognition Face Liveness. Les meilleures pratiques de Face Liveness incluent des directives concernant les endroits où les contrôles de Face Liveness doivent être effectués, l'utilisation d'images d'audit et le choix des seuils de confiance.

Consultez Recommandations pour l'utilisation de Face Liveness pour avoir la liste complète des meilleures pratiques.

Programmation des API Face Liveness d'Amazon Rekognition

Pour utiliser l'API Face Liveness d'Amazon Rekognition, vous devez créer un backend qui exécute les étapes suivantes :

1. Appelez `CreateFaceLivenessSession` pour lancer une session Face Liveness. Lorsque l'opération `CreateFaceLivenessSession` est terminée, l'interface utilisateur invite l'utilisateur à soumettre un selfie vidéo. Le FaceLivenessDetector composant d'AWS Amplify appelle ensuite `StartFaceLivenessSession` pour effectuer la détection de Liveness.

2. Appel GetFaceLivenessSessionResults pour renvoyer les résultats de détection associés à une session Face Liveness.
3. Procédez à la configuration de votre application React pour utiliser le FaceLivenessDetector composant en suivant les étapes du guide Amplify Liveness.

Avant d'utiliser Face Liveness, assurez-vous d'avoir créé un compte AWS, d'avoir configuré l'AWS CLI et les AWS SDK, ainsi que l'AWS Amplify. Vous devez également vous assurer que la politique IAM de votre API backend comporte des autorisations couvrant les points suivants : `GetFaceLivenessSessionResults` et `CreateFaceLivenessSession`. Consultez la section Conditions préalables pour plus d'informations.

Étape 1 : CreateFaceLivenessSession

CreateFaceLivenessSession L'opération d'API crée une session Face Liveness et renvoie une session unique`SessionId`.

Dans le cadre de la saisie de cette opération, il est également possible de spécifier l'emplacement d'un compartiment Amazon S3. Cela permet de stocker une image de référence et d'auditer les images générées pendant la session Face Liveness. Le compartiment Amazon S3 doit résider dans le compte AWS de l'appelant et dans la même région que le point de terminaison Face Liveness. De plus, les clés d'objet S3 sont générées par le système Face Liveness.

Il est également possible de fournir un `AuditImagesLimit`, qui est un nombre compris entre 0 et 4. Par défaut, l'attribut est défini sur 0. Le nombre d'images renvoyées est le meilleur effort possible, et il est basé sur la durée du selfie vidéo.

Enfin, vous pouvez fournir un ChallengePreference, y compris le type et la version du défi que vous souhaitez utiliser par défaut pour votre session. Par défaut, cette valeur est utilisée, FaceLivenessMovementAndLightChallenge mais elle peut être définie sur FaceMovementChallenge.

Exemple de requête

```
{
   "ClientRequestToken": "string",
   "KmsKeyId": "string",
   "Settings": {
      "AuditImagesLimit": number,
      "ChallengePreferences": [
```

```
            {
               "Type": "string",
               "Versions": {
                  "Maximum": "string",
                  "Minimum": "string"
               }
            }
         ],
         "OutputConfig": {
            "S3Bucket": "string",
            "S3KeyPrefix": "string"
         }
      }
}
```

Exemple de réponse

```
{
    {"SessionId": "0f959dbb-37cc-45d8-a08d-dc42cce85fa8"}
}
```

Étape 2 : StartFaceLivenessSession

Lorsque l'opération CreateFaceLivenessSession d'API est terminée, le composant AWS Amplify exécute l'opération StartFaceLivenessSession d'API. L'utilisateur est invité à prendre un selfie vidéo. Pour que le contrôle soit réussi, l'utilisateur doit placer son visage dans l'ovale affiché à l'écran tout en maintenant un bon éclairage. Pour de plus amples informations, veuillez consulter Recommandations pour l'utilisation de Face Liveness.

Cette opération d'API nécessite la vidéo capturée pendant la session Face Liveness, le SessionID obtenu à partir de l'opération d' CreateFaceLivenessSession API et un rappel. `onAnalysisComplete` Le rappel peut être utilisé pour signaler au backend d'appeler l'opération d' GetFaceLivenessSessionResults API, qui renvoie un score de confiance, une référence et des images d'audit.

Notez que cette étape est exécutée par le FaceLivenessDetector composant AWS Amplify sur l'application cliente. Il n'est pas nécessaire d'effectuer une configuration supplémentaire pour appeler `StartFaceLivenessSession`.

Étape 3 : GetFaceLivenessSessionResults

L'opération GetFaceLivenessSessionResults API récupère les résultats d'une session Face Liveness spécifique. Cette opération nécessite le sessionId comme entrée et renvoie le score de confiance Face Liveness correspondant. Elle fournit également une image de référence qui inclut un cadre de délimitation des visages et des images d'audit contenant également des cadres de délimitation des visages. Le score de confiance de Face Liveness est compris entre 0 et 100.

Exemple de requête

```
{"SessionId": "0f959dbb-37cc-45d8-a08d-dc42cce85fa8"}
```

Exemple de réponse

```
{
    "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"
}
```

Étape 4 : répondre aux résultats

Après la session Face Liveness, comparez le score de confiance du check au seuil spécifié. Si le score est supérieur au seuil, l'utilisateur peut passer à l'écran ou à la tâche suivante. Si la vérification échoue, l'utilisateur sera averti et invité à réessayer.

Appeler les API Face Liveness

Vous pouvez tester Amazon Rekognition Face Liveness avec n'importe quel AWS SDK compatible, comme le SDK AWS Python Boto3 ou le SDK AWS pour Java. https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/welcome.html Vous pouvez appeler les API `CreateFaceLivenessSession` et `GetFaceLivenessSessionResults` avec le SDK de votre choix. La section suivante illustre comment appeler ces API avec les SDK Python et Java.

Pour appeler les API Face Liveness :

- Si vous ne l'avez pas déjà fait, créez ou mettez à jour un utilisateur avec des autorisations `AmazonRekognitionFullAccess`. Pour plus d'informations, consultez Étape 1 : configurer un compte AWS et créer un utilisateur.
- Si vous ne l'avez pas déjà fait, installez et configurez l'AWS CLI et les AWS SDK. Pour plus d'informations, consultez Étape 2 : configuration de l'AWS CLI et des AWS SDK.

Python

L'extrait suivant montre comment vous pouvez appeler ces API dans vos applications Python. Notez que pour exécuter cet exemple, vous devez utiliser au moins la version 1.26.110 du SDK Boto3, bien que la version la plus récente du SDK soit recommandée.

```
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

L’extrait suivant montre comment vous pouvez appeler ces API dans vos applications 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

L'extrait suivant montre comment appeler les API Face Liveness avec le SDK AWS Java V2 :

```
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

L'extrait suivant montre comment appeler les API Face Liveness avec le SDK : AWS Node.Js

```
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)

L'extrait suivant montre comment appeler les API Face Liveness avec le AWS Node.Js SDK pour 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
// };
```

Configuration et personnalisation de votre application

Configuration de votre application

Votre application Face Liveness peut fonctionner sur les appareils mobiles ou les navigateurs Web de bureau. Vous devez configurer les composants Face Liveness pour les intégrer à la solution que vous avez choisie. Vous devez également vous assurer que votre application est autorisée à utiliser la caméra d'un appareil. Le guide Amplify Liveness fournit des instructions détaillées sur la procédure pour :

- Installer et configurer AWS Amplify
- Importation et rendu du FaceLivenessDetector composant
- Écouter les rappels

- Exemple de message d'erreur avec Render Amplify

Personnaliser votre application

Vous pouvez personnaliser certains composants de votre application Liveness à l'aide d'AWS Amplify.

Pour plus d'informations sur la traduction, consultez la documentation Amplify Authenticator.

Pour plus d'informations sur la personnalisation des composants et des thèmes Amplify, consultez la documentation Amplify concernant la thématisation.

Modèle de responsabilité partagée Face Liveness

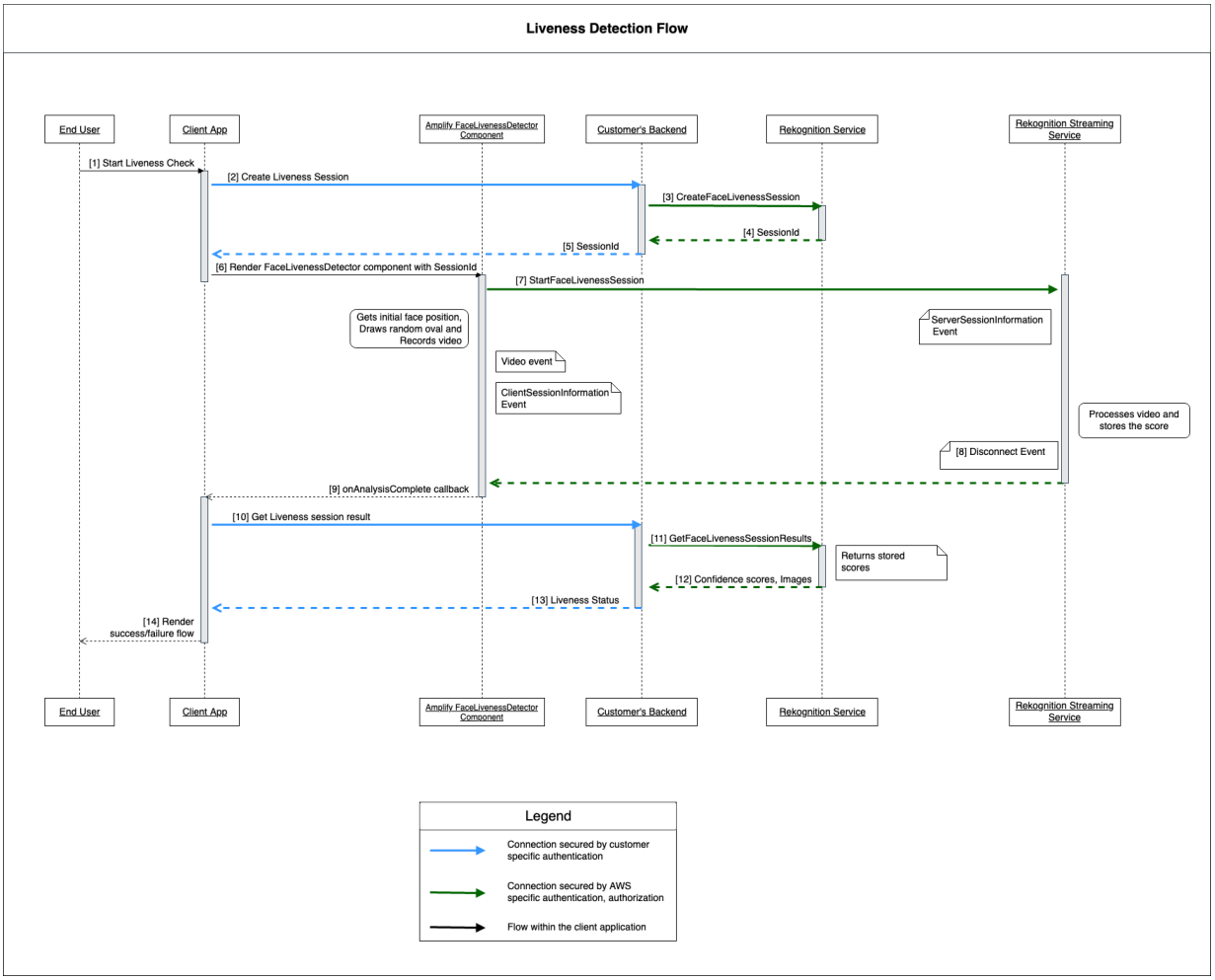
La sécurité et la conformité sont une responsabilité partagée entre vous AWS et vous, le client. Pour en savoir plus sur le modèle de responsabilité AWS partagée, cliquez ici.

1. Tous les appels au AWS service (via l'application client ou le backend client) sont authentifiés et autorisés avec AWS Auth (AWS Authentication). Il est de la responsabilité des propriétaires du service Face Liveness de s'en assurer.
2. Tous les appels vers le backend du client (depuis l'application client) sont authentifiés et autorisés par le client. Cette responsabilité incombe au client. Le client doit s'assurer que les appels provenant de l'application client sont authentifiés et qu'ils n'ont pas été manipulés de quelque manière que ce soit.
3. Le backend du client doit identifier l'utilisateur final qui exécute le défi Face Liveness. Il est de la responsabilité du client d'associer un utilisateur final à une session Face Liveness. Le service Face Liveness ne fait pas de distinction entre les utilisateurs finaux. Il est uniquement capable d'identifier l' AWS identité de l'appelant (que le client gère).
4. AWS recommande aux clients d'appliquer des contrôles de validation supplémentaires, tels que la géolocalisation (par exemple sur la base de l'adresse IP), les codes OTP (One Time Pass), etc., en plus de Face Liveness, conformément à leurs exigences en matière de cas d'utilisation et à leur niveau de sécurité.

Le réglage « FaceMovementAndLightChallenge » fournit la plus grande précision pour Rekognition Liveness en demandant aux utilisateurs de déplacer leur visage vers l'écran et de le maintenir immobile pendant une série de lumières clignotantes. Nous recommandons aux clients d'utiliser ce

paramètre par défaut. Les clients peuvent également activer le réglage « FaceMovementChallenge », qui réduit le temps de vérification de plusieurs secondes en éliminant les voyants clignotants. Bien que « FaceMovementAndLightChallenge » reste le meilleur paramètre pour optimiser la précision, « FaceMovementChallenge » permet aux clients de donner la priorité à des contrôles de durée de vie plus rapides. Lors du choix entre ces paramètres, les clients doivent tenir compte des exigences de leur cas d'utilisation, notamment les types d'attaques attendus, les taux de faux acceptations et de faux rejets souhaités, et également mettre en œuvre des contrôles supplémentaires tels que la géolocalisation (par exemple basée sur l'IP), les codes OTP (One Time Pass), etc. Les clients doivent prendre cette décision après avoir testé les performances de Liveness avec différents seuils de confiance en fonction de leur cas d'utilisation. Les clients sont tenus de mettre en œuvre des contrôles pour sécuriser l'appareil à partir duquel la vidéo est envoyée

Le schéma de flux suivant indique quels appels sont authentifiés par le service AWS ou par le client :



Tous les appels vers le service Amazon Rekognition Face Liveness sont AWS protégés par Auth (à l'aide d'un mécanisme de signature). AWS Cela comprend les appels suivants :

- [3] Appel CreateFaceLivenessSessiond'API (depuis le backend du client)
- [7] Appel StartFaceLivenessSessiond'API (depuis l'application cliente)
- [11] Appel GetFaceLivenessSessionResultsd'API (depuis le backend du client)

Tous les appels vers le backend du client doivent être dotés d'un mécanisme d'authentification et d'autorisation. Les clients doivent s'assurer que le tiers code/library /etc utilisé est activement maintenu et développé. Les clients doivent également s'assurer que le bon utilisateur final passe les appels vers la bonne session Face Liveness. Les clients doivent authentifier et autoriser les flux suivants :

- [2] Créer une session Face Liveness (depuis l'application cliente)
- [10] Obtenir le résultat de la session Face Liveness (depuis l'application cliente)

Les clients peuvent suivre le modèle de sécurité STRIDE pour s'assurer que leurs appels d'API sont protégés.

Type	Description	Contrôle de sécurité
Usurpation d'identité	Action de menace visant à accéder et à utiliser les informations d'identification d'un autre utilisateur, telles que le nom d'utilisateur et le mot de passe.	Authentification
Falsification	Action de menace visant à modifier ou à modifier des données persistantes de manière malveillante. Les exemples incluent les enregistrements d'une base de données et la modification de données en transit entre deux ordinateurs sur un réseau ouvert, tel qu'Internet.	Intégrité

Répudiation	Action de menace visant à effectuer des opérations interdites dans un système qui n'est pas en mesure de retracer les opérations.	Non-Repudiation
Divulgation d'informations	Action de menace visant à lire un fichier auquel l'accès n'a pas été autorisé ou à lire des données en transit.	Confidentialité
Déni de service	Action menaçante visant à refuser l'accès à des utilisate urs valides, par exemple en rendant un serveur Web temporairement indisponible ou inutilisable.	Disponibilité
Élévation de privilèges	Action contre les menaces visant à obtenir un accès privilégié aux ressources afin d'obtenir un accès non autorisé à des informations ou de compromettre un système.	Autorisation

AWS sécurise ses connexions de la manière suivante :

1. Calcul de la signature de la demande, puis vérification de la signature côté service. Les demandes sont authentifiées à l’aide de cette signature.
2. AWS les clients sont tenus de configurer les rôles IAM appropriés pour autoriser certains actions/operations. Ces rôles IAM sont nécessaires pour effectuer des appels vers le Service AWS.
3. Seules les requêtes HTTPS adressées au AWS service sont autorisées. Les demandes sont cryptées dans le réseau ouvert à l’aide du protocole TLS. Cela protège la confidentialité des demandes et préserve l’intégrité des demandes.
4. AWS le service enregistre suffisamment de données pour identifier les appels passés par les clients. Cela empêche les attaques de répudiation.

5. AWS le service est responsable du maintien d'une disponibilité suffisante

Le client est responsable de la sécurisation de ses appels de service et d'API de la manière suivante :

1. Le client doit s'assurer qu'il suit un mécanisme d'authentification approprié. Il existe différents mécanismes d'authentification qui peuvent être utilisés pour authentifier une demande. Les clients peuvent explorer l'authentification basée sur le condensé , OAuth , OpenID connect et d'autres mécanismes.
2. Les clients doivent s'assurer que leur service prend en charge les canaux de chiffrement appropriés (par exemple TLS/HTTPS) pour effectuer des appels d'API de service.
3. Les clients doivent s'assurer qu'ils enregistrent les données nécessaires pour identifier de manière unique un appel d'API et l'appelant. Ils doivent être en mesure d'identifier le client qui appelle leur API à l'aide de paramètres définis et de l'heure des appels.
4. Les clients doivent s'assurer que leur système est disponible et qu'ils sont protégés contre les attaques DDoS. Voici quelques exemples de techniques de défense contre les attaques DDoS.

Les clients ont pour responsabilité de garder leurs applications à jour. Pour de plus amples informations, veuillez consulter Mise à jour des directives de Face Liveness.

Mise à jour des directives de Face Liveness

AWS met régulièrement à jour AWS les SDK Face Liveness (utilisés dans le backend client) et les FaceLivenessDetector composants des SDK AWS Amplify (utilisés dans les applications clientes) afin de fournir de nouvelles fonctionnalités, des API mises à jour, une sécurité renforcée, des corrections de bogues, des améliorations de la convivialité, etc. Nous vous recommandons de maintenir les SDK à jour pour garantir un fonctionnement optimal de la fonctionnalité. Si vous continuez à utiliser les anciennes versions des SDK, les demandes peuvent être bloquées pour des raisons de maintenance et de sécurité.

Face Liveness nécessite que vous utilisiez le FaceLivenessDetector composant, inclus dans les SDK AWS Amplify (React, iOS, Android).

Gestion des versions et délais

Nous gérons les versions des composants clés suivants de la fonctionnalité Face Liveness. Nous suivons un format de gestion des versions sémantique. Par exemple, un format de version X.Y.Z

dans lequel X représente la version majeure, Y représente la version mineure et Z représente la version du correctif.

- Les défis des utilisateurs de Face Liveness (par exemple, FaceMovementAndLightChallenge challenge) font partie de l'API StartFaceLivenessSession
- FaceLivenessDetector les composants fournis via les SDK AWS Amplify sont utilisés dans les applications clientes

Versions majeures : nous réservons les mises à jour majeures pour des raisons de sécurité critiques, d'API défaillantes et de mises à jour exceptionnelles en termes d'utilisabilité. Les applications et le backend client doivent être mis à jour dès que possible pour que vous puissiez continuer à utiliser les fonctionnalités de Face Liveness. Une fois que nous avons publié une nouvelle version majeure, nous prenons en charge la version majeure précédente pendant 120 jours à compter de la date de publication de la nouvelle version. Nous pouvons bloquer les demandes provenant de la version majeure précédente après 120 jours.

Versions mineures : nous réservons les mises à jour mineures pour des fonctionnalités et améliorations importantes en matière de sécurité et d'utilisabilité. Nous vous recommandons vivement d'appliquer ces mises à jour. Bien que nous nous efforcions de garantir la rétrocompatibilité des mises à jour mineures le plus longtemps possible, nous pouvons annoncer la fin du support d'une version mineure précédente 180 jours après la sortie d'une nouvelle version mineure.

Versions de correctif : nous réservons les mises à jour des versions de correctif pour les corrections de bogues et les améliorations facultatives. Bien que nous vous recommandions de maintenir votre version à jour pour une sécurité et une expérience utilisateur optimales, nous nous efforçons de garantir que les mises à jour soient totalement rétrocompatibles jusqu'à ce que nous publiions une nouvelle version majeure ou mineure.

La période de gestion des versions (120 jours pour les versions majeures et 180 jours pour les versions mineures) s'applique à la mise à jour du SDK dans votre application, au téléchargement de votre application sur l'App Store ou le site Web, et au téléchargement de la dernière version de l'application par les utilisateurs.

Version, publication et matrice de compatibilité

La sortie d'une version majeure pour répondre à un défi lié à un FaceLivenessDetector composant ou à un utilisateur coïncide souvent. Pour vous aider à suivre les dépendances entre les versions, consultez les ressources liées dans les tableaux suivants.

Versions de SDK et journaux des modifications :

FaceLivenessDetector pour le SDK Web		FaceLivenessDetector pour iOS SDK	FaceLivenessDetector pour Android SDK
Version actuelle	Journal des modifications	Actuel version/Changelog	Actuel version/Changelog

Défis pour les utilisateurs :

Nom du défi	Version	Date de publication	Date de retraite
FaceMovementAndLightChallenge	v1.0.0	4/10/2023	N/A
FaceMovementChallenge	v1.0.0	4/30/2025	N/A

Communication des nouveautés

AWS communique les nouvelles versions via les canaux suivants :

- Notifications électroniques de mise à jour de l'état du service envoyées à l'adresse e-mail associée à l'identifiant de compte Face Liveness.
- Mises à jour publiées pour AWS les SDK et notifications associées sur les GitHub dépôts respectifs.
- Mises à jour publiées pour les SDK AWS Amplify et les notifications associées sur les dépôts respectifs. GitHub

Nous vous recommandons de vous abonner à ces chaînes pour rester informé.

Face Liveness FAQ

Utilisez les rubriques suivantes de la FAQ pour trouver des réponses aux questions fréquemment posées sur Rekognition Face Liveness.

- Quels sont les résultats d’un contrôle de la vivacité faciale ?

  Rekognition Face Liveness fournit les résultats suivants pour chaque contrôle de de la vivacité faciale :

  - Score de confiance : un score numérique compris entre 0 et 100 est renvoyé. Ce score indique la probabilité que la vidéo du selfie provienne d’une personne réelle et non d’un mauvais acteur utilisant une parodie.
  - High-Quality Image : une seule image de haute qualité est extraite de la vidéo du selfie. Ce cadre peut être utilisé à diverses fins, telles que la comparaison des visages, l’estimation de l’âge ou la recherche de visages.
  - Images d’audit : jusqu’à quatre images sont renvoyées par la vidéo selfie, qui peuvent être utilisées à des fins de piste d’audit.
- Est-ce que Rekognition Face Liveness est compatible avec les tests iBeta Presentation Attack Detection (PAD) ?

  Les tests de détection des attaques de présentation (PAD) d'iBeta Quality Assurance sont effectués conformément à la norme ISO/IEC 30107-3. iBeta est accrédité par NIST/NVLAP pour tester et fournir des résultats selon cette norme PAD. Rekognition Face Liveness a réussi les tests de conformité iBeta Presentation Attack Detection (PAD) de niveaux 1 et 2 avec un score PAD parfait. Le rapport peut être consulté sur la page Web d’iBeta ici.
- Comment puis-je obtenir un cadre de haute qualité et des cadres supplémentaires ?

  La trame de haute qualité et les images supplémentaires peuvent être renvoyées sous forme d'octets bruts ou téléchargées vers un compartiment Amazon S3 que vous spécifiez, en fonction des configurations de votre demande d'CreateFaceLivenessSessionAPI.
- Puis-je modifier l’emplacement des lumières ovales et colorées ?

  Non. L'emplacement ovale et les lumières colorées sont des caractéristiques qui permettent une plus grande précision et ne peuvent donc pas être personnalisées.
- Puis-je personnaliser l’interface utilisateur selon mon application ?

  Oui, vous pouvez personnaliser la plupart des composants de l’écran tels que le thème, la couleur, la langue, le contenu du texte et la police pour les aligner sur votre application. Vous trouverez des

informations détaillées sur la personnalisation de ces composants dans la documentation de nos composants d'interface utilisateur React, Swift et Android.

- Puis-je personnaliser le temps et l'heure du compte à rebours pour qu'il s'adapte à un visage ovale ?

  Non, le compte à rebours et le temps d'ajustement du visage ont été prédéterminés sur la base d'études internes à grande échelle menées auprès de milliers d'utilisateurs, dans le but de fournir un équilibre optimal entre une précision et une latence accrues. Pour cette raison, ces paramètres horaires ne peuvent pas être personnalisés.

- Les lumières de différentes couleurs répondent-elles aux directives d'accessibilité ?

  Oui, les différentes couleurs de lumière de notre produit sont conformes aux directives d'accessibilité décrites dans les WCAG 2.1. Comme l'ont vérifié plus de 1 000 utilisateurs, l'expérience utilisateur affiche environ deux couleurs par seconde, ce qui est conforme à la recommandation de limiter les couleurs à trois par seconde. Cela réduit le risque de déclencher des crises d'épilepsie dans la majorité de la population.

- Le SDK ajuste-t-il la luminosité de l'écran pour des résultats optimaux ?

  Les SDK mobiles Face Liveness (pour Android et iOS) ajustent automatiquement la luminosité lorsque le contrôle est lancé. Cependant, pour le SDK Web, certaines restrictions relatives aux pages Web empêchent le réglage automatique de la luminosité. Dans de tels cas, nous nous attendons à ce que l'application Web demande aux utilisateurs finaux d'augmenter manuellement la luminosité de l'écran pour des résultats optimaux.

- Faut-il que ce soit un ovale ? Pourrions-nous utiliser d'autres formes similaires ?

  Non, la taille, la forme et l'emplacement de l'ovale ne sont pas personnalisables. Le design ovale spécifique a été soigneusement choisi pour son efficacité à capturer et à analyser avec précision les mouvements du visage. Par conséquent, la forme ovale ne peut pas être modifiée.

- Qu'est-ce que la latence de bout en bout ?

  Nous mesurons la latence de bout en bout entre le moment où l'utilisateur lance l'action requise pour effectuer le test de réactivité et le moment où il obtient le résultat (réussite ou échec). Dans le meilleur des cas, la latence est de 5 s. En moyenne, nous nous attendons à ce qu'elle soit d'environ 7 secondes. Dans le pire des cas, la latence est de 11 s. Nous constatons une variation

de la latence de bout en bout, car elle dépend du temps nécessaire à l'utilisateur pour effectuer l'action requise (c'est-à-dire déplacer son visage dans l'ovale), de la connectivité réseau, de la latence de l'application, etc.

- Puis-je utiliser la fonction Face Liveness sans le SDK Amplify ?

  Non, le SDK Amplify est nécessaire pour utiliser la fonctionnalité Rekognition Face Liveness.

- Où puis-je trouver les états d'erreur associés à Face Liveness ?

  Vous pouvez voir ici les différents états d'erreur de Face Liveness.

- Face Liveness n'est pas disponible dans ma région. Comment puis-je utiliser cette fonctionnalité ?

  Vous pouvez choisir d'appeler Face Liveness dans toutes les régions où il est disponible, en fonction de votre trafic et de votre proximité. Face Liveness est actuellement disponible dans les AWS régions suivantes :

  - USA Est (Virginie du Nord)
  - USA Ouest (Oregon)
  - Europe (Irlande)
  - Asie-Pacifique (Tokyo, Mumbai)
  - Amérique du Sud (São Paulo)
  - Asie-Pacifique (Malaisie)
  - Asie-Pacifique (Thaïlande)

  Même si votre AWS compte est situé dans une autre région, la différence de latence ne devrait pas être significative. Vous pouvez obtenir des cadres de selfie et des images d'audit de haute qualité via la localisation Amazon S3 ou sous forme d'octets bruts, mais votre compartiment Amazon S3 doit correspondre à la AWS région de Face Liveness. Si elles sont différentes, vous devez recevoir les images sous forme d'octets bruts.

- Amazon Rekognition Liveness Detection utilise-t-il le contenu des clients pour améliorer le service ?

  Vous pouvez refuser que vos entrées d'image et de vidéo soient utilisées pour améliorer ou développer la qualité de Rekognition et d'autres technologies d'learning/artificialintelligence artificielle d'Amazon en utilisant la politique de désinscription de l'organisation AWS . Pour plus

d'informations sur la procédure de désinscription, consultez la politique de désinscription de Managing AI Services.

Analyse en bloc

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Amazon Rekognition Bulk Analysis vous permet de traiter une grande collection d'images de manière asynchrone en utilisant un fichier manifeste avec l'opération. StartMediaAnalysisJob La sortie de chaque image individuelle correspond à la sortie renvoyée par l'opération que vous utilisez pour l'analyse.

Actuellement, Rekognition prend en charge l'analyse par le biais de cette opération. DetectModerationLabels

Le nombre d'images traitées avec succès par la tâche vous sera facturé. Les résultats d'une tâche terminée sont envoyés dans un compartiment Amazon S3 spécifié.

Notez que l'analyse en bloc ne prend pas en charge l'intégration Amazon A2I.

L'API peut détecter les types de contenu animés ou illustrés, et les informations relatives au type de contenu détecté sont renvoyées dans le cadre de la réponse.

Rubriques

- Traitement d'images en bloc
- Manifestes de sortie d'analyse en bloc
- Type de contenu
- Vérification des prédictions et des adaptateurs d'entraînement

Traitement d'images en bloc

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Vous pouvez démarrer une nouvelle tâche d'analyse en bloc en soumettant un fichier manifeste et en appelant l' StartMediaAnalysisJob opération. Le fichier manifeste d'entrée contient des références à des images d'un compartiment Amazon S3 et est formaté comme suit :

```
{"source-ref": "s3://foo/bar/1.jpg"}
```

Pour créer une tâche d'analyse en bloc (CLI)

1. Si vous ne l'avez pas déjà fait :

   a. Créez ou mettez à jour un utilisateur avec `AmazonRekognitionFullAccess` et autorisations `AmazonS3ReadOnlyAccess`. Pour de plus amples informations, veuillez consulter Étape 1 : configurer un compte AWS et créer un utilisateur.

   b. Installez et configurez AWS CLI les AWS SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Téléchargez des images dans votre compartiment S3.

   Pour en savoir plus, consultez Chargement d'objets dans Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

3. Utilisez les commandes suivantes pour créer et récupérer des tâches d'analyse en bloc.

CLI

Utilisez la commande suivante pour appeler l'StartMediaAnalysisJobopération à des fins d'analyse avec l' DetectModerationLabels opération :

```
# 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"
```

Vous pouvez obtenir des informations sur une tâche donnée, telles que le chemin Amazon S3 du compartiment dans lequel les résultats et les fichiers récapitulatifs sont stockés, à l'aide de cette GetMediaAnalysisJobopération. Vous lui fournissez un numéro de poste renvoyé par StartMediaAnalysisJob ou ListMediaAnalysisJob. Les informations relatives aux différentes tâches ne sont conservées que pendant un an.

```
# Request
aws rekognition get-media-analysis-job \
--job-id customer-job-id
```

Vous pouvez répertorier toutes vos analyses groupées à l'aide de l'opération de ListMediaAnalysisJobstâche, qui renvoie des pages de tâches. Avec l'argument `max-results`, vous pouvez spécifier le nombre maximum de tâches à renvoyer par page, limité à la valeur de `max-results`. Un maximum de 100 résultats sont renvoyés par page. Les informations relatives aux différentes tâches ne sont conservées que pendant un an.

```
# Request
# Specify number of jobs to return per page, limited to max-results.
aws rekognition list-media-analysis-jobs --max-results 1
```

Manifestes de sortie d'analyse en bloc

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

La tâche d'analyse en bloc génère un fichier manifeste de sortie contenant les résultats de la tâche, ainsi qu'un résumé du manifeste contenant des statistiques et des détails sur les erreurs éventuelles lors du traitement des entrées du manifeste d'entrée.

Si des entrées dupliquées ont été incluses dans le manifeste d'entrée, la tâche ne tente pas de filtrer les entrées uniques, et traite à la place toutes les entrées fournies.

Le fichier manifeste de sortie est formaté comme suit :

```
// 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"}]}}
```

Le résumé du manifeste de sortie est formaté comme suit :

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

Type de contenu

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

Les informations relatives au type de contenu multimédia analysé par StartMediaAnalysisJob opération sont renvoyées par l' GetMediaAnalysisJob opération. ContentType peut appartenir à l'une des deux catégories suivantes :

- Contenu animé, qui inclut les jeux vidéo et les animations (par exemple, dessins animés, bandes dessinées, mangas, anime).
- Contenu illustré, qui comprend des dessins, des peintures et des croquis.

Vérification des prédictions et des adaptateurs d'entraînement

> **Note**
>
> Les vidéos en streaming et l'analyse d'images en masse ne sont plus disponibles pour les nouveaux clients. Pour de plus amples informations, veuillez consulter Modifications relatives à la disponibilité des fonctionnalités Amazon Rekognition.
> Cette modification n'a aucune incidence sur la disponibilité des autres fonctionnalités d'Amazon Rekognition.

L’analyse en bloc peut également être utilisée via la Console Rekognition pour obtenir des prédictions pour un lot d’images, vérifier ces prédictions, puis créer un adaptateur à l’aide des prédictions vérifiées. Les adaptateurs vous permettent d’améliorer la précision de toutes les opérations de Rekognition prises en charge.

À l’heure actuelle, vous pouvez créer des adaptateurs à utiliser avec la fonctionnalité de modération personnalisée de Rekognition. En créant un adaptateur et en le fournissant à

l'DetectModerationLabelsopération, vous pouvez obtenir une meilleure précision pour les tâches de modération du contenu liées à votre cas d'utilisation spécifique.

Pour plus d’informations sur la modération personnalisée, consultez Amélioration de la précision grâce à la modération personnalisée. Consultez Analyse et vérification en bloc pour savoir comment vérifier les prédictions établies à l’aide de l’analyse en bloc. Pour un didacticiel expliquant comment utiliser la console Rekognition pour vérifier les prédictions et créer un adaptateur, consultez Tutoriel sur l’adaptateur de modération personnalisé.

Tutoriels

Ces didacticiels multiservices montrent comment utiliser les opérations d'API de Rekognition conjointement avec d'autres AWS services pour créer des exemples d'applications et accomplir diverses tâches. La plupart de ces didacticiels utilisent Amazon S3 pour stocker des images ou des vidéos. Les autres services couramment utilisés incluent AWS Lambda.

Rubriques

- Stockage des données Amazon Rekognition avec Amazon RDS et DynamoDB
- Utilisation d'Amazon Rekognition et Lambda pour étiqueter les actifs dans un compartiment Amazon S3
- Création AWS applications d'analyse vidéo
- Création d'une fonction Lambda Amazon Rekognition
- Utilisation d'Amazon Rekognition pour la vérification d'identité
- Utilisation d'Amazon Rekognition (REK) pour détecter les étiquettes destinées aux applications marketing
- Détection d'étiquettes dans une image à l'aide de Lambda et Python

Stockage des données Amazon Rekognition avec Amazon RDS et DynamoDB

Lorsque vous utilisez les API d'Amazon Rekognition, il est important de se rappeler que les opérations d'API n'enregistrent aucune des étiquettes générées. Vous pouvez enregistrer ces étiquettes en les plaçant dans la base de données, avec les identifiants des images respectives.

Ce didacticiel explique comment détecter les étiquettes et enregistrer les étiquettes détectées dans une base de données. L'exemple d'application développé dans ce didacticiel lira les images d'un compartiment Amazon S3, appellera l'DetectLabelsopération sur ces images et stockera les étiquettes obtenues dans une base de données. L'application stocke les données dans une instance de base de données Amazon RDS, ou dans une base de données DynamoDB, selon le type de base de données que vous souhaitez utiliser.

Vous allez utiliser l'AWS SDK pour Python ou ce didacticiel. Vous pouvez également consulter le GitHub référentiel d'exemples du SDK de AWS documentation pour d'autres didacticiels Python.

Rubriques

- Conditions préalables
- Obtenir des étiquettes pour les images dans un compartiment Amazon S3
- Création d'un tableau Amazon DynamoDB
- Chargement de données vers DynamoDB
- Création d'une base de données MySQL dans Amazon RDS
- Téléchargement de données vers un tableau MySQL Amazon RDS

Conditions préalables

Avant de commencer ce didacticiel, vous devez installer Python et suivre les étapes requises pour configurer le AWS SDK Python. En outre, vérifiez que vous avez effectué les opérations suivantes :

Création d'un compte AWS et d'un rôle IAM

Installation de SDK Python (Boto3)

Configurez correctement vos informations d' AWS accès

Compartiment Amazon S3 créé rempli d'images

Création d'une instance de base de données RDS, si vous utilisez RDS pour stocker des données

Obtenir des étiquettes pour les images dans un compartiment Amazon S3

Commencez par écrire une fonction qui prend le nom d'une image dans votre compartiment Amazon S3 et récupére cette image. Cette image sera affichée pour confirmer que les images correctes sont transmises à un appel DetectLabelsqui figure également dans la fonction.

1. Trouvez le compartiment Amazon S3 que vous souhaitez utiliser et notez son nom. Appelez ce compartiment Amazon S3 et lisez les images qu'il contient. Assurez-vous que votre bucket contient des images à transmettre à l'DetectLabelsopération.
2. Écrivez le code pour la connexion à votre compartiment Amazon S3. Vous pouvez vous connecter à la ressource Amazon S3 à l'aide de Boto3 pour récupérer une image à partir d'un compartiment Amazon S3. Une fois connecté à la ressource Amazon S3, vous pouvez accéder à votre compartiment en fournissant à la méthode Bucket le nom de votre compartiment Amazon S3. Une fois connecté au compartiment Amazon S3, vous pouvez récupérer des images du compartiment à l'aide de la méthode Object. En utilisant Matplotlib, vous pouvez utiliser cette

connexion pour visualiser vos images au fur et à mesure de leur traitement. Boto3 est également utilisé pour la connexion au client Rekognition.

Dans le code suivant, indiquez votre région dans le paramètre region_name. Vous allez transmettre le nom du compartiment Amazon S3 et le nom de l'image à DetectLabels, qui renvoie les étiquettes de l'image correspondante. Après avoir sélectionné uniquement les étiquettes dans la réponse, le nom de l'image et les étiquettes sont renvoyés.

```
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. Enregistrez ce code dans un fichier appelé get_images.py.

Création d'un tableau Amazon DynamoDB

Le code suivant utilise Boto3 pour se connecter à DynamoDB et il utilise la méthode DynamoDB `CreateTable` pour créer un tableau nommé Images. Le tableau possède une clé primaire composite comprenant une clé de partition appelée Image, et une clé de tri appelée Étiquettes. La clé Image contient le nom de l'image, tandis que la clé Étiquettes stocke les étiquettes attribuées à cette image.

```
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)
```

Enregistrez ce code dans un éditeur et exécutez-le une fois pour créer un tableau DynamoDB.

Chargement de données vers DynamoDB

Maintenant que la base de données DynamoDB a été créée et que vous disposez d'une fonction permettant d'obtenir des étiquettes pour les images, vous pouvez stocker les étiquettes dans DynamoDB. Le code suivant récupère toutes les images d'un compartiment S3, obtient leurs étiquettes et stocke les données dans DynamoDB.

1. Vous devez écrire le code permettant de télécharger les données vers DynamoDB. Une fonction appelée `get_image_names` est utilisée pour se connecter à votre compartiment Amazon S3 et renvoie les noms de toutes les images du compartiment sous forme de liste. Vous allez transmettre cette liste à la fonction `read_image_from_S3`, qui est importée à partir du fichier `get_images.py` que vous avez créé.

   ```
   import boto3
   import json
   from get_images import read_image_from_s3

   boto3 = boto3.Session()

   def get_image_names(name_of_bucket):

       s3_resource = boto3.resource('s3')
       my_bucket = s3_resource.Bucket(name_of_bucket)
       file_list = []
       for file in my_bucket.objects.all():
           file_list.append(file.key)
       return file_list
   ```

2. La fonction `read_image_from_S3` que nous avons créée précédemment renvoie le nom de l'image en cours de traitement et le dictionnaire des libellés associés à cette image. Une fonction

appelée `find_values` est utilisée pour obtenir uniquement les étiquettes de la réponse. Le nom de l'image et ses étiquettes sont alors prêts à être téléchargés dans votre tableau 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. Vous allez utiliser une troisième fonction appelée `load_data`, pour charger réellement les images et les étiquettes dans le tableau DynamoDB que vous avez créé.

```
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. C'est ici que les trois fonctions que nous avons définies précédemment sont appelées et que les opérations sont effectuées. Ajoutez les trois fonctions définies ci-dessus, ainsi que le code ci-dessous, à un fichier Python. Exécutez le code.

```
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!")
```

Vous venez de générer des étiquettes pour vos images et de les stocker dans une instance DynamoDB. DetectLabels Assurez-vous de supprimer toutes les ressources que vous avez créées pendant que vous suivez ce didacticiel. Vous évitez ainsi d'être facturé pour des ressources que vous n'utilisez pas.

Création d'une base de données MySQL dans Amazon RDS

Avant d'aller plus loin, assurez-vous d'avoir terminé la procédure de configurationpour Amazon RDS et d'avoir créé une instance de base de données MySQL à l'aide d'Amazon RDS.

Le code suivant utilise la bibliothèque PyMySQL et votre instance de base de données Amazon RDS. Il crée un tableau contenant les noms de vos images et les étiquettes associées à ces images. Amazon RDS reçoit des commandes pour créer des tableaux et insérer des données dans des tableaux. Pour utiliser Amazon RDS, vous devez vous connecter à l'hôte Amazon RDS à l'aide de votre nom d'hôte, de votre nom d'utilisateur et de votre mot de passe. Vous allez vous connecter à Amazon RDS en fournissant ces arguments à la `connect` fonction PyMy SQL et en créant une instance de curseur.

1. Dans le code suivant, remplacez la valeur d'hôte par le point de terminaison de votre hôte Amazon RDS, et remplacez la valeur d'utilisateur par le nom d'utilisateur principal associé à votre instance Amazon RDS. Vous devez également remplacer le mot de passe par le mot de passe principal de votre utilisateur principal.

```
import pymysql

host = "host-endpoint"
user = "username"
password = "master-password"
```

2. Créez une base de données et un tableau dans lesquels insérer les données de votre image et de votre étiquette. Pour ce faire, exécutez et validez une requête de création. Le code suivant crée une base de données. N'exécutez ce code qu'une seule fois.

```
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. Une fois la base de données créée, vous devez créer un tableau dans lequel insérer les noms et les étiquettes de vos images. Pour créer un tableau, vous devez d'abord transmettre la commande d'utilisation SQL, ainsi que le nom de votre base de données, à la fonction `execute`. Une fois la connexion établie, une requête pour créer un tableau est exécutée. Le code suivant se connecte à la base de données, puis crée un tableau contenant à la fois une clé primaire, appelée `image_id`, et un attribut de texte stockant les étiquettes. Utilisez les importations et les variables que vous avez définies précédemment, puis exécutez ce code pour créer un tableau dans votre base de données.

```
# 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!")
```

Téléchargement de données vers un tableau MySQL Amazon RDS

Après avoir créé la base de données Amazon RDS et un tableau dans la base de données, vous pouvez obtenir des étiquettes pour vos images et les stocker dans la base de données Amazon RDS.

1. Connectez-vous à votre compartiment Amazon S3 et récupérez les noms de toutes les images qu'il contient. Ces noms d'images seront transmis à la fonction `read_image_from_s3` que vous avez créée précédemment pour obtenir les étiquettes de toutes vos images. Le code

suivant se connecte à votre compartiment Amazon S3 et renvoie une liste de toutes les images qu'il contient.

```
import pymysql
from get_images import read_image_from_s3
import json
import boto3

host = "host-endpoint"
user = "username"
password = "master-password"

conn = pymysql.connect(host=host, user=user, passwd=password)
print(conn)
cursor = conn.cursor()
print("Connection successful")

def get_image_names(name_of_bucket):

    s3_resource = boto3.resource('s3')
    my_bucket = s3_resource.Bucket(name_of_bucket)
    file_list = []
    for file in my_bucket.objects.all():
        file_list.append(file.key)
    return file_list
```

2. La réponse de l'DetectLabelsAPI ne contient pas que les étiquettes. Écrivez donc une fonction pour extraire uniquement les valeurs des étiquettes. La fonction suivante renvoie une liste contenant uniquement les étiquettes.

```
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. Vous avez besoin d'une fonction pour insérer les noms et les étiquettes des images dans votre tableau. La fonction suivante exécute une requête d'insertion et insère une paire donnée de nom d'image et d'étiquettes.

```
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. Enfin, vous devez exécuter les fonctions que vous avez définies ci-dessus. Dans le code suivant, les noms de toutes les images de votre bucket sont collectés et fournis à la fonction qui appelle DetectLabels. Les étiquettes et le nom de l'image à laquelle elles s'appliquent sont ensuite chargés dans votre base de données Amazon RDS. Ajoutez les trois fonctions définies ci-dessus, ainsi que le code ci-dessous, à un fichier Python. Exécutez le fichier 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!")
```

Vous avez réussi DetectLabels à générer des étiquettes pour vos images et à les stocker dans une base de données MySQL à l'aide d'Amazon RDS. Assurez-vous de supprimer toutes les ressources

que vous avez créées pendant que vous suivez ce didacticiel. Vous évitez ainsi d'être facturé pour des ressources que vous n'utilisez pas.

Pour d'autres exemples AWS multiservices, consultez le référentiel d'exemples GitHub du SDK de AWS documentation.

Utilisation d'Amazon Rekognition et Lambda pour étiqueter les actifs dans un compartiment Amazon S3

Dans ce didacticiel, vous allez créer une AWS Lambda fonction qui balise automatiquement les actifs numériques situés dans un compartiment Amazon S3. La fonction Lambda lit tous les objets dans un compartiment Amazon S3 donné. Pour chaque objet du compartiment, elle transmet l'image au service Amazon Rekognition afin de générer une série d'étiquettes. Chaque étiquette est utilisée pour créer une étiquette qui est appliquée à l'image. Une fois que vous avez exécuté la fonction Lambda, elle crée automatiquement des balises basées sur toutes les images d'un compartiment Amazon S3 donné et les applique aux images.

Supposons, par exemple, que vous exécutiez la fonction Lambda et que vous disposiez de cette image dans un compartiment Amazon S3.

L'application crée ensuite automatiquement des balises et les applique à l'image.

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**
>
> Les services que vous utilisez dans ce didacticiel font partie du niveau AWS gratuit. Lorsque vous aurez terminé le didacticiel, nous vous recommandons de supprimer toutes les ressources que vous avez créées pendant le didacticiel afin qu'elles ne vous soient pas facturées.

Ce didacticiel utilise le AWS SDK pour Java version 2. Consultez le GitHub référentiel d'exemples du SDK de AWS documentation pour des didacticiels supplémentaires sur Java V2.

Rubriques

- Conditions préalables
- Configuration du rôle IAM Lambda
- Création du projet
- Ecriture du code
- Package du projet
- Déployez la fonction Lambda.
- Test de la méthode Lambda

Conditions préalables

Avant de commencer, vous devez suivre les étapes décrites dans Configuration du AWS SDK for Java. Veillez à effectuer les opérations suivantes :

- Java 1.8 JDK.
- Apache Maven 3.6 ou version ultérieure.
- Un compartiment Amazon S3 contenant 5 à 7 images nature. Ces images sont lues par la fonction Lambda.

Configuration du rôle IAM Lambda

Ce didacticiel utilise les services Amazon Rekognition et Amazon S3. Configurez le rôle Lambda support pour disposer de politiques lui permettant d'appeler ces services à partir d'une fonction Lambda.

Pour configurer le rôle

1. Connectez-vous à la console IAM Console de gestion AWS et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Dans le panneau de navigation, choisissez Rôles, puis Créer un rôle.
3. Choisissez Service AWS , puis choisissez Lambda.
4. Choisissez l'onglet Permissions (Autorisations).
5. Recherchez AWSLambdaBasicExecutionRole.
6. Choisissez Balises suivantes.
7. Choisissez Examiner.
8. Nommez le rôle lambda-support.
9. Choisissez Créer un rôle.
10. Choisissez lambda-support pour afficher la page de présentation.
11. Choisissez Attach Policies (Attacher des politiques).
12. AmazonRekognitionFullAccessChoisissez dans la liste des politiques.
13. Choisissez Attach policy (Attacher la politique).
14. Recherchez AmazonS3FullAccess, puis choisissez Attacher une politique.

Création du projet

Créez un nouveau projet Java, puis configurez le fichier Maven pom.xml avec les paramètres et dépendances requis. Assurez-vous que votre fichier pom.xml ressemble à ce qui suit :

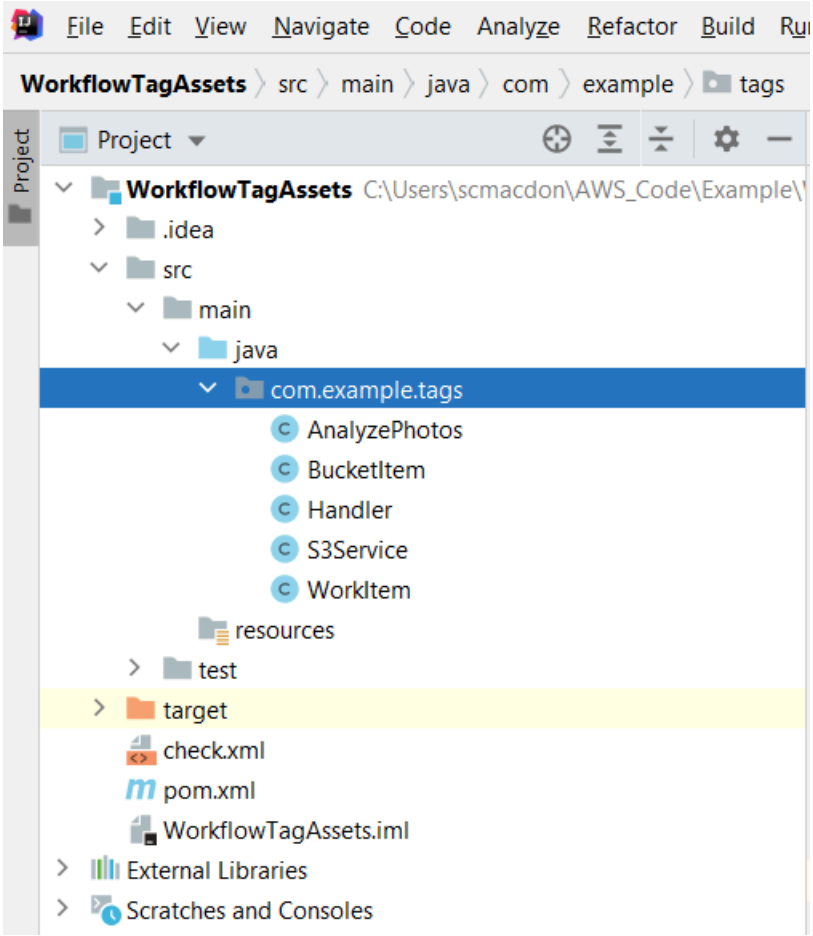
```
<?xml version="1.0" encoding="UTF-8"?>
 <project xmlns="http://maven.apache.org/POM/4.0.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>org.example</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>
```

Ecriture du code

Utilisez l'API Java AWS Lambda d'exécution pour créer la classe Java qui définit la fonction Lambda. Dans cet exemple, il existe une classe Java pour la fonction Lambda nommée Handler et des classes supplémentaires sont requises pour ce cas d’utilisation. L’illustration suivante représente les classes Java du projet. Notez que toutes les classes Java se trouvent dans un package nommé com.example.tags.



Créez les classes Java suivantes pour le code :

- Handler utilise l'API d'exécution Lambda Java et exécute le cas d'utilisation décrit dans ce didacticiel. AWS La logique de l'application exécutée se trouve dans la méthode handleRequest.
- S3Service utilise l'API Amazon S3 pour effectuer des opérations S3.
- AnalyzePhotosutilise l'API Amazon Rekognition pour analyser les images.
- BucketItemdéfinit un modèle qui stocke les informations du compartiment Amazon S3.
- WorkItemdéfinit un modèle qui stocke les données Amazon Rekognition.

Classe de gestionnaire :

Ce code Java représente la classe Gestionnaire. La classe lit un drapeau qui est transmis à la fonction Lambda. La Service.ListBucketObjects méthode s3 renvoie un objet List où chaque élément est une valeur de chaîne représentant la clé de l'objet. Si la valeur de l'indicateur est vraie, les balises sont appliquées en parcourant la liste et en appliquant des balises à chaque

objet en appelant la Service.tagAssets méthode s3. Si la valeur de l'indicateur est fausse, la Service.deleteTagFromObject méthode s3 est invoquée pour supprimer les balises. Notez également que vous pouvez enregistrer des messages dans Amazon CloudWatch Logs à l'aide d'un LambdaLoggerobjet.

> **Note**
>
> Assurez-vous d'attribuer le nom de votre compartiment à la variable bucketName.

```
package com.example.tags;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

public class Handler implements RequestHandler<Map<String,String>, String> {

@Override
public String handleRequest(Map<String, String> event, Context context) {
    LambdaLogger logger = context.getLogger();
    String delFlag = event.get("flag");
    logger.log("FLAG IS: " + delFlag);
    S3Service s3Service = new S3Service();
    AnalyzePhotos photos = new AnalyzePhotos();

    String bucketName = "<Enter your bucket name>";
    List<String> myKeys = s3Service.listBucketObjects(bucketName);
    if (delFlag.compareTo("true") == 0) {

        // Create a List to store the data.
        List<ArrayList<WorkItem>> myList = new ArrayList<>();

        // loop through each element in the List and tag the assets.
        for (String key : myKeys) {

            byte[] keyData = s3Service.getObjectBytes(bucketName, key);

            // Analyze the photo and return a list where each element is a WorkItem.
```

```
            ArrayList<WorkItem> item = photos.detectLabels(keyData, key);
            myList.add(item);
        }

        s3Service.tagAssets(myList, bucketName);
        logger.log("All Assets in the bucket are tagged!");

    } else {

        // Delete all object tags.
        for (String key : myKeys) {
            s3Service.deleteTagFromObject(bucketName, key);
            logger.log("All Assets in the bucket are deleted!");
        }
     }
    return delFlag;
  }
 }
```

Classe S3Service

La classe suivante utilise l'API Amazon S3 pour effectuer des opérations S3. Par exemple, la ObjectBytes méthode get renvoie un tableau d'octets qui représente l'image. De même, la BucketObjects méthode list renvoie un objet List où chaque élément est une valeur de chaîne spécifiant le nom de la clé.

```
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

Le code Java suivant représente la AnalyzePhotosclasse. Cette classe utilise l'API Amazon Rekognition pour analyser les images.

```
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

Le code Java suivant représente la BucketItemclasse qui stocke les données des objets 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

Le code Java suivant représente la WorkItemclasse.

```
 package com.example.tags;

 public class WorkItem {

 private String key;
 private String name;
 private String confidence ;

public void setKey (String key) {
    this.key = key;
}

public String getKey() {
    return this.key;
}

public void setName (String name) {
```

```
    this.name = name;
}

public String getName() {
    return this.name;
}

public void setConfidence (String confidence) {
    this.confidence = confidence;
}

public String getConfidence() {
    return this.confidence;
}

}
```

Package du projet

Empaquetez le projet dans un fichier .jar (JAR) à l'aide de la commande Maven suivante.

```
mvn package
```

Le fichier JAR se trouve dans le dossier cible (qui est un dossier enfant du dossier du projet).

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**
>
> Notez l'utilisation du maven-shade-plugin dans le fichier POM du projet. Ce plugin est chargé de créer un fichier JAR contenant les dépendances requises. Si vous essayez d'empaqueter le projet sans ce plugin, les dépendances requises ne sont pas incluses dans le fichier JAR et vous rencontrerez un ClassNotFoundException.

Déployez la fonction Lambda.

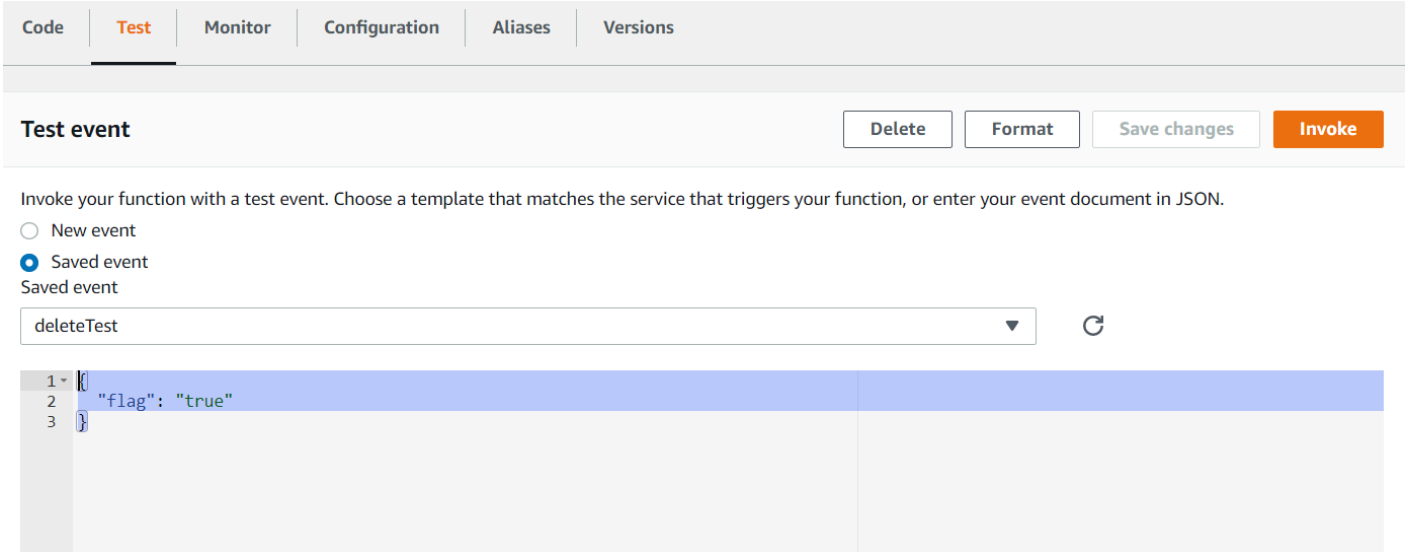
1. Ouvrez la console Lambda.
2. Choisissez Créer une fonction.
3. Choisissez Créer à partir de zéro.
4. Dans la section Informations de base, entrez cron comme nom.
5. Dans Exécution, choisissez Java 8.
6. Choisissez Utiliser un rôle existant, puis le rôle service Lambda (le rôle IAM que vous avez créé).
7. Choisissez Créer une fonction.
8. Dans Type d'entrée de code, choisissez Charger un fichier .zip ou .jar.
9. Choisissez Charger, puis naviguez jusqu'au fichier JAR que vous avez créé.
10. Pour Handler, entrez le nom complet de la fonction, par exemple com.example.tags.Handler:handleRequest(com.example.tags indique le package, Handler est la classe suivie de : : et du nom de la méthode).
11. Choisissez Enregistrer.

Test de la méthode Lambda

À ce stade du didacticiel, vous pouvez tester la fonction Lambda.

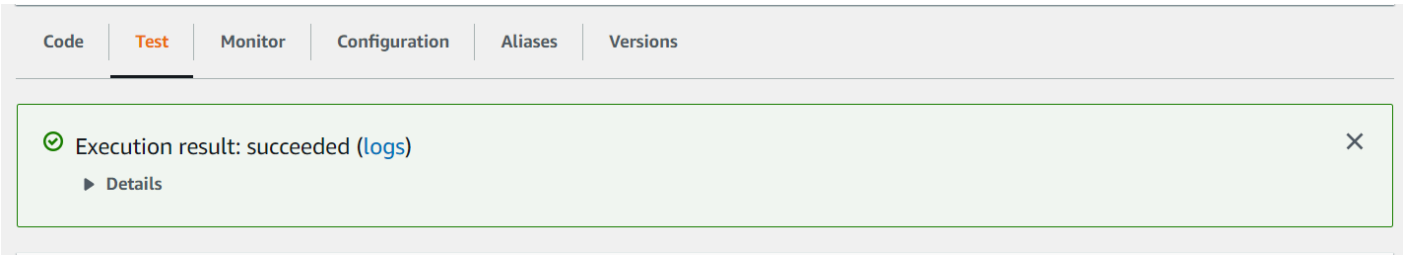
1. Dans la console Lambda, cliquez sur l'onglet Test, puis entrez le code JSON suivant.

```
                {
"flag": "true"
 }
```



> **Note**
>
> Transmission des balises true aux actifs numériques et transmission false supprime les balises.

2. Choisissez le bouton Invoquer. Une fois la fonction Lambda invoquée, vous voyez un message de réussite.



Félicitations, vous avez créé une AWS Lambda fonction qui applique automatiquement des balises aux actifs numériques situés dans un compartiment Amazon S3. Comme indiqué au début de ce didacticiel, veillez à supprimer toutes les ressources que vous avez créées au cours de ce didacticiel pour vous assurer que vous n'êtes pas facturé.

Pour d'autres exemples AWS multiservices, consultez le référentiel d'exemples GitHub du SDK de AWS documentation.

Création AWS applications d'analyse vidéo

Vous pouvez créer une application Web Java qui analyse les vidéos pour détecter les étiquettes à l'aide du AWS SDK pour Java version 2. L'application créée dans ce AWS didacticiel vous permet de télécharger une vidéo (fichier MP4) dans un compartiment Amazon S3. L'application utilise ensuite le service Amazon Rekognition pour analyser la vidéo. Les résultats sont utilisés pour remplir un modèle de données, puis un rapport est généré et envoyé par e-mail à un utilisateur spécifique à l'aide d'Amazon Simple Email Service.

L'illustration suivante montre un rapport généré une fois que l'application a terminé d'analyser la vidéo. Les colonnes du tableau ci-dessous indiquent la tranche d'âge, la barbe, les lunettes et les yeux ouverts, ainsi que les valeurs de confiance pour les différentes prédictions d'attributs.

	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				

Dans ce didacticiel, vous allez créer une application Spring Boot qui invoque divers AWS services. Les API Spring Boot sont utilisées pour créer un modèle, différentes vues et un contrôleur. Pour de plus amples informations, veuillez consulter Spring Boot.

Ce service utilise les AWS services suivants :

- Amazon Rekognition
- Amazon S3
- Amazon SES
- AWS Elastic Beanstalk

Les AWS services inclus dans ce didacticiel sont inclus dans le niveau AWS gratuit. Nous vous recommandons de mettre fin à toutes les ressources que vous créez dans le didacticiel lorsque vous en avez fini avec elles afin d'éviter d'être facturé.

Conditions préalables

Avant de commencer, vous devez suivre les étapes décrites dans Configuration du AWS SDK for Java. Veillez à effectuer les opérations suivantes :

- Java 1.8 JDK.
- Maven 3.6 ou ultérieure.
- Un compartiment Amazon S3 nommé video [somevalue]. Assurez-vous d'utiliser ce nom de compartiment dans le code Java de votre Amazon S3. Pour de plus amples informations, veuillez consulter Créer un compartiment.
- Un rôle IAM. Vous en avez besoin pour la VideoDetectFacesclasse que vous allez créer. Pour de plus amples informations, veuillez consulter Configuration de Vidéo Amazon Rekognition.
- Une rubrique Amazon SNS valide. Vous en avez besoin pour la VideoDetectFacesclasse que vous allez créer. Pour de plus amples informations, veuillez consulter Configuration de Vidéo Amazon Rekognition.

Procédure

Au cours du didacticiel, vous effectuez les opérations suivantes :

1. Créer un projet
2. Ajouter des dépendances POM à votre projet
3. Créer des classes Java
4. Créer les fichiers HTML
5. Créer les fichiers de script
6. Emballer le projet dans un fichier JAR
7. Déployez l'application sur AWS Elastic Beanstalk

Pour poursuivre le didacticiel, suivez les instructions détaillées dans le GitHub référentiel d'exemples du SDK de AWS documentation.

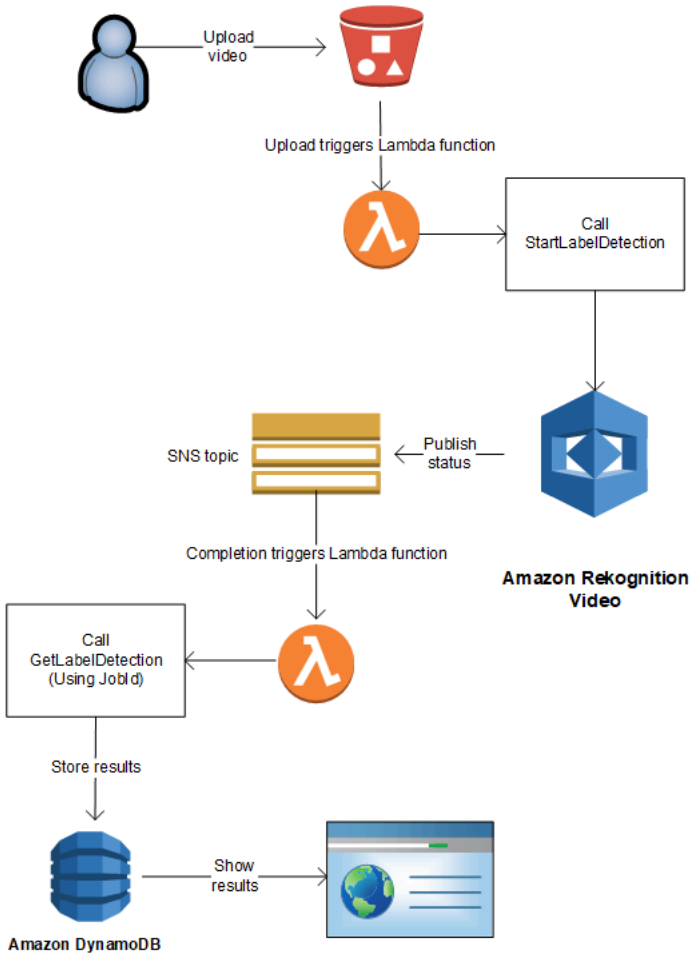
Création d’une fonction Lambda Amazon Rekognition

Ce didacticiel montre comment obtenir les résultats d’une opération d’analyse vidéo destinée à détecter des étiquettes à l’aide d’une fonction Lambda.

> **Note**
>
> Ce didacticiel utilise le AWS SDK pour Java 1.x. Pour un didacticiel utilisant Rekognition et le SDK pour AWS Java version 2, consultez le référentiel d'exemples du SDK de documentation.AWS GitHub

Vous pouvez utiliser les fonctions Lambda avec les opérations Vidéo Amazon Rekognition. Par exemple, le schéma suivant illustre un site web qui utilise une fonction Lambda pour lancer automatiquement l'analyse d'une vidéo au moment où celle-ci est chargée dans un compartiment Amazon S3. Lorsque la fonction Lambda est déclenchée, elle lance un appel StartLabelDetectionpour commencer à détecter les étiquettes dans la vidéo mise en ligne. Pour plus d'informations sur l'utilisation de la fonction Lambda pour traiter les notifications d'événements à partir d'un compartiment Amazon S3, consultez Utilisation d'AWS Lambda avec Amazon S3 Events.

Une deuxième fonction Lambda est déclenchée lorsque le statut d'achèvement de l'analyse est envoyé à la rubrique Amazon SNS enregistrée. La deuxième fonction Lambda appelle GetLabelDetectionpour obtenir les résultats de l'analyse. Les résultats sont ensuite stockés dans une base de données en vue d'être affichés sur une page web. Cette deuxième fonction lambda est l'objet de ce didacticiel.



Dans ce didacticiel, la fonction Lambda est déclenchée lorsque Vidéo Amazon Rekognition envoie le statut d'achèvement de l'analyse vidéo à la rubrique Amazon SNS enregistrée. Elle recueille ensuite les résultats de l'analyse vidéo en appelant GetLabelDetection. À des fins de démonstration, ce didacticiel enregistre les résultats de détection d'étiquettes dans un CloudWatch journal. Dans la fonction Lambda de votre application, vous avez tout intérêt à stocker les résultats d'analyse en vue d'une utilisation ultérieure. Par exemple, vous pouvez utiliser Amazon DynamoDB pour enregistrer les résultats d'analyse. Pour plus d'informations, consultez Utilisation de DynamoDB.

Les procédures suivantes vous montrent comment :

- Créer la rubrique Amazon SNS et configurer des autorisations.
- Créez la fonction Lambda à l'aide du Console de gestion AWS et abonnez-la à la rubrique Amazon SNS.
- Configurer la fonction Lambda à l'aide d' Console de gestion AWS.
- Ajoutez un exemple de code à un AWS Toolkit for Eclipse projet et chargez-le dans la fonction Lambda.

- Tester la fonction Lambda à l'aide de l' AWS CLI.

> **Note**
>
> Utilisez la même AWS région tout au long du didacticiel.

Conditions préalables

Ce didacticiel part du principe que vous maîtrisez AWS Toolkit for Eclipse. Pour plus d'informations, consultez AWS Toolkit pour Eclipse.

Création de la rubrique SNS

Le statut d'achèvement d'une opération d'analyse vidéo Vidéo Amazon Rekognition est envoyé à une rubrique Amazon SNS. Cette procédure crée la rubrique Amazon SNS, ainsi que la fonction de service IAM qui donne à Vidéo Amazon Rekognition l'accès à vos rubriques Amazon SNS. Pour de plus amples informations, veuillez consulter Appeler les opérations de Vidéo Amazon Rekognition.

Pour créer une rubrique Amazon SNS

1. Si vous ne l'avez pas encore fait, créez une fonction de service IAM pour donner à Vidéo Amazon Rekognition l'accès à vos rubriques Amazon SNS. Notez l'Amazon Resource Name (ARN). Pour de plus amples informations, veuillez consulter Accorder l'accès à plusieurs rubriques Amazon SNS.
2. Créez une rubrique Amazon SNS à l'aide de la console Amazon SNS. Vous devez uniquement spécifier le nom de la rubrique. Ajoutez le nom du sujet avec AmazonRekognition. Notez l'ARN de la rubrique.

Créer la fonction Lambda

Vous devez créer la fonction Lambda à l'aide d' Console de gestion AWS. Vous devez ensuite utiliser un projet AWS Toolkit for Eclipse pour charger le package de fonctions Lambda dans AWS Lambda. Il est également possible de créer la fonction Lambda avec AWS Toolkit for Eclipse. Pour plus d'informations, consultez Didacticiel : comment créer, charger et appeler une fonction AWS Lambda.

Pour créer la fonction Lambda

1. Connectez-vous à AWS Management Console et ouvrez la console AWS Lambda à l'adresse https://console.aws.amazon.com/lambda/.
2. Choisissez Créer une fonction.
3. Choisissez Créer à partir de zéro.
4. Dans Function name (Nom de la fonction), saisissez un nom pour votre fonction.
5. Dans Runtime (Exécution), choisissez Java 8.
6. Choisissez Choose or create an execution role (Choisir ou créer un rôle d'exécution).
7. Pour Execution role (Rôle d'exécution), choisissez Create a new role with basic Lambda permissions (Créer un nouveau rôle avec les autorisations Lambda de base).
8. Notez le nom du nouveau rôle affiché en bas de la rubrique Basic information (Informations de base).
9. Choisissez Créer une fonction.

Configurer la fonction Lambda

Une fois la fonction Lambda créée, vous devez la configurer de sorte qu'elle soit déclenchée par la rubrique Amazon SNS que vous créez dans Création de la rubrique SNS. Vous devez également ajuster les besoins en mémoire et le délai d'expiration de la fonction Lambda.

Pour configurer la fonction Lambda

1. Dans Function Code (Code de fonction), tapez `com.amazonaws.lambda.demo.JobCompletionHandler` pour Handler (Gestionnaire).
2. Dans Basic settings (Paramètres de base), choisissez Edit (Modifier). La boîte de dialogue Edit basic settings (Modifier les paramètres de base) s'affiche.
   a. Choisissez 1024 pour Memory (Mémoire).
   b. Choisissez 10 secondes pour Timeout (Expiration).
   c. Choisissez Enregistrer.
3. Sous Designer (Concepteur), choisissez + Add trigger (+ Ajouter un déclencheur). La boîte de dialogue Ajouter un déclencheur s'affiche.
4. Dans Trigger configuration (Configuration du déclencheur), choisissez SNS.

Dans Rubrique SNS, choisissez la rubrique Amazon SNS que vous avez créée dans Création de la rubrique SNS.

5. Choisissez Activer le déclencheur.
6. Pour ajouter le déclencheur, choisissez Ajouter.
7. Choisissez Enregistrer pour enregistrer la fonction Lambda.

Configuration du rôle IAM Lambda

Pour appeler les opérations Amazon Rekognition Video, vous ajoutez la politique gérée par AmazonRekognitionFullAccessAWS au rôle IAM Lambda. Les opérations de démarrage, telles que StartLabelDetection, nécessitent également des autorisations de transfert pour le rôle de service IAM utilisé par Amazon Rekognition Video pour accéder à la rubrique Amazon SNS.

Pour configurer le rôle

1. Connectez-vous à la console IAM Console de gestion AWS et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Dans le panneau de navigation, choisissez Rôles.
3. Dans la liste, choisissez le nom du rôle d'exécution que vous avez créé à l'étape Créer la fonction Lambda.
4. Choisissez l'onglet Permissions (Autorisations).
5. Choisissez Attacher des politiques.
6. AmazonRekognitionFullAccessChoisissez dans la liste des politiques.
7. Choisissez Attach policy (Attacher la politique).
8. Choisissez encore une fois le rôle d'exécution.
9. Sélectionnez Ajouter une politique en ligne.
10. Sélectionnez l'onglet JSON.
11. Remplacez la stratégie existante par la stratégie suivante. Remplacez `servicerole` par la fonction de service IAM que vous avez créée à l'étape Création de la rubrique SNS.
12. Choisissez Examiner une politique.
13. Dans Name* (Nom), attribuez un nom à la stratégie.
14. Choisissez Create Policy (Créer une politique).

Créez le AWS Toolkit for Eclipse Projet Lambda

Lorsque la fonction Lambda est déclenchée, le code suivant obtient l'état d'achèvement dans la rubrique Amazon SNS et GetLabelDetectionappelle pour obtenir les résultats de l'analyse. Le nombre d'étiquettes détectées et la liste des étiquettes détectées sont consignés dans un CloudWatch journal. Votre fonction Lambda doit stocker les résultats d'analyse vidéo en vue d'une utilisation ultérieure.

Pour créer le AWS Toolkit for Eclipse Projet Lambda

1. Créez un projet AWS Toolkit for EclipseAWS Lambda.
   - Pour Project name: (Nom du projet), tapez le nom de projet de votre choix.
   - Pour Nom de classe :, entrez JobCompletionHandler.
   - Pour Input type: (Type d'entrée), choisissez SNS Event (Événement SNS).
   - Laissez les autres champs en l'état.
2. Dans l'explorateur de projet Eclipse, ouvrez la méthode de gestion Lambda générée (JobCompletionHandler.java) et remplacez le contenu par le suivant :

```
//Copyright 2018 Amazon.com, Inc. or its 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. Les espaces de noms Rekognition ne sont pas résolus. Pour corriger cela :
   - Placez le curseur de votre souris sur la partie sous-lignée de la ligne `import com.amazonaws.services.rekognition.AmazonRekognition;`.
   - Choisissez Fix project set up... (Corriger la configuration du projet).
   - Choisissez la version la plus récente de l'archive Amazon Rekognition.
   - Choisissez OK pour ajouter l'archive au projet.
4. Enregistrez le fichier.
5. Right-click dans votre fenêtre de code Eclipse, choisissez AWS Lambda, puis choisissez Upload function to AWS Lambda.
6. Sur la page Select Target Lambda Function, choisissez la région AWS à utiliser.
7. Choisissez Choose an existing lambda function (Choisir une fonction Lambda existante), puis sélectionnez la fonction Lambda que vous avez créée à l'étape Créer la fonction Lambda.
8. Choisissez Suivant. La boîte de dialogue Function Configuration (Configuration de la fonction) s'affiche.
9. Dans IAM Role (Rôle IAM), choisissez le rôle IAM que vous avez créé à l'étape Créer la fonction Lambda.
10. Choisissez Terminer ; la fonction Lambda est alors chargée dans AWS.

Test de la fonction Lambda

Utilisez la AWS CLI commande suivante pour tester la fonction Lambda en démarrant l'analyse de détection des étiquettes d'une vidéo. Une fois l'analyse terminée, la fonction Lambda est déclenchée. Vérifiez que l'analyse a réussi en consultant CloudWatch les journaux.

Pour tester la fonction Lambda

1. Téléchargez un fichier MOV ou un fichier vidéo MPEG-4 formaté dans votre compartiment S3. À des fins de test, téléchargez une vidéo de 30 secondes au maximum.

   Pour en savoir plus, consultez Chargement d'objets dans Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

2. Exécutez la AWS CLI commande suivante pour commencer à détecter les étiquettes dans une vidéo.

   ```
   aws rekognition start-label-detection --video
    "S3Object={Bucket="bucketname",Name="videofile"}" \
   --notification-channel "SNSTopicArn=TopicARN,RoleArn=RoleARN" \
   --region Region
   ```

   Mettez à jour les valeurs suivantes :

   - Remplacez `bucketname` et `videofile` par le nom du compartiment Amazon S3 et le nom de fichier de la vidéo dans laquelle vous souhaitez détecter les étiquettes.
   - Remplacez `TopicARN` par l'ARN de la rubrique Amazon SNS que vous avez créée à l'étape Création de la rubrique SNS.
   - Remplacez `RoleARN` par l'ARN du rôle IAM que vous avez créé à l'étape Création de la rubrique SNS.
   - Passez `Region` à la AWS région que vous utilisez.

3. Notez la valeur de `JobId` dans la réponse. La réponse est semblable à l'exemple JSON suivant.

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

4. Ouvrez la console https://console.aws.amazon.com/cloudwatch/.

5. Une fois l'analyse terminée, une entrée de journal pour la fonction Lambda s'affiche dans le Groupe de journaux.
6. Choisissez la fonction Lambda pour afficher les flux de journaux.
7. Choisissez le flux de journal le plus récent pour voir les entrées de journal créées par la fonction Lambda. Si l'opération a réussi, cela ressemble à la sortie suivante, qui montre les détails de l'opération de reconnaissance vidéo, y compris l'identifiant de la tâche, le type d'opération StartLabelDetection « » et une liste des catégories d'étiquettes détectées telles que Bottle, Clothing, Crowd et Food :

Time (UTC +00:00)	Message
2018-02-28	
19:48:01	START RequestId: 47cb1472-1cc0-11e8-860a-4d00aa2ff96b Version: $LATEST
19:48:02	Rekognition Video Operation:==========
19:48:02	Job id: "9c7c3b1403a375a044c6dbe793d5c78d06014ee16f5efde083ad654b06f6c59a"
19:48:02	Status : "SUCCEEDED"
19:48:02	Job tag : null
19:48:02	Operation : "StartLabelDetection"
19:48:09	Total number of labels : 29
19:48:09	labels : Audience / Badge / Bottle / Clothing / Coat / Crowd / Electric Guitar / Flora / Food / Guitar / Hu
19:48:09	Result: {}
19:48:09	END RequestId: 47cb1472-1cc0-11e8-860a-4d00aa2ff96b
19:48:09	REPORT RequestId: 47cb1472-1cc0-11e8-860a-4d00aa2ff96b Duration: 8036.70 ms Billed Duration:

La valeur de Jod id (ID de tâche) doit correspondre à la valeur de `JobId` que vous avez notée à l'étape 3.

Utilisation d'Amazon Rekognition pour la vérification d'identité

Amazon Rekognition propose aux utilisateurs plusieurs opérations qui permettent de créer facilement des systèmes de vérification d'identité. Amazon Rekognition permet à l'utilisateur de détecter des visages sur une image, puis de les comparer à d'autres visages en confrontant les données faciales. Ces données faciales sont stockées dans des conteneurs côté serveur appelés collections. En utilisant les opérations de détection des visages, de comparaison des visages et de gestion des collections d'Amazon Rekognition, vous pouvez créer une application avec une solution de vérification d'identité.

Ce didacticiel présente deux flux de travail courants pour la création d'applications nécessitant une vérification d'identité.

Le premier flux de travail implique l'enregistrement d'un nouvel utilisateur dans une collection. Le deuxième flux de travail consiste à rechercher une collection existante dans le but de connecter un utilisateur récurrent.

Vous allez utiliser l'AWS SDK pour Python dans ce didacticiel. Vous pouvez également consulter le GitHub référentiel d'exemples du SDK de AWS documentation pour d'autres didacticiels Python.

Rubriques

- Conditions préalables
- Création d'une collection
- Enregistrement d'un nouvel utilisateur
- Connexion utilisateur existante

Conditions préalables

Avant de commencer ce didacticiel, vous devez installer Python et suivre les étapes requises pour configurer le AWS SDK Python. En outre, vérifiez que vous avez effectué les opérations suivantes :

- Création d'un compte AWS et d'un rôle IAM
- Installation de SDK Python (Boto3)
- Configurez correctement vos informations d' AWS accès
- Création d'un compartiment Amazon Simple Storage Service et téléchargement d'une image que vous souhaitez utiliser comme identifiant à des fins de vérification d'identité.
- Sélection d'une deuxième image servant d'image cible pour la vérification d'identité.

Création d'une collection

Avant de pouvoir enregistrer ou rechercher un nouvel utilisateur dans une collection, vous devez disposer d'une collection avec laquelle travailler. Une collection Amazon Rekognition est un conteneur côté serveur qui sert à stocker des informations sur les visages détectés.

Création de la collection

Vous allez commencer par écrire une fonction qui crée une collection qu'utilisera votre application. Amazon Rekognition stocke des informations sur les visages détectés dans des conteneurs côté serveur appelés collections. Vous pouvez rechercher des informations faciales stockées dans une collection pour trouver des visages connus. Pour stocker des informations faciales, vous devez d'abord créer une collection à l'aide de l'opération `CreateCollection`.

1. Sélectionnez un nom pour la collection que vous souhaitez créer. Dans le code suivant, remplacez la valeur de `collection_id` par le nom de la collection que vous souhaitez créer, et remplacez la valeur de `region` par le nom de la région définie dans vos informations d'identification utilisateur. Vous pouvez utiliser l'argument `Tags` pour appliquer les balises de votre choix à la collection, bien que cela ne soit pas obligatoire. L'opération `CreateCollection` renvoie des informations sur la collection que vous avez créée, y compris l'ARN de la collection. Notez l'ARN que vous recevez à la suite de l'exécution du code.

```
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. Enregistrez et exécutez le code. Copiez l'ARN de la collection.

   Maintenant que la collection Rekognition a été créée, vous pouvez y stocker des informations faciales et des identifiants. Vous allez également pouvoir comparer les visages aux informations enregistrées à des fins de vérification.

Enregistrement d'un nouvel utilisateur

Vous devez être en mesure d'enregistrer de nouveaux utilisateurs et d'ajouter leurs informations à une collection. Le processus d'enregistrement d'un nouvel utilisateur implique généralement les étapes suivantes :

Appelez l'opération **`DetectFaces`**

Écrivez le code pour vérifier la qualité de l'image du visage via l'opération `DetectFaces`. Vous allez utiliser cette opération `DetectFaces` pour déterminer si une image capturée par l'appareil photo peut être traitée par l'opération `SearchFacesByImage`. L'image ne doit contenir qu'un seul visage. Vous fournissez un fichier image d'entrée local à l'opération `DetectFaces` et recevez des informations sur les visages détectés sur l'image. L'exemple de code suivant fournit l'image d'entrée à `DetectFaces` puis vérifie si un seul visage a été détecté dans l'image.

1. Dans l'exemple de code suivant, remplacez `photo` par le nom de l'image cible dans laquelle vous souhaitez détecter des visages. Vous devez aussi remplacer la valeur de `region` par le nom de la région d'exploitation associée à votre compte.

```
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. Enregistrez, puis exécutez le code suivant.

Appelez l'opération **`CompareFaces`**.

Votre application doit être en mesure d'enregistrer de nouveaux utilisateurs dans une collection et de confirmer l'identité des utilisateurs récurrents. Vous allez d'abord créer les fonctions qui servent à enregistrer un nouvel utilisateur. Vous allez commencer par utiliser cette `CompareFaces` opération pour comparer une input/target image locale de l'utilisateur et une ID/stored image. S'il existe une correspondance entre le visage détecté sur les deux images, vous pouvez effectuer une recherche dans la collection pour voir si l'utilisateur y est enregistré.

Commencez par écrire une fonction qui compare une image d'entrée à l'image d'identification que vous avez stockée dans votre compartiment Amazon S3. Dans l'exemple de code suivant, vous devez fournir vous-même l'image d'entrée, qui doit être capturée après l'utilisation d'un type de détecteur de vivacité. Vous devez également transmettre le nom d'une image stockée dans votre compartiment Amazon S3.

1. Remplacez la valeur de `bucket` par le nom du compartiment Amazon S3 qui contient votre fichier source. Vous devez également remplacer la valeur de `source_file` par le nom de l'image source que vous utilisez. Remplacez la valeur de `target_file` par le nom du fichier cible que vous avez fourni. Remplacez la valeur de `region` par le nom de la `region` définie dans vos informations d'identification utilisateur.

   Vous devez également spécifier le niveau minimal de fiabilité de la correspondance que vous souhaitez voir incluse dans la réponse à l'aide de l'argument `similarityThreshold`. Les visages détectés ne sont renvoyés dans le tableau `FaceMatches` que si le niveau de confiance est supérieur à ce seuil. Votre choix `similarityThreshold` doit refléter la nature de votre cas d'utilisation spécifique. Tout cas d'utilisation impliquant des applications de sécurité critiques doit utiliser 99 comme seuil sélectionné.

```
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. Enregistrez, puis exécutez le code suivant.

   Vous recevez un objet de réponse contenant des informations sur le visage correspondant et le niveau de confiance.

Appelez l'opération **`SearchFacesByImage`**.

Si le niveau de confiance de l'opération `CompareFaces` est supérieur au `SimilarityThreshold` que vous avez choisi, vous devez rechercher dans votre collection un visage susceptible de correspondre à l'image d'entrée. Si une correspondance est trouvée dans votre collection, cela signifie que l'utilisateur est probablement déjà enregistré dans la collection, et qu'il n'est pas nécessaire d'enregistrer un nouvel utilisateur dans votre collection. S'il n'y a pas de correspondance, vous pouvez enregistrer le nouvel utilisateur dans votre collection.

1. Commencez par écrire le code invoquant l'opération `SearchFacesByImage`. L'opération prend un fichier image local comme argument, puis recherche dans votre `Collection` un visage qui correspond aux plus grands visages détectés dans l'image fournie.

   Dans l'exemple de code suivant, modifiez la valeur `collectionId` de la collection où vous souhaitez faire la recherche. Remplacez la valeur de `region` par le nom de la région d'exploitation associée à votre compte. Vous devez également remplacer la valeur de `photo` par le nom de votre fichier d'entrée. Vous souhaiterez également spécifier un seuil de similarité en remplaçant la valeur de `threshold` par un percentile choisi.

```
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. Enregistrez, puis exécutez le code suivant. S'il y a une correspondance, cela signifie que la personne reconnue sur l'image fait déjà partie de la collection, et qu'il n'est pas nécessaire de passer aux étapes suivantes. Dans ce cas, vous pouvez simplement autoriser l'utilisateur à accéder à l'application.

Appelez l'opération **`IndexFaces`**.

En supposant qu'aucune correspondance n'ait été trouvée dans la collection que vous recherchez, vous souhaitez ajouter le visage de l'utilisateur à votre collection. Vous pouvez le faire en appelant l'opération `IndexFaces`. Lorsque vous appelez IndexFaces, Amazon Rekognition extrait les traits du visage d'un visage identifié sur votre image d'entrée et stocke les données dans la collection spécifiée.

1. Commencez par écrire le code à appeler `IndexFaces`. Remplacez la valeur `image` de par le nom du fichier local que vous souhaitez utiliser comme image d'entrée pour l' IndexFaces opération. Vous devrez également remplacer la valeur de `photo_name` par le nom souhaité pour votre image d'entrée. Remplacez la valeur de `collection_id` par l'ID de la collection que vous avez créée précédemment. Enfin, remplacez la valeur de `region` par le nom de la région d'exploitation associée à votre compte. Vous avez également besoin de spécifier une valeur pour le paramètre `MaxFaces` d'entrée, qui définit le nombre maximum de visages à indexer dans une image. La valeur par défaut de ce paramètre est 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. Enregistrez, puis exécutez le code suivant. Déterminez si vous souhaitez enregistrer les données renvoyées par l'opération `IndexFaces`, telles que le FaceID attribué à la personne sur l'image. La section suivante examine comment enregistrer ces données. Copiez les valeurs renvoyées `FaceId`, `ImageId` et `Confidence` avant de continuer.

Stockez des données d'image et d'identification faciale dans Amazon S3 et Amazon DynamoDB

Une fois que l'identifiant facial de l'image d'entrée a été obtenu, les données de l'image peuvent être enregistrées dans Amazon S3, tandis que les données faciales et l'URL de l'image peuvent être saisies dans une base de données telle que DynamoDB.

1. Écrivez le code pour charger l'image d'entrée dans votre base de données Amazon S3. Dans l'exemple de code suivant, remplacez la valeur de `bucket` par le nom du compartiment dans lequel vous souhaitez télécharger le fichier, puis remplacez la valeur de `file_name` par le nom

du fichier local que vous souhaitez stocker dans votre compartiment Amazon S3. Fournissez un nom de clé qui permettra d'identifier le fichier dans le compartiment Amazon S3 en remplaçant la valeur de `key_name` par le nom que vous souhaitez donner au fichier image. Le fichier que vous souhaitez télécharger est le même que celui défini dans les exemples de code précédents, à savoir le fichier d'entrée que vous avez utilisé IndexFaces. Enfin, remplacez la valeur de `region` par le nom de la région d'exploitation associée à votre compte.

```
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. Enregistrez et exécutez l'exemple de code suivant pour télécharger votre image d'entrée sur Amazon S3.
3. Vous devez également enregistrer l'identifiant facial renvoyé dans une base de données. Pour ce faire, créez un tableau de base de données DynamoDB et téléchargez l'identifiant facial dans ce tableau. L'exemple de code suivant crée un tableau DynamoDB. Notez que vous ne devez exécuter qu'une seule fois le code qui crée ce tableau. Enfin, remplacez la valeur de `region` par le nom de la région d'exploitation associée à votre compte. Vous devrez également remplacer la valeur de `database_name` par le nom que vous souhaitez attribuer au tableau 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. Enregistrez et exécutez le code suivant pour créer votre tableau.
5. Après avoir créé la table, vous pouvez y télécharger le FaceId contenu renvoyé. Pour ce faire, vous allez établir une connexion au tableau à l'aide de la fonction Tableau, puis utiliser cette fonction `put_item` pour télécharger les données.

   Dans l'exemple de code suivant, remplacez la valeur de `bucket` par le nom du compartiment contenant l'image d'entrée que vous avez chargée sur Amazon S3. Vous devez également remplacer la valeur de `file_name` par le nom du fichier d'entrée que vous avez chargé dans votre compartiment Amazon S3, et la valeur de `key_name` par la clé que vous avez précédemment utilisée pour identifier le fichier d'entrée. Enfin, remplacez la valeur de `region` par le nom de la région d'exploitation associée à votre compte. Ces valeurs doivent correspondre à celles fournies à l'étape 1.

   `AddDBEntry`Stocke les valeurs FaceId ImageId,, et Confidence attribuées à un visage dans une collection. Fournissez à la fonction ci-dessous les valeurs que vous avez enregistrées à l'étape 2 de la section `IndexFaces` suivante.

```
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. Enregistrez et exécutez l’exemple de code suivant pour stocker les données d’identifiant facial renvoyées dans un tableau.

Connexion utilisateur existante

Une fois qu’un utilisateur a été enregistré dans une collection, il peut être authentifié à son retour en utilisant l’opération `SearchFacesByImage`. Vous devrez obtenir une image d’entrée, puis vérifier la qualité de l’image d’entrée à l’aide de `DetectFaces`. Cela permet de déterminer si une image appropriée a été utilisée avant d’exécuter l’opération `SearchFacesbyImage`.

Appelez l' DetectFaces opération

1. Vous allez utiliser cette opération `DetectFaces` pour vérifier la qualité de l'image du visage et déterminer si une image capturée par l'appareil photo peut être traitée par l'opération `SearchFacesByImage`. L'image d'entrée ne doit contenir qu'un seul visage. L'exemple de code suivant prend une image d'entrée et la fournit à l'opération `DetectFaces`.

   Dans l'exemple de code suivant, remplacez la valeur de `photo` par le nom de l'image cible locale, et remplacez la valeur de `region` par le nom de la région associée à votre compte.

```
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. Enregistrez et exécutez le code.

Appelez l' SearchFacesByImage opération

1. Écrivez le code pour comparer le visage détecté aux visages de la collection avec le `SearchFacesByImage`. Vous allez utiliser le code affiché dans la section « Enregistrement d'un nouvel utilisateur » et fournir l'image d'entrée pour l'opération `SearchFacesByImage`.

   Dans l'exemple de code suivant, remplacez la valeur de `collectionId` par la collection où vous souhaitez effectuer la recherche. Vous allez également remplacer la valeur de `bucket` par le nom d'un compartiment Amazon S3, et la valeur de `fileName` par un fichier image dans ce compartiment. Enfin, remplacez la valeur de `region` par le nom de la région d'exploitation associée à votre compte. Vous allez également devoir spécifier un seuil de similarité en remplaçant la valeur de `threshold` par un percentile choisi.

```
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. Enregistrez et exécutez le code.

Vérifiez l'identifiant facial renvoyé et le niveau de confiance

Vous pouvez désormais vérifier les informations relatives à la correspondance FaceId en imprimant des éléments de réponse tels que FaceId les attributs Similarité et Confiance.

```
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']))
```

Utilisation d'Amazon Rekognition (REK) pour détecter les étiquettes destinées aux applications marketing

Ce didacticiel vous explique comment créer un exemple d'application Python qui pourrait être utilisée pour envoyer des e-mails à des personnes, sur la base d'images téléchargées sur un site Web. L'exemple d'application est conçu pour impliquer les utilisateurs dans une campagne de marketing sur les réseaux sociaux en leur envoyant des e-mails personnalisés sur les offres de voyage si des points de repère sont reconnus sur leurs photos.

La solution intègre différents AWS services, notamment Amazon Rekognition, Amazon S3 CloudWatch, DynamoDB et Amazon SES. L'application utilise Amazon Rekognition pour détecter les étiquettes sur les images téléchargées par des comptes reconnus dans une base de données DynamoDB, puis le titulaire du compte reçoit un e-mail marketing basé sur les étiquettes détectées dans l'image. L'architecture complète de la solution est la suivante :

- Stockez les données utilisateur dans la base de données DynamoDB.
- L'utilisateur télécharge les données d'image et les métadonnées (numéro de compte utilisateur) sur Amazon S3.
- Une fonction Lambda permet d'identifier et DetectLabelsd'enregistrer les libellés de l'image téléchargée, de rechercher l'adresse e-mail de l'utilisateur dans DynamoDB et d'envoyer un e-mail marketing à l'utilisateur qui a téléchargé l'image.
- Les résultats de la fonction Lambda sont enregistrés CloudWatch pour un examen ultérieur.

Voici un aperçu de toutes les étapes du didacticiel :

1. Créez et renseignez une table DynamoDB.
2. Écrivez une fonction Lambda avec Logging et Notification.
3. Configurez la fonction Lambda et les autorisations.
4. Téléchargez des images et des métadonnées avec Amazon S3.
5. CloudWatch Journaux de sondages.

Rubriques

- Conditions préalables
- Création et remplissage d'une table DynamoDB
- Création d'une fonction Lambda avec journalisation et notification
- Configuration des autorisations et de la fonction Lambda
- Chargement d'images vers Amazon S3 avec des métadonnées
- CloudWatch Journaux de sondage fournis par le client

Conditions préalables

Avant de commencer ce didacticiel, vous aurez besoin des éléments suivants :

- Un AWS compte et les autorisations IAM appropriées.
- Python et Boto3 installés sur votre environnement de développement.
- Compréhension de base de Lambda, Amazon S3, DynamoDB, Amazon SES et. CloudWatch
- Connaissance de base de Lambda et d'Amazon SES.

Création et remplissage d'une table DynamoDB

Pour commencer ce didacticiel, nous allons créer une table DynamoDB pour stocker les données des clients telles que l'adresse e-mail, l'âge, le numéro de téléphone et le statut d'adhésion, en utilisant une variable AccountNumber « » comme clé primaire. Nous allons également insérer des exemples de données dans le tableau.

Création de la table DynamoDB

Nous allons d'abord créer la table DynamoDB. Voici comment le configurer à l'aide de Python et Boto3. Nous créons une fonction appelée `create_user_table` et nous nous connectons à la ressource DynamoDB qu'elle contient. Dans l'exemple de code ci-dessous, remplacez la valeur de « region_name » par le nom de la région dans laquelle votre compte opère, puis exécutez la cellule de code pour créer votre tableau.

```
import boto3
def create_user_table(dynamodb=None):
    if not dynamodb:
        dynamodb = boto3.resource('dynamodb', region_name='us-east-1')

    table = dynamodb.create_table(
        TableName='CustomerDataTable',
        KeySchema=[
            {
                'AttributeName': 'AccountNumber',
                'KeyType': 'HASH'  # Partition key
            },
        ],
        AttributeDefinitions=[
            {
                'AttributeName': 'AccountNumber',
                'AttributeType': 'S'
            },
        ],
        ProvisionedThroughput={
            'ReadCapacityUnits': 10,
            'WriteCapacityUnits': 10
        }
    )

    # Wait until the table exists.
    table.wait_until_exists()
    print("Table status:", table.table_status)

# Create the DynamoDB table.
create_user_table()
```

L'exécution de ce script a permis de configurer une table DynamoDB `CustomerDataTable` nommée `AccountNumber` avec comme clé primaire.

Insertion d'échantillons de données

Nous allons maintenant insérer des exemples de données dans le tableau. Ces exemples de données nous aideront à tester toutes les fonctionnalités de l'application.

Nous allons créer une deuxième fonction qui ajoutera des exemples de données à celles que `CustomerDataTable` nous avons créées précédemment. Trois exemples d'entrées sont créés à l'aide du code ci-dessous, et chaque entrée inclut un numéro de compte, une adresse e-mail, un âge, un numéro de téléphone et un statut de membre. Dans l'exemple de code ci-dessous, remplacez la valeur de `region_name` par le nom de la région dans laquelle votre compte opère, puis exécutez la cellule de code pour créer votre tableau. Si vous souhaitez tester la partie de l'application dédiée à la livraison par e-mail, remplacez la valeur de `EmailAddress` la première entrée client ci-dessous par une adresse e-mail à laquelle vous pouvez recevoir des e-mails. Enregistrez et exécutez le code.

```
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()
```

Une fois la table DynamoDB configurée et remplie, nous pouvons maintenant intégrer cette récupération de données dans la fonction Lambda. Cela permet à notre application de récupérer les informations de l'utilisateur en fonction du numéro de compte qui sera identifié dans le prochain flux de traitement d'image.

Création d'une fonction Lambda avec journalisation et notification

Nous pouvons maintenant créer une fonction Lambda. Nous voulons nous assurer que la fonction Lambda, qui est déclenchée lors du téléchargement de l'image, peut lire à la fois les données d'image et les métadonnées du compte et les utiliser pour effectuer une recherche dans la table DynamoDB pour les données utilisateur associées. Cela signifie que nous devrons non seulement invoquer la fonction d'Amazon Rekognition, mais également une DetectLabels fonction qui l'intègre AccountNumber et l'utilise pour récupérer l'adresse e-mail associée dans DynamoDB. De plus, nous avons besoin d'une fonction qui envoie un e-mail à l'adresse e-mail via Amazon SES. Enfin, nous utilisons un enregistreur pour enregistrer les informations relatives au processus. Ces données seront affichées par CloudWatch.

Création de la fonction Lambda

Voici un aperçu de la manière dont une fonction Lambda peut répondre à ces exigences. Dans l'exemple de code suivant, assurez-vous que la table DynamoDB correcte est spécifiée, si vous avez utilisé un autre nom CustomerDataTable que le nom de votre table. De plus, dans la fonction « send_marketing_email », vous devez remplacer la valeur de l'argument « Source » par une adresse e-mail à laquelle vous avez accès et qui servira d'e-mail d'envoi.

```
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")
```

Maintenant que nous avons écrit la fonction Lambda, nous devons configurer les autorisations pour Lambda et créer une instance de notre fonction Lambda dans la console de gestion. AWS

Configuration des autorisations et de la fonction Lambda

Créer ou mettre à jour un rôle IAM

Avant de pouvoir utiliser une fonction Lambda pour gérer les images téléchargées par les utilisateurs, nous devons gérer les autorisations correspondantes. La fonction Lambda a besoin d'un rôle IAM défini et doté de politiques lui permettant d'interagir avec Amazon S3, DynamoDB, Amazon SES, Amazon Rekognition et. CloudWatch

Pour configurer votre rôle IAM pour la fonction Lambda :

1. Ouvrez AWS la console de gestion.
2. Accédez à IAM > Rôles > Créer un rôle.

   Choisissez « Créer un rôle ».
3. Sélectionnez Lambda comme service qui utilisera ce rôle. Cliquez sur « Suivant ».
4. Joignez les politiques suivantes (Notez que ces politiques sont sélectionnées uniquement à des fins de démonstration. Dans un environnement de production réel, vous souhaiteriez limiter les autorisations uniquement à celles qui sont nécessaires.) :

   - AmazonS3ReadOnlyAccess
   - AmazonRekognitionReadOnlyAccess
   - AmazonDynamoDBFullAccess
   - AmazonSESFullAccess
   - AWSLambdaExecute
   - AWSLambdaBasicExecutionRole (pour la CloudWatch journalisation)

   Cliquez sur « Suivant ».
5. Nommez le rôle et donnez-lui une description, puis créez-le en choisissant « Créer un rôle ».
6. Une fois que nous avons configuré les autorisations appropriées pour Lambda, nous pouvons créer une instance de la fonction Lambda à l'aide de la console de gestion. AWS

   Accédez au service Lambda dans l'AWS Management Console.
7. Cliquez sur « Créer une fonction ». Choisissez « Auteur à partir de zéro ».
8. Entrez le nom de la fonction et sélectionnez le moteur d'exécution Python. Sous « Modifier le rôle d'exécution par défaut », sélectionnez « Utiliser un rôle existant », puis choisissez le rôle IAM que vous avez créé précédemment.
9. Vous devez maintenant créer et mettre à jour le code dans l'onglet de la fonction Lambda. Accédez à l'onglet appelé « lambda_function » et remplacez le code par l'exemple de code Lambda suivant.

   Enregistrez les modifications et déployez-les.
10. Vous devez maintenant configurer un événement Amazon S3 en tant que déclencheur Lambda.

Dans la configuration de votre fonction Lambda tab/page, allez dans Déclencheur, cliquez sur « Ajouter un déclencheur ».

11. Sélectionnez Amazon S3 dans la liste des déclencheurs disponibles.
12. Pour configurer le déclencheur : sélectionnez le compartiment à partir duquel la fonction doit être déclenchée.

    Choisissez le type d'événement, PUT. Vous pouvez éventuellement spécifier un préfixe ou un suffixe si vous souhaitez traiter uniquement des fichiers portant certains noms ou types.
13. Activez le déclencheur en cliquant sur « Ajouter » et enregistrez la configuration.

Vérification des adresses e-mail Amazon SES

Avant de pouvoir utiliser Amazon SES pour envoyer des e-mails, vous devez vérifier les adresses e-mail de l'expéditeur et du destinataire. Pour cela :

1. Accédez à la console Amazon SES. Accédez à « Gestion des identités », puis à « Adresses e-mail ».
2. Cliquez sur « Vérifier une nouvelle adresse e-mail ». Entrez l'adresse e-mail que vous souhaitez vérifier, puis cliquez sur « Vérifier cette adresse e-mail ». Vous recevrez un e-mail contenant un lien de vérification. Cliquez sur le lien pour terminer le processus de vérification.
3. Après avoir vérifié les adresses e-mail des deux comptes, configuré votre fonction Lambda avec les autorisations appropriées et créé des exemples de données clients, vous pouvez tester la fonction Lambda en téléchargeant une image de test dans le compartiment Amazon S3 de votre choix.

Chargement d'images vers Amazon S3 avec des métadonnées

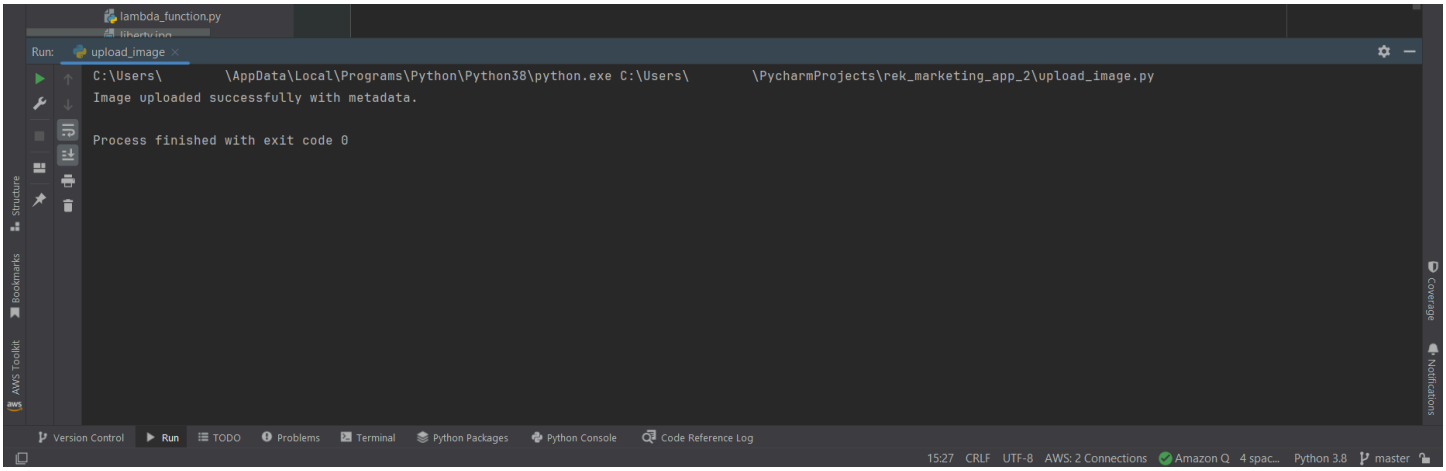
Nous pouvons désormais télécharger une image de test dans le compartiment Amazon S3 de notre choix à l'aide de la AWS console ou du script que nous avons préparé précédemment. Écrivez un script qui télécharge une image dans le compartiment Amazon S3 que vous avez précédemment spécifié dans votre fonction Lambda. Dans l'exemple de code ci-dessous, spécifiez le chemin de votre image comme premier argument de « upload_image_to_s3 », le nom du bucket comme deuxième argument et le numéro de compte de l'utilisateur qui télécharge l'image comme dernier argument.

```
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')
```

Dans cette fonction, le `ExtraArgs` paramètre de la `upload_fileobj` méthode est utilisé pour inclure des métadonnées définies par l'utilisateur (`account_number`) avec l'image. Ces métadonnées peuvent ensuite être utilisées AWS pour traiter l'image en conséquence.

Enregistrez et exécutez le script. Cela téléchargera l'image.



Quelques minutes après le téléchargement de l'image, vous devriez recevoir un e-mail à l'adresse que vous avez précédemment associée au compte indiqué ici.

CloudWatch Journaux de sondage fournis par le client

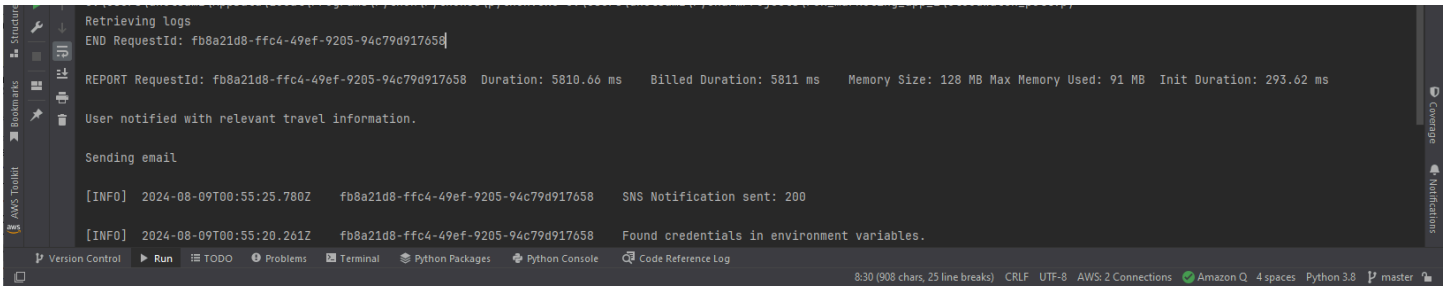
Consultez les CloudWatch journaux de votre fonction Lambda pour voir si elle a été déclenchée et exécutée comme prévu. Les journaux se trouvent sous CloudWatch > Journaux > Groupes de journaux >/AWS/lambda/your_function_name. Vous pouvez également écrire un script pour accéder aux journaux et les imprimer. L'exemple de code suivant interroge le groupe Lambda à la recherche de journaux et imprime les journaux générés au cours de la dernière heure. Enregistrez et exécutez le code.

```
import boto3
import time

def fetch_lambda_logs(log_group_name, start_time):
    client = boto3.client('logs')
    query = "fields @timestamp, @message | sort @timestamp desc | limit 20"
    start_query_response = client.start_query(
        logGroupName=log_group_name,
        startTime=int(start_time),
        endTime=int(time.time()),
        queryString=query,
    )
    query_id = start_query_response['queryId']
    response = None
    while response is None or response['status'] == 'Running':
        time.sleep(1)  # Wait for 1 second before checking the query status again
        response = client.get_query_results(queryId=query_id)
    return response['results']

# Usage
log_group = '/aws/lambda/RekMediaFunction'
logs = fetch_lambda_logs(log_group, int(time.time()) - 3600)  # Fetch logs from the
 last hour
print("Retrieving logs")
for log in logs:
    #print(log)
    print(log[1]['value'])
```

L'exécution du code devrait imprimer les journaux et vous devriez être en mesure de voir que l'utilisateur a été averti par e-mail contenant les informations de voyage pertinentes.



Vous avez créé avec succès une application capable de détecter les étiquettes dans les images téléchargées dans un compartiment Amazon S3, puis d'envoyer un message promotionnel à l'utilisateur qui a chargé l'image. Assurez-vous de supprimer toutes les ressources dont vous n'avez plus besoin afin qu'elles ne vous soient pas facturées.

Détection d’étiquettes dans une image à l’aide de Lambda et Python

AWS Lambda est un service de calcul que vous pouvez utiliser pour exécuter du code sans provisionner ni gérer de serveurs. Vous pouvez appeler les opérations de l’API Rekognition depuis une fonction Lambda. Les instructions suivantes montrent comment créer une fonction Lambda dans Python qui appelle `DetectLabels`.

La fonction Lambda appelle `DetectLabels` et renvoie un tableau d’étiquettes détectées dans l’image qui indique le niveau de fiabilité de la détection.

Les instructions incluent un exemple de code Python qui explique comment appeler la fonction Lambda et lui fournir une image provenant d’un compartiment Amazon S3 ou de votre ordinateur local.

Assurez-vous que les images que vous avez choisies respectent les limites de Rekognition. Consultez les directives et les quotas dans Rekognition et la référence de l'DetectLabels API pour plus d'informations sur le type de fichier image et les limites de taille.

Créer une fonction Lambda (console)

Au cours de cette étape, vous créez une fonction Lambda vide et un rôle d’exécution IAM qui permet à votre fonction Lambda d’appeler l’opération `DetectLabels`. Dans les étapes suivantes, vous ajoutez le code source, et en option une couche à la fonction Lambda.

Si vous utilisez des documents stockés dans un compartiment Amazon S3, cette étape explique également comment accorder l'accès au compartiment qui stocke vos documents.

Pour créer un AWS Lambda fonction (console)

1. Connectez-vous à Console de gestion AWS et ouvrez la AWS Lambda console à l'adresse https://console.aws.amazon.com/lambda/.
2. Choisissez Créer une fonction. Pour plus d'informations, consultez Créer une fonction Lambda à l'aide de la console.
3. Choisissez les options suivantes :
   - Choisissez Créer à partir de zéro.
   - Saisissez une valeur relative à Nom de fonction.
   - Pour Exécution, choisissez la version la plus récente de Python.
   - Pour Architecture, choisissez x86_64.
4. Choisissez Créer une fonction pour créer la fonction AWS Lambda .
5. Sur la page de la fonction, choisissez l'onglet Configuration.
6. Dans le volet Autorisations, sous Rôle d'exécution, choisissez le nom du rôle pour ouvrir le rôle dans la console IAM.
7. Sous l'onglet Autorisations, choisissez Ajouter des autorisations puis Créer une politique en ligne.
8. Choisissez l'onglet JSON et remplacez la politique par la politique suivante :

   JSON

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

9. Choisissez Examiner une politique.
10. Entrez un nom pour la politique, par exemple DetectLabels-access.
11. Choisissez Create Policy (Créer une politique).
12. Si vous stockez des documents à des fins d'analyse dans un compartiment Amazon S3, vous devez ajouter une stratégie d'accès Amazon S3. Pour ce faire, répétez les étapes 7 à 11 dans la console AWS Lambda et apportez les modifications suivantes.

    a. Pour l'étape 8, appliquez la politique suivante. *bucket/folder path*Remplacez-les par le chemin du compartiment et du dossier Amazon S3 vers les documents que vous souhaitez analyser.

       JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "S3Access",
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::bucket/folder path/*"
        }
    ]
}
```

    b. Pour l'étape 10, choisissez un autre nom de politique, tel que S3Bucket-access.

(Facultatif) Création d'une couche (console)

Il n'est pas nécessaire d'effectuer cette étape pour utiliser une fonction Lambda et un appel `DetectLabels`.

L'`DetectLabels`opération est incluse dans l'environnement Lambda Python par défaut dans le cadre du AWS SDK pour Python (Boto3).

Si d'autres parties de votre fonction Lambda nécessitent des mises à jour de AWS service récentes qui ne se trouvent pas dans l'environnement Lambda Python par défaut, vous pouvez effectuer cette étape pour ajouter la version la plus récente du SDK Boto3 en tant que couche à votre fonction.

Pour ajouter le SDK en tant que couche, vous devez d'abord créer une archive de fichier zip contenant le SDK Boto3. Ensuite, vous créez une couche et ajoutez l'archive du fichier zip à la couche. Pour plus d'informations, voir Utilisation de couches avec votre fonction Lambda.

Pour créer et ajouter une couche (console)

1. Ouvrez une invite de commande et saisissez les commandes suivantes pour créer un package de déploiement avec la version la plus récente du AWS SDK.

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

2. Notez le nom du fichier zip (boto3-layer.zip) que vous utilisez à l'étape 8 de cette procédure.
3. Ouvrez la AWS Lambda console à l'adresse https://console.aws.amazon.com/lambda/.
4. Choisissez Layers dans le volet de navigation.
5. Choisissez Créer un calque.
6. Saisissez un Name (Nom) et une Description pour la règle.
7. Dans Type d'entrée de code, choisissez Charger un fichier .zip et sélectionnez Charger.
8. Dans la boîte de dialogue, choisissez l'archive de fichiers zip (boto3-layer.zip) que vous avez créée à l'étape 1 de cette procédure.
9. Pour les exécutions compatibles, choisissez la version la plus récente de Python.
10. Choisissez Créer pour créer la couche.
11. Choisissez l'icône du menu du volet de navigation.
12. Dans le volet de navigation, choisissez Fonctions.
13. Dans la liste des ressources, choisissez la fonction que vous avez créée précédemment dans Créer une fonction Lambda (console).
14. Cliquez sur l'onglet Code.
15. Dans la zone Couches, choisissez Ajouter une couche.
16. Choisissez Couches personnalisées.
17. Dans Couches personnalisées, choisissez le nom de couche que vous avez saisi à l'étape 6.
18. Dans Version, choisissez la version de la couche, qui doit être 1.
19. Choisissez Ajouter.

Ajouter le code Python (console)

Au cours de cette étape, vous allez ajouter votre code Python à votre fonction Lambda via l'éditeur de code de la console Lambda. Le code détecte les étiquettes dans une image à l'aide de l'opération `DetectLabels`. Il renvoie un tableau d'étiquettes détectées dans l'image, ainsi que le niveau de confiance dans les étiquettes détectées.

Le document que vous fournissez pour l'opération `DetectLabels` peut se trouver dans un compartiment Amazon S3 ou sur un ordinateur local.

Pour ajouter le code Python (console)

1. Accédez à l'onglet Code.
2. Dans l'éditeur de code, remplacez le code dans lambda_function.py par le code suivant :

```
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. Pour déployer votre fonction Lambda, choisissez Deploy.

Pour ajouter le code Python (console)

Maintenant que vous avez créé votre fonction Lambda, vous pouvez l'invoquer pour détecter des étiquettes dans une image.

Au cours de cette étape, vous exécutez du code Python sur votre ordinateur, qui transmet une image locale ou une image d'un compartiment Amazon S3 à votre fonction Lambda.

Assurez-vous d'exécuter le code dans la même AWS région que celle dans laquelle vous avez créé la fonction Lambda. Vous pouvez afficher la AWS région de votre fonction Lambda dans la barre de navigation de la page des détails de la fonction dans la console Lambda.

Si la fonction Lambda renvoie une erreur d'expiration, prolongez le délai d'expiration de la fonction Lambda. Pour plus d'informations, consultez Configuration du délai d'expiration d'une fonction (console).

Pour plus d'informations sur l'appel d'une fonction Lambda depuis votre code, consultez Invoquer fonctions AWS Lambda.

Pour essayer votre fonction Lambda

1. Si vous ne l'avez pas déjà fait, effectuez les opérations suivantes :

   a. Assurez-vous que l'utilisateur dispose des autorisations `lambda:InvokeFunction` nécessaires. Vous pouvez utiliser la politique suivante :

      Vous pouvez obtenir l'ARN de votre fonction Lambda à partir de l'aperçu des fonctions dans la console Lambda.

      Pour activer l'accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

      - Utilisateurs et groupes dans AWS IAM Identity Center :

        Créez un jeu d'autorisations. Suivez les instructions de la rubrique Création d'un jeu d'autorisations du Guide de l'utilisateur AWS IAM Identity Center .
      - Utilisateurs gérés dans IAM par un fournisseur d'identité :

Créez un rôle pour la fédération d'identité. Suivez les instructions de la rubrique Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM.

- Utilisateurs IAM :
  - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d'un rôle pour un utilisateur IAM dans le Guide de l'utilisateur IAM.
  - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique Ajout d'autorisations à un utilisateur (console) du Guide de l'utilisateur IAM.

b. Installez et configurez le AWS SDK pour Python. Pour de plus amples informations, veuillez consulter Étape 2 : configurer le AWS CLI and AWS Kits SDK.

2. Copiez le code JSON suivant dans un fichier nommé `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. Exécutez le code. Si le document se trouve dans un compartiment Amazon S3, assurez-vous qu'il s'agit du même compartiment que celui que vous avez spécifié précédemment à l'étape 12 de Créer une fonction Lambda (console).

   En cas de succès, votre code renvoie une réponse JSON partielle pour chaque type de bloc détecté dans le document.

Exemples de code pour Amazon Rekognition à l'aide de AWS Kits SDK

Les exemples de code suivants montrent comment utiliser Amazon Rekognition AWS avec un kit de développement logiciel (SDK). Les exemples de code présentés dans ce chapitre sont destinés à compléter les exemples de code trouvés dans le reste de ce guide.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit de développement logiciel (SDK).

Exemples de code

- Exemples de base pour l'utilisation d'Amazon Rekognition AWS Kits SDK
  - Bonjour Amazon Rekognition
  - Actions pour Amazon Rekognition utilisant AWS Kits SDK
    - À utiliser CompareFacesavec un AWS SDK ou CLI
    - À utiliser CreateCollectionavec un AWS SDK ou CLI
    - À utiliser DeleteCollectionavec un AWS SDK ou CLI
    - À utiliser DeleteFacesavec un AWS SDK ou CLI
    - À utiliser DescribeCollectionavec un AWS SDK ou CLI
    - À utiliser DetectFacesavec un AWS SDK ou CLI
    - À utiliser DetectLabelsavec un AWS SDK ou CLI
    - À utiliser DetectModerationLabelsavec un AWS SDK ou CLI
    - À utiliser DetectTextavec un AWS SDK ou CLI
    - À utiliser GetCelebrityInfoavec un AWS SDK ou CLI

    - À utiliser IndexFacesavec un AWS SDK ou CLI
    - À utiliser ListCollectionsavec un AWS SDK ou CLI
    - À utiliser ListFacesavec un AWS SDK ou CLI
    - À utiliser RecognizeCelebritiesavec un AWS SDK ou CLI
    - À utiliser SearchFacesavec un AWS SDK ou CLI
    - À utiliser SearchFacesByImageavec un AWS SDK ou CLI
- Scénarios d'utilisation d'Amazon Rekognition AWS Kits SDK
  - Créez une collection Amazon Rekognition et retrouvez-y des visages à l'aide d'un AWS Kit SDK
  - Création d'une application de gestion des ressources photographiques permettant aux utilisateurs de gérer les photos à l'aide d'étiquettes
  - Détectez le PPE dans les images avec Amazon Rekognition à l'aide d'un AWS Kit SDK
  - Détectez et affichez des éléments dans des images avec Amazon Rekognition à l'aide d'un AWS Kit SDK
  - Détecter les visages dans une image à l'aide d'un AWS Kit SDK
  - Détectez les informations contenues dans les vidéos à l'aide d'Amazon Rekognition et du AWS Kit SDK
  - Détectez des objets dans des images avec Amazon Rekognition à l'aide d'un AWS Kit SDK
  - Détectez les personnes et les objets dans une vidéo avec Amazon Rekognition à l'aide d'un AWS Kit SDK
  - Enregistrez les informations EXIF et autres informations sur l'image à l'aide d'un AWS Kit SDK

Exemples de base pour l'utilisation d'Amazon Rekognition AWS Kits SDK

Les exemples de code suivants montrent les bases de l'utilisation d'Amazon Rekognition avec des kits AWS SDK.

Exemples

- Bonjour Amazon Rekognition
- Actions pour Amazon Rekognition utilisant AWS Kits SDK
  - À utiliser CompareFacesavec un AWS SDK ou CLI
  - À utiliser CreateCollectionavec un AWS SDK ou CLI

- À utiliser DeleteCollectionavec un AWS SDK ou CLI
- À utiliser DeleteFacesavec un AWS SDK ou CLI
- À utiliser DescribeCollectionavec un AWS SDK ou CLI
- À utiliser DetectFacesavec un AWS SDK ou CLI
- À utiliser DetectLabelsavec un AWS SDK ou CLI
- À utiliser DetectModerationLabelsavec un AWS SDK ou CLI
- À utiliser DetectTextavec un AWS SDK ou CLI
- À utiliser GetCelebrityInfoavec un AWS SDK ou CLI
- À utiliser IndexFacesavec un AWS SDK ou CLI
- À utiliser ListCollectionsavec un AWS SDK ou CLI
- À utiliser ListFacesavec un AWS SDK ou CLI
- À utiliser RecognizeCelebritiesavec un AWS SDK ou CLI
- À utiliser SearchFacesavec un AWS SDK ou CLI
- À utiliser SearchFacesByImageavec un AWS SDK ou CLI

Bonjour Amazon Rekognition

L'exemple de code suivant montre comment faire ses premiers pas avec Amazon Rekognition.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Code pour le fichier CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
```

```
set(SERVICE_COMPONENTS rekognition)

# Set this project's name.
project("hello_rekognition")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_rekognition.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code pour le fichier source 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;
}
```

- Pour plus de détails sur l'API, reportez-vous ListCollectionsà la section Référence des AWS SDK pour C++ API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Actions pour Amazon Rekognition utilisant AWS Kits SDK

Les exemples de code suivants montrent comment effectuer des actions Amazon AWS Rekognition individuelles avec des SDK. Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions pour configurer et exécuter le code.

Ces extraits appellent l'API Amazon Rekognition et sont des extraits de code de programmes plus volumineux qui doivent être exécutés en contexte. Vous pouvez voir les actions dans leur contexte dans Scénarios d'utilisation d'Amazon Rekognition AWS Kits SDK.

Les exemples suivants incluent uniquement les actions les plus couramment utilisées. Pour obtenir la liste complète, consultez la Référence de l'API Amazon Rekognition.

Exemples

- À utiliser CompareFacesavec un AWS SDK ou CLI
- À utiliser CreateCollectionavec un AWS SDK ou CLI
- À utiliser DeleteCollectionavec un AWS SDK ou CLI
- À utiliser DeleteFacesavec un AWS SDK ou CLI
- À utiliser DescribeCollectionavec un AWS SDK ou CLI
- À utiliser DetectFacesavec un AWS SDK ou CLI

- À utiliser DetectLabelsavec un AWS SDK ou CLI
- À utiliser DetectModerationLabelsavec un AWS SDK ou CLI
- À utiliser DetectTextavec un AWS SDK ou CLI
- À utiliser GetCelebrityInfoavec un AWS SDK ou CLI
- À utiliser IndexFacesavec un AWS SDK ou CLI
- À utiliser ListCollectionsavec un AWS SDK ou CLI
- À utiliser ListFacesavec un AWS SDK ou CLI
- À utiliser RecognizeCelebritiesavec un AWS SDK ou CLI
- À utiliser SearchFacesavec un AWS SDK ou CLI
- À utiliser SearchFacesByImageavec un AWS SDK ou CLI

À utiliser **`CompareFaces`**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `CompareFaces`.

Pour plus d'informations, veuillez consulter Comparaison de visages dans des images.

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to compare faces in two images.
/// </summary>
public class CompareFaces
{
```

```
        public static async Task Main()
        {
            float similarityThreshold = 70F;
            string sourceImage = "source.jpg";
            string targetImage = "target.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            Amazon.Rekognition.Model.Image imageSource = new
 Amazon.Rekognition.Model.Image();

            try
            {
                using FileStream fs = new FileStream(sourceImage, FileMode.Open,
 FileAccess.Read);
                byte[] data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageSource.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine($"Failed to load source image: {sourceImage}");
                return;
            }

            Amazon.Rekognition.Model.Image imageTarget = new
 Amazon.Rekognition.Model.Image();

            try
            {
                using FileStream fs = new FileStream(targetImage, FileMode.Open,
 FileAccess.Read);
                byte[] data = new byte[fs.Length];
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageTarget.Bytes = new MemoryStream(data);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Failed to load target image: {targetImage}");
                Console.WriteLine(ex.Message);
                return;
            }
```

```
            var compareFacesRequest = new CompareFacesRequest
            {
                SourceImage = imageSource,
                TargetImage = imageTarget,
                SimilarityThreshold = similarityThreshold,
            };

            // Call operation
            var compareFacesResponse = await
 rekognitionClient.CompareFacesAsync(compareFacesRequest);

            // Display results
            compareFacesResponse.FaceMatches.ForEach(match =>
            {
                ComparedFace face = match.Face;
                BoundingBox position = face.BoundingBox;
                Console.WriteLine($"Face at {position.Left} {position.Top}
 matches with {match.Similarity}% confidence.");
            });

            Console.WriteLine($"Found {compareFacesResponse.UnmatchedFaces.Count}
 face(s) that did not match.");
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous CompareFacesà la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour comparer les visages dans deux images

La commande `compare-faces` suivante compare les visages dans deux images stockées dans un compartiment Amazon S3.

```
aws rekognition compare-faces \
    --source-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"source.jpg"}}' \
```

```
    --target-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"target.jpg"}}'
```

Sortie :

```
{
    "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
    }
}
```

Pour plus d'informations, consultez Comparaison de visages dans les images dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir CompareFaceslа section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

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

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

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

            Where:
                bucketName - The name of the S3 bucket where the images are
 stored.
                sourceKey  - The S3 key (file name) for the source image.
                targetKey  - The S3 key (file name) for the target image.
           """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceKey = args[1];
        String targetKey = args[2];

        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
```

```
        compareTwoFaces(rekClient, bucketName, sourceKey, targetKey);
    }

   /**
    * Compares two faces from images stored in an Amazon S3 bucket using AWS
Rekognition.
    *
    * <p>This method takes two image keys from an S3 bucket and compares the
faces within them.
    * It prints out the confidence level of matched faces and reports the number
of unmatched faces.</p>
    *
    * @param rekClient   The {@link RekognitionClient} used to call AWS
Rekognition.
    * @param bucketName  The name of the S3 bucket containing the images.
    * @param sourceKey   The object key (file path) for the source image in the
S3 bucket.
    * @param targetKey   The object key (file path) for the target image in the
S3 bucket.
    * @throws RuntimeException If the Rekognition service returns an error.
    */
   public static void compareTwoFaces(RekognitionClient rekClient, String
bucketName, String sourceKey, String targetKey) {
       try {
           Float similarityThreshold = 70F;
           S3Object s3ObjectSource = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceKey)
                   .build();

           Image sourceImage = Image.builder()
                   .s3Object(s3ObjectSource)
                   .build();

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

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

           CompareFacesRequest facesRequest = CompareFacesRequest.builder()
```

```
                    .sourceImage(sourceImage)
                    .targetImage(targetImage)
                    .similarityThreshold(similarityThreshold)
                    .build();

            // Compare the two images.
            CompareFacesResponse compareFacesResult =
 rekClient.compareFaces(facesRequest);
            List<CompareFacesMatch> faceDetails =
 compareFacesResult.faceMatches();

            for (CompareFacesMatch match : faceDetails) {
                ComparedFace face = match.face();
                BoundingBox position = face.boundingBox();
                System.out.println("Face at " + position.left().toString()
                        + " " + position.top()
                        + " matches with " + face.confidence().toString()
                        + "% confidence.");
            }

            List<ComparedFace> unmatchedFaces =
 compareFacesResult.unmatchedFaces();
            System.out.println("There were " + unmatchedFaces.size() + " face(s)
 that did not match.");

        } catch (RekognitionException e) {
            System.err.println("Error comparing faces: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous CompareFacesà la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun compareTwoFaces(
    similarityThresholdVal: Float,
    sourceImageVal: String,
    targetImageVal: String,
) {
    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage =
        Image {
            bytes = sourceBytes
        }

    val tarImage =
        Image {
            bytes = targetBytes
        }

    val facesRequest =
        CompareFacesRequest {
            sourceImage = souImage
            targetImage = tarImage
            similarityThreshold = similarityThresholdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->

        val compareFacesResult = rekClient.compareFaces(facesRequest)
        val faceDetails = compareFacesResult.faceMatches

        if (faceDetails != null) {
```

```
            for (match: CompareFacesMatch in faceDetails) {
                val face = match.face
                val position = face?.boundingBox
                if (position != null) {
                    println("Face at ${position.left} ${position.top} matches
 with ${face.confidence} % confidence.")
                }
            }
        }

        val uncompared = compareFacesResult.unmatchedFaces
        if (uncompared != null) {
            println("There was ${uncompared.size} face(s) that did not match")
        }

        println("Source image rotation:
 ${compareFacesResult.sourceImageOrientationCorrection}")
        println("target image rotation:
 ${compareFacesResult.targetImageOrientationCorrection}")
    }
}
```

- Pour plus de détails sur l'API, consultez CompareFaceslа section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
```

```
    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def compare_faces(self, target_image, similarity):
        """
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value
greater
                           than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                 reference image. The second element is the list of faces that
have
                 a similarity value below the specified threshold.
        """
        try:
            response = self.rekognition_client.compare_faces(
                SourceImage=self.image,
                TargetImage=target_image.image,
                SimilarityThreshold=similarity,
            )
            matches = [
                RekognitionFace(match["Face"]) for match in
response["FaceMatches"]
            ]
            unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
            logger.info(
                "Found %s matched faces and %s unmatched faces.",
                len(matches),
                len(unmatches),
```

```
            )
        except ClientError:
            logger.exception(
                "Couldn't match faces from %s to %s.",
                self.image_name,
                target_image.image_name,
            )
            raise
        else:
            return matches, unmatches
```

- Pour plus de détails sur l'API, consultez CompareFacesle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    TRY.
        " Create S3 object reference for the source image
        DATA(lo_source_s3obj) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_source_s3_bucket
          iv_name = iv_source_s3_key ).

        " Create source image object
        DATA(lo_source_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_source_s3obj ).

        " Create S3 object reference for the target image
        DATA(lo_target_s3obj) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_target_s3_bucket
          iv_name = iv_target_s3_key ).
```

```
      " Create target image object
      DATA(lo_target_image) = NEW /aws1/cl_rekimage(
        io_s3object = lo_target_s3obj ).

      " Compare faces
      oo_result = lo_rek->comparefaces(
        io_sourceimage = lo_source_image
        io_targetimage = lo_target_image
        iv_similaritythreshold = iv_similarity ).

      DATA(lt_face_matches) = oo_result->get_facematches( ).
      DATA(lt_unmatched_faces) = oo_result->get_unmatchedfaces( ).

      " Get counts of matched and unmatched faces
      DATA(lv_matched_count) = lines( lt_face_matches ).
      DATA(lv_unmatched_count) = lines( lt_unmatched_faces ).

      " Output detailed comparison results
      DATA(lv_message) = |Face comparison completed: | &&
                         |{ lv_matched_count } matched face(s), | &&
                         |{ lv_unmatched_count } unmatched face(s).|.
      MESSAGE lv_message TYPE 'I'.
    CATCH /aws1/cx_rekinvalids3objectex.
      MESSAGE 'Invalid S3 object.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Pour plus de détails sur l'API, consultez CompareFacesla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **`CreateCollection`**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `CreateCollection`.

Pour plus d'informations, consultez Création d'une collection.

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses Amazon Rekognition to create a collection to which you can add
/// faces using the IndexFaces operation.
/// </summary>
public class CreateCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine("Creating collection: " + collectionId);

        var createCollectionRequest = new CreateCollectionRequest
        {
            CollectionId = collectionId,
        };

        CreateCollectionResponse createCollectionResponse = await
rekognitionClient.CreateCollectionAsync(createCollectionRequest);
        Console.WriteLine($"CollectionArn :
{createCollectionResponse.CollectionArn}");
        Console.WriteLine($"Status code :
{createCollectionResponse.StatusCode}");
    }
}
```

- Pour plus de détails sur l'API, reportez-vous CreateCollectionà la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour créer une collection

La commande `create-collection` suivante crée une collection portant le nom spécifié.

```
aws rekognition create-collection \
    --collection-id "MyCollection"
```

Sortie :

```
{
    "CollectionArn": "aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
    "FaceModelVersion": "4.0",
    "StatusCode": 200
}
```

Pour plus d'informations, consultez Création d'une collection dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir CreateCollectionla section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

            Usage: <collectionName>\s

            Where:
                collectionName - The name of the collection.\s
            """;

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

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

        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
```

```
     * Creates a new Amazon Rekognition collection.
     *
     * @param rekClient    the Amazon Rekognition client used to interact with
 the Rekognition service
     * @param collectionId the unique identifier for the collection to be created
     */
    public static void createMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            CreateCollectionRequest collectionRequest =
 CreateCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            CreateCollectionResponse collectionResponse =
 rekClient.createCollection(collectionRequest);
            System.out.println("CollectionArn: " +
 collectionResponse.collectionArn());
            System.out.println("Status code: " +
 collectionResponse.statusCode().toString());

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

- Pour plus de détails sur l'API, reportez-vous CreateCollectionà la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l’exemple complet et découvrez comment le configurer et l’exécuter dans le référentiel d’exemples de code AWS.

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- Pour plus de détails sur l'API, consultez CreateCollectionla section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
 API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client
```

```
    def create_collection(self, collection_id):
        """
        Creates an empty collection.

        :param collection_id: Text that identifies the collection.
        :return: The newly created collection.
        """
        try:
            response = self.rekognition_client.create_collection(
                CollectionId=collection_id
            )
            response["CollectionId"] = collection_id
            collection = RekognitionCollection(response, self.rekognition_client)
            logger.info("Created collection %s.", collection_id)
        except ClientError:
            logger.exception("Couldn't create collection %s.", collection_id)
            raise
        else:
            return collection
```

- Pour plus de détails sur l'API, consultez CreateCollectionle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    TRY.
        oo_result = lo_rek->createcollection(
          iv_collectionid = iv_collection_id ).
        MESSAGE 'Collection created successfully.' TYPE 'I'.
```

```
      CATCH /aws1/cx_rekresrcalrdyexistsex.
        MESSAGE 'Collection already exists.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Pour plus de détails sur l'API, consultez CreateCollectionla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **`DeleteCollection`**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `DeleteCollection`.

Pour plus d'informations, consultez Suppression d'une collection.

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to delete an existing collection.
/// </summary>
public class DeleteCollection
{
    public static async Task Main()
```

```
        {
            var rekognitionClient = new AmazonRekognitionClient();

            string collectionId = "MyCollection";
            Console.WriteLine("Deleting collection: " + collectionId);

            var deleteCollectionRequest = new DeleteCollectionRequest()
            {
                CollectionId = collectionId,
            };

            var deleteCollectionResponse = await
 rekognitionClient.DeleteCollectionAsync(deleteCollectionRequest);
            Console.WriteLine($"{collectionId}:
 {deleteCollectionResponse.StatusCode}");
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous DeleteCollectionà la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour supprimer une collection

La commande `delete-collection` suivante supprime la collection spécifiée.

```
aws rekognition delete-collection \
    --collection-id MyCollection
```

Sortie :

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

Pour plus d’informations, consultez Suppression d’une collection dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir DeleteCollectionla section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

            Where:
                collectionId - The id of the collection to delete.\s
            """;

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

```
        }

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

        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Deletes an Amazon Rekognition collection.
     *
     * @param rekClient      An instance of the {@link RekognitionClient} class,
 which is used to interact with the Amazon Rekognition service.
     * @param collectionId   The ID of the collection to be deleted.
     */
    public static void deleteMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            DeleteCollectionRequest deleteCollectionRequest =
 DeleteCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            DeleteCollectionResponse deleteCollectionResponse =
 rekClient.deleteCollection(deleteCollectionRequest);
            System.out.println(collectionId + ": " +
 deleteCollectionResponse.statusCode().toString());

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

- Pour plus de détails sur l'API, reportez-vous DeleteCollectionà la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- Pour plus de détails sur l'API, consultez DeleteCollectionla section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
```

```
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def delete_collection(self):
        """
        Deletes the collection.
        """
        try:

self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
```

```
            logger.exception("Couldn't delete collection %s.",
self.collection_id)
            raise
```

- Pour plus de détails sur l'API, consultez DeleteCollectionle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    TRY.
        lo_rek->deletecollection(
          iv_collectionid = iv_collection_id ).
        MESSAGE 'Collection deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Pour plus de détails sur l'API, consultez DeleteCollectionla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **`DeleteFaces`**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `DeleteFaces`.

Pour plus d'informations, veuillez consulter Supprimer des visages d'une collection.

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to delete one or more faces from
    /// a Rekognition collection.
    /// </summary>
    public class DeleteFaces
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection";
            var faces = new List<string> { "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx" };

            var rekognitionClient = new AmazonRekognitionClient();

            var deleteFacesRequest = new DeleteFacesRequest()
            {
                CollectionId = collectionId,
                FaceIds = faces,
            };

            DeleteFacesResponse deleteFacesResponse = await
 rekognitionClient.DeleteFacesAsync(deleteFacesRequest);
            deleteFacesResponse.DeletedFaces.ForEach(face =>
            {
```

```
                Console.WriteLine($"FaceID: {face}");
            });
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous DeleteFacesà la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour supprimer des visages d'une collection

La commande `delete-faces` suivante supprime le visage spécifié d'une collection.

```
aws rekognition delete-faces \
    --collection-id MyCollection
    --face-ids '["0040279c-0178-436e-b70a-e61b074e96b0"]'
```

Sortie :

```
{
    "DeletedFaces": [
        "0040279c-0178-436e-b70a-e61b074e96b0"
    ]
}
```

Pour plus d'informations, consultez Suppression de visages d'une collection dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir DeleteFaceslа section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

            Where:
                collectionId - The id of the collection from which faces are
 deleted.\s
                faceId - The id of the face to delete.\s
           """;

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

        String collectionId = args[0];
        String faceId = args[1];
```

```
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }

    /**
     * Deletes a face from the specified Amazon Rekognition collection.
     *
     * @param rekClient     an instance of the Amazon Rekognition client
     * @param collectionId  the ID of the collection from which the face should
 be deleted
     * @param faceId        the ID of the face to be deleted
     * @throws RekognitionException if an error occurs while deleting the face
     */
    public static void deleteFacesCollection(RekognitionClient rekClient,
            String collectionId,
            String faceId) {

        try {
            DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceIds(faceId)
                    .build();

            rekClient.deleteFaces(deleteFacesRequest);
            System.out.println("The face was deleted from the collection.");

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

- Pour plus de détails sur l'API, reportez-vous DeleteFacesà la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
        println("$faceIdVal was deleted from the collection")
    }
}
```

- Pour plus de détails sur l'API, consultez DeleteFacesla section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def delete_faces(self, face_ids):
        """
        Deletes faces from the collection.

        :param face_ids: The list of IDs of faces to delete.
        :return: The list of IDs of faces that were deleted.
```

```
        """
        try:
            response = self.rekognition_client.delete_faces(
                CollectionId=self.collection_id, FaceIds=face_ids
            )
            deleted_ids = response["DeletedFaces"]
            logger.info(
                "Deleted %s faces from %s.", len(deleted_ids), self.collection_id
            )
        except ClientError:
            logger.exception("Couldn't delete faces from %s.",
self.collection_id)
            raise
        else:
            return deleted_ids
```

- Pour plus de détails sur l'API, consultez DeleteFacesle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    TRY.
        oo_result = lo_rek->deletefaces(
          iv_collectionid = iv_collection_id
          it_faceids = it_face_ids ).

        DATA(lt_deleted_faces) = oo_result->get_deletedfaces( ).
        DATA(lv_deleted_count) = lines( lt_deleted_faces ).
        DATA(lv_msg6) = |{ lv_deleted_count } face(s) deleted successfully.|.
        MESSAGE lv_msg6 TYPE 'I'.
```

```
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Pour plus de détails sur l'API, consultez DeleteFaceslа section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **`DescribeCollection`**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `DescribeCollection`.

Pour plus d'informations, veuillez consulter Description d'une collection.

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to describe the contents of a
/// collection.
/// </summary>
public class DescribeCollection
{
```

```
        public static async Task Main()
        {
            var rekognitionClient = new AmazonRekognitionClient();

            string collectionId = "MyCollection";
            Console.WriteLine($"Describing collection: {collectionId}");

            var describeCollectionRequest = new DescribeCollectionRequest()
            {
                CollectionId = collectionId,
            };

            var describeCollectionResponse = await
 rekognitionClient.DescribeCollectionAsync(describeCollectionRequest);
            Console.WriteLine($"Collection ARN:
 {describeCollectionResponse.CollectionARN}");
            Console.WriteLine($"Face count:
 {describeCollectionResponse.FaceCount}");
            Console.WriteLine($"Face model version:
 {describeCollectionResponse.FaceModelVersion}");
            Console.WriteLine($"Created:
 {describeCollectionResponse.CreationTimestamp}");
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous DescribeCollectionà la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour décrire une collection

L'exemple `describe-collection` suivant affiche les détails sur la collection spécifiée.

```
aws rekognition describe-collection \
    --collection-id MyCollection
```

Sortie :

```
{
    "FaceCount": 200,
    "CreationTimestamp": 1569444828.274,
    "CollectionARN": "arn:aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
    "FaceModelVersion": "4.0"
}
```

Pour plus d'informations, consultez Description d'une collection dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir DescribeCollectionla section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

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

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
            """;

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

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

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    /**
     * Describes an Amazon Rekognition collection.
     *
     * @param rekClient         The Amazon Rekognition client used to make the
 request.
     * @param collectionName    The name of the collection to describe.
     *
     * @throws RekognitionException If an error occurs while describing the
 collection.
     */
    public static void describeColl(RekognitionClient rekClient, String
 collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
 DescribeCollectionRequest.builder()
                    .collectionId(collectionName)
                    .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
                    .describeCollection(describeCollectionRequest);
```

```
            System.out.println("Collection Arn : " +
 describeCollectionResponse.collectionARN());
            System.out.println("Created : " +
 describeCollectionResponse.creationTimestamp().toString());

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

- Pour plus de détails sur l'API, reportez-vous DescribeCollectionà la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

- Pour plus de détails sur l'API, consultez DescribeCollectionla section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
```

```
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        """
        try:
            response = self.rekognition_client.describe_collection(
                CollectionId=self.collection_id
            )
            # Work around capitalization of Arn vs. ARN
            response["CollectionArn"] = response.get("CollectionARN")
            (
                self.collection_arn,
                self.face_count,
                self.created,
            ) = self._unpack_collection(response)
            logger.info("Got data for collection %s.", self.collection_id)
        except ClientError:
            logger.exception("Couldn't get data for collection %s.",
self.collection_id)
            raise
        else:
            return self.to_dict()
```

- Pour plus de détails sur l'API, consultez DescribeCollectionle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   TRY.
       oo_result = lo_rek->describecollection(
         iv_collectionid = iv_collection_id ).
       DATA(lv_face_count) = oo_result->get_facecount( ).
       DATA(lv_msg) = |Collection described: { lv_face_count } face(s)
indexed.|.
       MESSAGE lv_msg TYPE 'I'.
     CATCH /aws1/cx_rekresourcenotfoundex.
       MESSAGE 'Collection not found.' TYPE 'E'.
     CATCH /aws1/cx_rekinvalidparameterex.
       MESSAGE 'Invalid parameter value.' TYPE 'E'.
   ENDTRY.
```

- Pour plus de détails sur l'API, consultez DescribeCollectionla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **`DetectFaces`**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `DetectFaces`.

Pour plus d'informations, veuillez consulter Détecter des visages dans une image.

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DetectFaces
{
    public static async Task Main()
    {
        string photo = "input.jpg";
        string bucket = "amzn-s3-demo-bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectFacesRequest = new DetectFacesRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },

            // Attributes can be "ALL" or "DEFAULT".
```

```
                // "DEFAULT": BoundingBox, Confidence, Landmarks, Pose, and
 Quality.
                // "ALL": See https://docs.aws.amazon.com/sdkfornet/v3/apidocs/
items/Rekognition/TFaceDetail.html
                Attributes = new List<string>() { "ALL" },
            };

            try
            {
                DetectFacesResponse detectFacesResponse = await
 rekognitionClient.DetectFacesAsync(detectFacesRequest);
                bool hasAll = detectFacesRequest.Attributes.Contains("ALL");
                foreach (FaceDetail face in detectFacesResponse.FaceDetails)
                {
                    Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
 left={face.BoundingBox.Top} width={face.BoundingBox.Width}
 height={face.BoundingBox.Height}");
                    Console.WriteLine($"Confidence: {face.Confidence}");
                    Console.WriteLine($"Landmarks: {face.Landmarks.Count}");
                    Console.WriteLine($"Pose: pitch={face.Pose.Pitch}
 roll={face.Pose.Roll} yaw={face.Pose.Yaw}");
                    Console.WriteLine($"Brightness:
 {face.Quality.Brightness}\tSharpness: {face.Quality.Sharpness}");

                    if (hasAll)
                    {
                        Console.WriteLine($"Estimated age is between
 {face.AgeRange.Low} and {face.AgeRange.High} years old.");
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Afficher les informations du cadre de délimitation pour tous les visages d’une image.

```
    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}");
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous DetectFacesà la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour détecter des visages sur une image

La commande `detect-faces` suivante détecte les visages sur l'image spécifiée stockée dans un compartiment Amazon S3.

```
aws rekognition detect-faces \
    --image '{"S3Object":{"Bucket":"MyImageS3Bucket","Name":"MyFriend.jpg"}}' \
    --attributes "ALL"
```

Sortie :

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

Pour plus d'informations, consultez Détection de visages sur une image dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir DetectFaceslа section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

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

import java.util.List;

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

```
        final String usage = """

            Usage:   <bucketName> <sourceImage>

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

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

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

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

    /**
     * Detects faces in an image stored in an Amazon S3 bucket using the Amazon
Rekognition service.
     *
     * @param rekClient    The Amazon Rekognition client used to interact with
the Rekognition service.
     * @param bucketName   The name of the Amazon S3 bucket where the source
image is stored.
     * @param sourceImage  The name of the source image file in the Amazon S3
bucket.
     */
    public static void detectFacesinImage(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
```

```
                .build();

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

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

            DetectFacesResponse facesResponse =
 rekClient.detectFaces(facesRequest);
            List<FaceDetail> faceDetails = facesResponse.faceDetails();
            for (FaceDetail face : faceDetails) {
                AgeRange ageRange = face.ageRange();
                System.out.println("The detected face is estimated to be between
 "
                        + ageRange.low().toString() + " and " +
 ageRange.high().toString()
                        + " years old.");

                System.out.println("There is a smile : " +
 face.smile().value().toString());
            }

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

- Pour plus de détails sur l'API, reportez-vous DetectFacesà la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between
 ${ageRange?.low} and ${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}
```

- Pour plus de détails sur l'API, consultez DetectFacesla section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
```

```
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
            raise
        else:
            return faces
```

- Pour plus de détails sur l'API, consultez DetectFaceslle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Detect faces in the image with all attributes
        DATA(lt_attributes) = VALUE /aws1/cl_rekattributes_w=>tt_attributes( ).
        DATA(lo_attr_wrapper) = NEW /aws1/cl_rekattributes_w( iv_value = 'ALL' ).
        INSERT lo_attr_wrapper INTO TABLE lt_attributes.

        oo_result = lo_rek->detectfaces(
          io_image = lo_image
          it_attributes = lt_attributes ).

        DATA(lt_face_details) = oo_result->get_facedetails( ).
```

```
        DATA(lv_detected_count) = lines( lt_face_details ).
        DATA(lv_msg8) = |{ lv_detected_count } face(s) detected in image.|.
        MESSAGE lv_msg8 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Pour plus de détails sur l'API, consultez DetectFaceslа section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **`DetectLabels`**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `DetectLabels`.

Pour plus d'informations, veuillez consulter Détection des étiquettes dans une image.

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect labels within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
```

```
    /// </summary>
    public class DetectLabels
    {
        public static async Task Main()
        {
            string photo = "del_river_02092020_01.jpg"; // "input.jpg";
            string bucket = "amzn-s3-demo-bucket"; // "bucket";

            var rekognitionClient = new AmazonRekognitionClient();

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

            try
            {
                DetectLabelsResponse detectLabelsResponse = await
 rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (Label label in detectLabelsResponse.Labels)
                {
                    Console.WriteLine($"Name: {label.Name} Confidence:
 {label.Confidence}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Détectez les étiquettes dans un fichier image stocké sur votre ordinateur.

```
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);
            }
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous DetectLabelsà la section Référence des AWS SDK pour .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! Detect instances of real-world entities within an image by using Amazon
 Rekognition
/*!
  \param imageBucket: The Amazon Simple Storage Service (Amazon S3) bucket
 containing an image.
  \param imageKey: The Amazon S3 key of an image object.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::Rekognition::detectLabels(const Aws::String &imageBucket,
```

```
                                        const Aws::String &imageKey,
                                        const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::Rekognition::RekognitionClient rekognitionClient(clientConfiguration);

    Aws::Rekognition::Model::DetectLabelsRequest request;
    Aws::Rekognition::Model::S3Object s3Object;
    s3Object.SetBucket(imageBucket);
    s3Object.SetName(imageKey);

    Aws::Rekognition::Model::Image image;
    image.SetS3Object(s3Object);

    request.SetImage(image);

    const Aws::Rekognition::Model::DetectLabelsOutcome outcome =
 rekognitionClient.DetectLabels(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::Rekognition::Model::Label> &labels =
 outcome.GetResult().GetLabels();
        if (labels.empty()) {
            std::cout << "No labels detected" << std::endl;
        } else {
            for (const Aws::Rekognition::Model::Label &label: labels) {
                std::cout << label.GetName() << ": " << label.GetConfidence() <<
 std::endl;
            }
        }
    } else {
        std::cerr << "Error while detecting labels: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, reportez-vous DetectLabelsà la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Pour détecter une étiquette dans une image

L'exemple `detect-labels` suivant détecte les scènes et les objets dans une image stockée dans un compartiment Amazon S3.

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

Sortie :

```
{
    "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"
}
```

Pour plus d’informations, consultez Détection des étiquettes dans une image dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir DetectLabelsla section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

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

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

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

            Where:
                bucketName - The name of the Amazon S3 bucket where the image is
 stored
```

```
                sourceImage - The name of the image file (for example, pic1.png).
\s
            """;

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

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

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

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

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

            DetectLabelsRequest detectLabelsRequest =
 DetectLabelsRequest.builder()
```

```
                    .image(souImage)
                    .maxLabels(10)
                    .build();

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

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

- Pour plus de détails sur l'API, voir DetectLabelsla section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }
    val request =
        DetectLabelsRequest {
```

```
            image = souImage
            maxLabels = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- Pour plus de détails sur l'API, consultez DetectLabelsla section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
```

```
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_labels(self, max_labels):
        """
        Detects labels in the image. Labels are objects and people.

        :param max_labels: The maximum number of labels to return.
        :return: The list of labels detected in the image.
        """
        try:
            response = self.rekognition_client.detect_labels(
                Image=self.image, MaxLabels=max_labels
            )
            labels = [RekognitionLabel(label) for label in response["Labels"]]
            logger.info("Found %s labels in %s.", len(labels), self.image_name)
        except ClientError:
            logger.info("Couldn't detect labels in %s.", self.image_name)
            raise
        else:
            return labels
```

- Pour plus de détails sur l'API, consultez DetectLabelsle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    TRY.
        " Create S3 object reference for the image
```

```
      DATA(lo_s3object) = NEW /aws1/cl_reks3object(
        iv_bucket = iv_s3_bucket
        iv_name = iv_s3_key ).

      " Create image object
      DATA(lo_image) = NEW /aws1/cl_rekimage(
        io_s3object = lo_s3object ).

      " Detect labels in the image
      oo_result = lo_rek->detectlabels(
        io_image = lo_image
        iv_maxlabels = iv_max_labels ).

      DATA(lt_labels) = oo_result->get_labels( ).
      DATA(lv_label_count) = lines( lt_labels ).
      DATA(lv_msg9) = |{ lv_label_count } label(s) detected in image.|.
      MESSAGE lv_msg9 TYPE 'I'.
    CATCH /aws1/cx_rekinvalids3objectex.
      MESSAGE 'Invalid S3 object.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Pour plus de détails sur l'API, consultez DetectLabelsla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **`DetectModerationLabels`**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `DetectModerationLabels`.

Pour plus d'informations, veuillez consulter Détecter des images inappropriées.

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect unsafe content in a
/// JPEG or PNG format image.
/// </summary>
public class DetectModerationLabels
{
    public static async Task Main(string[] args)
    {
        string photo = "input.jpg";
        string bucket = "amzn-s3-demo-bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectModerationLabelsRequest = new
DetectModerationLabelsRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },
            MinConfidence = 60F,
        };
```

```
            try
            {
                var detectModerationLabelsResponse = await
 rekognitionClient.DetectModerationLabelsAsync(detectModerationLabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (ModerationLabel label in
 detectModerationLabelsResponse.ModerationLabels)
                {
                    Console.WriteLine($"Label: {label.Name}");
                    Console.WriteLine($"Confidence: {label.Confidence}");
                    Console.WriteLine($"Parent: {label.ParentName}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

- Pour plus de détails sur l'API, voir DetectModerationLabelsla section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour détecter le contenu inapproprié dans une image

La commande `detect-moderation-labels` suivante détecte le contenu inapproprié dans l'image spécifiée stockée dans un compartiment Amazon S3.

```
aws rekognition detect-moderation-labels \
    --image "S3Object={Bucket=MyImageS3Bucket,Name=gun.jpg}"
```

Sortie :

```
{
    "ModerationModelVersion": "3.0",
    "ModerationLabels": [
```

```
        {
            "Confidence": 97.29618072509766,
            "ParentName": "Violence",
            "Name": "Weapon Violence"
        },
        {
            "Confidence": 97.29618072509766,
            "ParentName": "",
            "Name": "Violence"
        }
    ]
}
```

Pour plus d'informations, consultez Détection d'images inappropriées dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir DetectModerationLabelsla section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

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

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

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

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

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

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

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

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

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

    /**
     * Detects moderation labels in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient      the Amazon Rekognition client to use for the
 detection
     * @param bucketName     the name of the Amazon S3 bucket where the image is
 stored
     * @param sourceImage    the name of the image file to be analyzed
     *
     * @throws RekognitionException if there is an error during the image
 detection process
```

```
     */
    public static void detectModLabels(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

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

            DetectModerationLabelsRequest moderationLabelsRequest =
 DetectModerationLabelsRequest.builder()
                    .image(targetImage)
                    .minConfidence(60F)
                    .build();

            DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                    .detectModerationLabels(moderationLabelsRequest);
            List<ModerationLabel> labels =
 moderationLabelsResponse.moderationLabels();
            System.out.println("Detected labels for image");
            for (ModerationLabel label : labels) {
                System.out.println("Label: " + label.name()
                        + "\n Confidence: " + label.confidence().toString() + "%"
                        + "\n Parent:" + label.parentName());
            }

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

- Pour plus de détails sur l'API, voir DetectModerationLabelsla section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} %
 Parent: ${label.parentName}")
        }
    }
}
```

- Pour plus de détails sur l'API, consultez DetectModerationLabelsla section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify
 content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image
            )
```

```
            labels = [
                RekognitionModerationLabel(label)
                for label in response["ModerationLabels"]
            ]
            logger.info(
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels
```

- Pour plus de détails sur l'API, consultez DetectModerationLabelsle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Detect moderation labels
```

```
        oo_result = lo_rek->detectmoderationlabels(
          io_image = lo_image ).

        DATA(lt_moderation_labels) = oo_result->get_moderationlabels( ).
        DATA(lv_mod_count) = lines( lt_moderation_labels ).
        DATA(lv_msg10) = |{ lv_mod_count } moderation label(s) detected.|.
        MESSAGE lv_msg10 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Pour plus de détails sur l'API, consultez DetectModerationLabelsla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **`DetectText`**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `DetectText`.

Pour plus d'informations, consultez Détection de texte dans une image.

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;
```

```
    /// <summary>
    /// Uses the Amazon Rekognition Service to detect text in an image. The
    /// example was created using the AWS SDK for .NET version 3.7 and .NET
    /// Core 5.0.
    /// </summary>
    public class DetectText
    {
        public static async Task Main()
        {
            string photo = "Dad_photographer.jpg"; // "input.jpg";
            string bucket = "amzn-s3-demo-bucket"; // "bucket";

            var rekognitionClient = new AmazonRekognitionClient();

            var detectTextRequest = new DetectTextRequest()
            {
                Image = new Image()
                {
                    S3Object = new S3Object()
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },
            };

            try
            {
                DetectTextResponse detectTextResponse = await
rekognitionClient.DetectTextAsync(detectTextRequest);
                Console.WriteLine($"Detected lines and words for {photo}");
                detectTextResponse.TextDetections.ForEach(text =>
                {
                    Console.WriteLine($"Detected: {text.DetectedText}");
                    Console.WriteLine($"Confidence: {text.Confidence}");
                    Console.WriteLine($"Id : {text.Id}");
                    Console.WriteLine($"Parent Id: {text.ParentId}");
                    Console.WriteLine($"Type: {text.Type}");
                });
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
```

```
            }
        }
    }
```

- Pour plus de détails sur l'API, voir DetectTextla section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour détecter le texte dans une image

La commande `detect-text` suivante détecte le texte dans l'image spécifiée.

```
aws rekognition detect-text \
    --image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"ExamplePicture.jpg"}}'
```

Sortie :

```
{
    "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
        }
    ]
}
```

- Pour plus de détails sur l'API, voir DetectTextla section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

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

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

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:   <bucketName> <sourceImage>\n" +
            "\n" +
            "Where:\n" +
            "   bucketName - The name of the S3 bucket where the image is stored
\n" +
            "   sourceImage - The path to the image that contains text (for
 example, pic1.png). \n";

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

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

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

    /**
     * Detects text labels in an image stored in an S3 bucket using Amazon
 Rekognition.
     *
     * @param rekClient    an instance of the Amazon Rekognition client
     * @param bucketName   the name of the S3 bucket where the image is stored
     * @param sourceImage  the name of the image file in the S3 bucket
     * @throws RekognitionException if an error occurs while calling the Amazon
 Rekognition API
```

```
     */
    public static void detectTextLabels(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

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

            DetectTextRequest textRequest = DetectTextRequest.builder()
                    .image(souImage)
                    .build();

            DetectTextResponse textResponse = rekClient.detectText(textRequest);
            List<TextDetection> textCollection = textResponse.textDetections();
            System.out.println("Detected lines and words");
            for (TextDetection text : textCollection) {
                System.out.println("Detected: " + text.detectedText());
                System.out.println("Confidence: " +
 text.confidence().toString());
                System.out.println("Id : " + text.id());
                System.out.println("Parent Id: " + text.parentId());
                System.out.println("Type: " + text.type());
                System.out.println();
            }

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

- Pour plus de détails sur l'API, voir DetectTextla section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id:  ${text.parentId}")
            println("Type: ${text.type}")
        }
    }
}
```

- Pour plus de détails sur l'API, consultez DetectTextla section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in
 response["TextDetections"]]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
```

```
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

- Pour plus de détails sur l'API, consultez DetectTextle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Detect text in the image
        oo_result = lo_rek->detecttext(
          io_image = lo_image ).

        DATA(lt_text_detections) = oo_result->get_textdetections( ).
        DATA(lv_text_count) = lines( lt_text_detections ).
        DATA(lv_msg11) = |{ lv_text_count } text detection(s) found.|.
        MESSAGE lv_msg11 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
```

```
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Pour plus de détails sur l'API, consultez DetectTextla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **`GetCelebrityInfo`**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `GetCelebrityInfo`.

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to retrieve information about the
/// celebrity identified by the supplied celebrity Id.
/// </summary>
public class CelebrityInfo
{
    public static async Task Main()
    {
        string celebId = "nnnnnnnn";

        var rekognitionClient = new AmazonRekognitionClient();
```

```
            var celebrityInfoRequest = new GetCelebrityInfoRequest
            {
                Id = celebId,
            };

            Console.WriteLine($"Getting information for celebrity: {celebId}");

            var celebrityInfoResponse = await
 rekognitionClient.GetCelebrityInfoAsync(celebrityInfoRequest);

            // Display celebrity information.
            Console.WriteLine($"celebrity name: {celebrityInfoResponse.Name}");
            Console.WriteLine("Further information (if available):");
            celebrityInfoResponse.Urls.ForEach(url =>
            {
                Console.WriteLine(url);
            });
        }
    }
```

- Pour plus de détails sur l'API, voir GetCelebrityInfola section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour obtenir les informations sur une célébrité

La commande `get-celebrity-info` suivante affiche des informations sur la célébrité spécifiée. Le paramètre `id` provient d'un appel précédent à `recognize-celebrities`.

```
aws rekognition get-celebrity-info --id nnnnnnn
```

Sortie :

```
{
    "Name": "Celeb A",
    "Urls": [
```

```
        "www.imdb.com/name/aaaaaaaaa"
    ]
}
```

Pour plus d'informations, consultez Obtention d'informations sur une célébrité dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir GetCelebrityInfola section Référence des AWS CLI commandes.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser `IndexFaces`avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `IndexFaces`.

Pour plus d'informations, veuillez consulter Ajouter des visages à une collection.

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces in an image
/// that has been uploaded to an Amazon Simple Storage Service (Amazon S3)
/// bucket and then adds the information to a collection.
/// </summary>
```

```
    public class AddFaces
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection2";
            string bucket = "amzn-s3-demo-bucket";
            string photo = "input.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            var image = new Image
            {
                S3Object = new S3Object
                {
                    Bucket = bucket,
                    Name = photo,
                },
            };

            var indexFacesRequest = new IndexFacesRequest
            {
                Image = image,
                CollectionId = collectionId,
                ExternalImageId = photo,
                DetectionAttributes = new List<string>() { "ALL" },
            };

            IndexFacesResponse indexFacesResponse = await
rekognitionClient.IndexFacesAsync(indexFacesRequest);

            Console.WriteLine($"{photo} added");
            foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
            {
                Console.WriteLine($"Face detected: Faceid is
{faceRecord.Face.FaceId}");
            }
        }
    }
```

- Pour plus de détails sur l'API, voir IndexFaceslа section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour ajouter des visages à une collection

La commande `index-faces` suivante ajoute les visages trouvés dans une image à la collection spécifiée.

```
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"
```

Sortie :

```
{
    "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"
}
```

Pour plus d'informations, consultez Ajout de visages à une collection dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir IndexFaceslа section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import java.util.List;

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

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

            Where:
                collectionName - The name of the collection.
                sourceImage - The name of the image (for example, pic1.png).
                bucketName - The name of the S3 bucket.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        String bucketName = args[2];;
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        addToCollection(rekClient, collectionId, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Adds a face from an image to an Amazon Rekognition collection.
     *
     * @param rekClient     the Amazon Rekognition client
     * @param collectionId  the ID of the collection to add the face to
     * @param bucketName    the name of the Amazon S3 bucket containing the image
     * @param sourceImage   the name of the image file to add to the collection
     * @throws RekognitionException if there is an error while interacting with
 the Amazon Rekognition service
     */
```

```
   public static void addToCollection(RekognitionClient rekClient, String
collectionId, String bucketName, String sourceImage) {
       try {
           S3Object s3ObjectTarget = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceImage)
                   .build();

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

           IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                   .collectionId(collectionId)
                   .image(targetImage)
                   .maxFaces(1)
                   .qualityFilter(QualityFilter.AUTO)
                   .detectionAttributes(Attribute.DEFAULT)
                   .build();

           IndexFacesResponse facesResponse =
rekClient.indexFaces(facesRequest);
           System.out.println("Results for the image");
           System.out.println("\n Faces indexed:");
           List<FaceRecord> faceRecords = facesResponse.faceRecords();
           for (FaceRecord faceRecord : faceRecords) {
               System.out.println("  Face ID: " + faceRecord.face().faceId());
               System.out.println("  Location:" +
faceRecord.faceDetail().boundingBox().toString());
           }

           List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
           System.out.println("Faces not indexed:");
           for (UnindexedFace unindexedFace : unindexedFaces) {
               System.out.println("  Location:" +
unindexedFace.faceDetail().boundingBox().toString());
               System.out.println("  Reasons:");
               for (Reason reason : unindexedFace.reasons()) {
                   System.out.println("Reason:  " + reason);
               }
           }

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

```
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, voir IndexFacesla section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun addToCollection(
    collectionIdVal: String?,
    sourceImage: String,
) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        IndexFacesRequest {
            collectionId = collectionIdVal
            image = souImage
            maxFaces = 1
            qualityFilter = QualityFilter.Auto
            detectionAttributes = listOf(Attribute.Default)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)

        // Display the results.
```

```
        println("Results for the image")
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
            println("Face ID: ${faceRecord.face?.faceId}")
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
            println("Reasons:")

            unindexedFace.reasons?.forEach { reason ->
                println("Reason:  $reason")
            }
        }
    }
}
```

- Pour plus de détails sur l'API, consultez IndexFacesla section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
```

```
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def index_faces(self, image, max_faces):
        """
        Finds faces in the specified image, indexes them, and stores them in the
        collection.

        :param image: The image to index.
        :param max_faces: The maximum number of faces to index.
        :return: A tuple. The first element is a list of indexed faces.
                 The second element is a list of faces that couldn't be indexed.
        """
        try:
            response = self.rekognition_client.index_faces(
                CollectionId=self.collection_id,
                Image=image.image,
                ExternalImageId=image.image_name,
```

```
                MaxFaces=max_faces,
                DetectionAttributes=["ALL"],
            )
            indexed_faces = [
                RekognitionFace({**face["Face"], **face["FaceDetail"]})
                for face in response["FaceRecords"]
            ]
            unindexed_faces = [
                RekognitionFace(face["FaceDetail"])
                for face in response["UnindexedFaces"]
            ]
            logger.info(
                "Indexed %s faces in %s. Could not index %s faces.",
                len(indexed_faces),
                image.image_name,
                len(unindexed_faces),
            )
        except ClientError:
            logger.exception("Couldn't index faces in image %s.",
image.image_name)
            raise
        else:
            return indexed_faces, unindexed_faces
```

- Pour plus de détails sur l'API, consultez IndexFacesle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    TRY.
        " Create S3 object reference for the image
```

```
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Index faces in the image
        oo_result = lo_rek->indexfaces(
          iv_collectionid = iv_collection_id
          io_image = lo_image
          iv_externalimageid = iv_external_id
          iv_maxfaces = iv_max_faces ).

        DATA(lt_face_records) = oo_result->get_facerecords( ).
        DATA(lv_indexed_count) = lines( lt_face_records ).
        DATA(lv_msg2) = |{ lv_indexed_count } face(s) indexed successfully.|.
        MESSAGE lv_msg2 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Pour plus de détails sur l'API, consultez IndexFacesla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **`ListCollections`**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `ListCollections`.

Pour en savoir plus, consultez Répertoriage de collections.

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses Amazon Rekognition to list the collection IDs in the
/// current account.
/// </summary>
public class ListCollections
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        Console.WriteLine("Listing collections");
        int limit = 10;

        var listCollectionsRequest = new ListCollectionsRequest
        {
            MaxResults = limit,
        };

        var listCollectionsResponse = new ListCollectionsResponse();

        do
        {
            if (listCollectionsResponse is not null)
            {
                listCollectionsRequest.NextToken =
listCollectionsResponse.NextToken;
            }
```

```
                listCollectionsResponse = await
 rekognitionClient.ListCollectionsAsync(listCollectionsRequest);

                listCollectionsResponse.CollectionIds.ForEach(id =>
                {
                    Console.WriteLine(id);
                });
            }
            while (listCollectionsResponse.NextToken is not null);
        }
    }
```

- Pour plus de détails sur l'API, voir ListCollectionsla section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour répertorier les collections disponibles

La `list-collections` commande suivante répertorie les collections disponibles dans le AWS compte.

```
aws rekognition list-collections
```

Sortie :

```
{
    "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"
    ]
}
```

Pour plus d'informations, consultez Créer une liste de collections dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir ListCollectionsla section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListCollections {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Listing collections");
        listAllCollections(rekClient);
        rekClient.close();
    }

    public static void listAllCollections(RekognitionClient rekClient) {
        try {
            ListCollectionsRequest listCollectionsRequest =
 ListCollectionsRequest.builder()
                    .maxResults(10)
                    .build();

            ListCollectionsResponse response =
 rekClient.listCollections(listCollectionsRequest);
            List<String> collectionIds = response.collectionIds();
            for (String resultId : collectionIds) {
                System.out.println(resultId);
            }

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

- Pour plus de détails sur l'API, voir ListCollectionsla section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- Pour plus de détails sur l'API, consultez ListCollectionsla section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionCollectionManager:
```

```
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def list_collections(self, max_results):
        """
        Lists collections for the current account.

        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
            response =
self.rekognition_client.list_collections(MaxResults=max_results)
            collections = [
                RekognitionCollection({"CollectionId": col_id},
self.rekognition_client)
                for col_id in response["CollectionIds"]
            ]
        except ClientError:
            logger.exception("Couldn't list collections.")
            raise
        else:
            return collections
```

- Pour plus de détails sur l'API, consultez ListCollectionsle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
TRY.
    oo_result = lo_rek->listcollections(
      iv_maxresults = iv_max_results ).

    DATA(lt_collection_ids) = oo_result->get_collectionids( ).
    DATA(lv_coll_count) = lines( lt_collection_ids ).
    DATA(lv_msg7) = |{ lv_coll_count } collection(s) found.|.
    MESSAGE lv_msg7 TYPE 'I'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Pour plus de détails sur l'API, consultez ListCollectionsla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser `ListFaces`avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `ListFaces`.

Pour plus d'informations, consultez Répertoriage de visages d'une collection.

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to retrieve the list of faces
/// stored in a collection.
/// </summary>
public class ListFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection2";

        var rekognitionClient = new AmazonRekognitionClient();

        var listFacesResponse = new ListFacesResponse();
        Console.WriteLine($"Faces in collection {collectionId}");

        var listFacesRequest = new ListFacesRequest
        {
            CollectionId = collectionId,
            MaxResults = 1,
        };

        do
        {
            listFacesResponse = await
rekognitionClient.ListFacesAsync(listFacesRequest);
            listFacesResponse.Faces.ForEach(face =>
            {
```

```
                    Console.WriteLine(face.FaceId);
                });

                listFacesRequest.NextToken = listFacesResponse.NextToken;
            }
            while (!string.IsNullOrEmpty(listFacesResponse.NextToken));
        }
    }
```

- Pour plus de détails sur l'API, voir ListFacesla section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour répertorier les visages d'une collection

La commande `list-faces` suivante répertorie les visages dans la collection spécifiée.

```
aws rekognition list-faces \
    --collection-id MyCollection
```

Sortie :

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

Pour plus d'informations, consultez Création d'une liste de visages d'une collection dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir ListFacesla section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
```

```
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

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

                Usage:    <collectionId>

                Where:
                   collectionId - The name of the collection.\s
                """;

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

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

        System.out.println("Faces in collection " + collectionId);
        listFacesCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void listFacesCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            ListFacesRequest facesRequest = ListFacesRequest.builder()
                    .collectionId(collectionId)
```

```
                    .maxResults(10)
                    .build();

            ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
            List<Face> faces = facesResponse.faces();
            for (Face face : faces) {
                System.out.println("Confidence level there is a face: " +
 face.confidence());
                System.out.println("The face Id value is " + face.faceId());
            }

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

- Pour plus de détails sur l'API, voir ListFaceslа section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listFaces(request)
```

```
        response.faces?.forEach { face ->
            println("Confidence level there is a face: ${face.confidence}")
            println("The face Id value is ${face.faceId}")
        }
    }
}
```

- Pour plus de détails sur l'API, consultez ListFaceslа section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client
```

```
    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def list_faces(self, max_results):
        """
        Lists the faces currently indexed in the collection.

        :param max_results: The maximum number of faces to return.
        :return: The list of faces in the collection.
        """
        try:
            response = self.rekognition_client.list_faces(
                CollectionId=self.collection_id, MaxResults=max_results
            )
            faces = [RekognitionFace(face) for face in response["Faces"]]
            logger.info(
                "Found %s faces in collection %s.", len(faces),
self.collection_id
            )
        except ClientError:
            logger.exception(
                "Couldn't list faces in collection %s.", self.collection_id
            )
            raise
        else:
            return faces
```

- Pour plus de détails sur l'API, consultez ListFacesle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
TRY.
    oo_result = lo_rek->listfaces(
      iv_collectionid = iv_collection_id
      iv_maxresults = iv_max_results ).

    DATA(lt_faces) = oo_result->get_faces( ).
    DATA(lv_face_count2) = lines( lt_faces ).
    DATA(lv_msg3) = |{ lv_face_count2 } face(s) found in collection.|.
    MESSAGE lv_msg3 TYPE 'I'.
  CATCH /aws1/cx_rekresourcenotfoundex.
    MESSAGE 'Collection not found.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Pour plus de détails sur l'API, consultez ListFacesla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **`RecognizeCelebrities`**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `RecognizeCelebrities`.

Pour plus d'informations, consultez Reconnaissance de célébrités dans une image.

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to identify celebrities in a photo.
/// </summary>
public class CelebritiesInImage
{
    public static async Task Main(string[] args)
    {
        string photo = "moviestars.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        var recognizeCelebritiesRequest = new RecognizeCelebritiesRequest();

        var img = new Amazon.Rekognition.Model.Image();
        byte[] data = null;
        try
        {
            using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
            data = new byte[fs.Length];
            fs.Read(data, 0, (int)fs.Length);
        }
        catch (Exception)
        {
```

```
                Console.WriteLine($"Failed to load file {photo}");
                return;
            }

            img.Bytes = new MemoryStream(data);
            recognizeCelebritiesRequest.Image = img;

            Console.WriteLine($"Looking for celebrities in image {photo}\n");

            var recognizeCelebritiesResponse = await
rekognitionClient.RecognizeCelebritiesAsync(recognizeCelebritiesRequest);


Console.WriteLine($"{recognizeCelebritiesResponse.CelebrityFaces.Count}
celebrity(s) were recognized.\n");
            recognizeCelebritiesResponse.CelebrityFaces.ForEach(celeb =>
            {
                Console.WriteLine($"Celebrity recognized: {celeb.Name}");
                Console.WriteLine($"Celebrity ID: {celeb.Id}");
                BoundingBox boundingBox = celeb.Face.BoundingBox;
                Console.WriteLine($"position: {boundingBox.Left}
{boundingBox.Top}");
                Console.WriteLine("Further information (if available):");
                celeb.Urls.ForEach(url =>
                {
                    Console.WriteLine(url);
                });
            });


Console.WriteLine($"{recognizeCelebritiesResponse.UnrecognizedFaces.Count}
face(s) were unrecognized.");
        }
    }
```

- Pour plus de détails sur l'API, voir RecognizeCelebritiesla section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Reconnaître les célébrités sur une image

La commande `recognize-celebrities` suivante reconnaît les célébrités sur l'image spécifiée stockée dans un compartiment Amazon S3 :

```
aws rekognition recognize-celebrities \
    --image "S3Object={Bucket=MyImageS3Bucket,Name=moviestars.jpg}"
```

Sortie :

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

Pour plus d'informations, consultez Reconnaissance de célébrités sur une image dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir RecognizeCelebritiesla section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

import software.amazon.awssdk.services.rekognition.model.*;

/**
```

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

                Where:
                   bucketName - The name of the S3 bucket where the images are
 stored.
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

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

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

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

    /**
     * Recognizes all celebrities in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient    the Amazon Rekognition client used to perform the
 celebrity recognition operation
     * @param bucketName   the name of the Amazon S3 bucket where the source
 image is stored
```

```
     * @param sourceImage  the name of the source image file stored in the Amazon
 S3 bucket
     */
    public static void recognizeAllCelebrities(RekognitionClient rekClient,
 String bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
                .build();

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

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

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

                System.out.println("Further information (if available):");
                for (String url : celebrity.urls()) {
                    System.out.println(url);
                }
                System.out.println();
            }
            System.out.println(result.unrecognizedFaces().size() + " face(s) were
 unrecognized.");

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

- Pour plus de détails sur l'API, voir RecognizeCelebritiesla section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
        println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
    }
}
```

- Pour plus de détails sur l'API, consultez RecognizeCelebritiesla section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def recognize_celebrities(self):
        """
        Detects celebrities in the image.

        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
```

```
        try:
            response =
self.rekognition_client.recognize_celebrities(Image=self.image)
            celebrities = [
                RekognitionCelebrity(celeb) for celeb in
response["CelebrityFaces"]
            ]
            other_faces = [
                RekognitionFace(face) for face in response["UnrecognizedFaces"]
            ]
            logger.info(
                "Found %s celebrities and %s other faces in %s.",
                len(celebrities),
                len(other_faces),
                self.image_name,
            )
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.",
self.image_name)
            raise
        else:
            return celebrities, other_faces
```

- Pour plus de détails sur l'API, consultez RecognizeCelebritiesle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    TRY.
        " Create S3 object reference for the image
```

```
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Recognize celebrities
        oo_result = lo_rek->recognizecelebrities(
          io_image = lo_image ).

        DATA(lt_celebrity_faces) = oo_result->get_celebrityfaces( ).
        DATA(lv_celeb_count) = lines( lt_celebrity_faces ).
        DATA(lv_msg12) = |{ lv_celeb_count } celebrity/celebrities recognized.|.
        MESSAGE lv_msg12 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Pour plus de détails sur l'API, consultez RecognizeCelebritiesla section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **`SearchFaces`**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `SearchFaces`.

Pour plus d'informations, veuillez consulter Recherche d'un visage (identification faciale).

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to find faces in an image that
/// match the face Id provided in the method request.
/// </summary>
public class SearchFacesMatchingId
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        string faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";

        var rekognitionClient = new AmazonRekognitionClient();

        // Search collection for faces matching the face id.
        var searchFacesRequest = new SearchFacesRequest
        {
            CollectionId = collectionId,
            FaceId = faceId,
            FaceMatchThreshold = 70F,
            MaxFaces = 2,
        };

        SearchFacesResponse searchFacesResponse = await
rekognitionClient.SearchFacesAsync(searchFacesRequest);

        Console.WriteLine("Face matching faceId " + faceId);
```

```
            Console.WriteLine("Matche(s): ");
            searchFacesResponse.FaceMatches.ForEach(face =>
            {
                Console.WriteLine($"FaceId: {face.Face.FaceId} Similarity:
 {face.Similarity}");
            });
        }
    }
```

- Pour plus de détails sur l'API, voir SearchFacesla section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour rechercher les visages d'une collection qui correspondent à un ID de visage

La commande `search-faces` suivante recherche les visages d'une collection qui correspondent à l'ID de visage spécifié.

```
aws rekognition search-faces \
    --face-id 8d3cfc70-4ba8-4b36-9644-90fba29c2dac \
    --collection-id MyCollection
```

Sortie :

```
{
    "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
        }
    ]
}
```

Pour plus d'informations, consultez Recherche d'un visage à l'aide de son ID dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir SearchFaceslа section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

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

                Usage:    <collectionId> <sourceImage>

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

                """;

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

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

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }

    public static void searchFaceInCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(new
File(sourceImage));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
                    .image(souImage)
                    .maxFaces(10)
                    .faceMatchThreshold(70F)
                    .collectionId(collectionId)
                    .build();

            SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " +
face.similarity());
                System.out.println();
            }

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

```
    }
}
```

- Pour plus de détails sur l'API, voir SearchFaceslа section Référence des AWS SDK for Java 2.x API.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
```

```
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def search_faces(self, face_id, threshold, max_faces):
        """
        Searches for faces in the collection that match another face from the
        collection.

        :param face_id: The ID of the face in the collection to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: The list of matching faces found in the collection. This list
does
                 not contain the face specified by `face_id`.
        """
        try:
            response = self.rekognition_client.search_faces(
                CollectionId=self.collection_id,
                FaceId=face_id,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
            logger.info(
                "Found %s faces in %s that match %s.",
                len(faces),
                self.collection_id,
                face_id,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
```

```
                self.collection_id,
                face_id,
            )
            raise
        else:
            return faces
```

- Pour plus de détails sur l'API, consultez SearchFacesle AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   TRY.
       oo_result = lo_rek->searchfaces(
         iv_collectionid = iv_collection_id
         iv_faceid = iv_face_id
         iv_facematchthreshold = iv_threshold
         iv_maxfaces = iv_max_faces ).

       DATA(lt_face_matches) = oo_result->get_facematches( ).
       DATA(lv_match_count2) = lines( lt_face_matches ).
       DATA(lv_msg5) = |Face search completed: { lv_match_count2 } match(es)
found.|.
       MESSAGE lv_msg5 TYPE 'I'.
     CATCH /aws1/cx_rekresourcenotfoundex.
       MESSAGE 'Collection or face not found.' TYPE 'E'.
     CATCH /aws1/cx_rekinvalidparameterex.
       MESSAGE 'Invalid parameter value.' TYPE 'E'.
   ENDTRY.
```

- Pour plus de détails sur l'API, consultez SearchFaceslа section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **`SearchFacesByImage`**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `SearchFacesByImage`.

Pour plus d'informations, voir Recherche d'un visage (image).

.NET

SDK pour .NET

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to search for images matching those
/// in a collection.
/// </summary>
public class SearchFacesMatchingImage
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        string bucket = "amzn-s3-demo-bucket";
        string photo = "input.jpg";

        var rekognitionClient = new AmazonRekognitionClient();
```

```
            // Get an image object from S3 bucket.
            var image = new Image()
            {
                S3Object = new S3Object()
                {
                    Bucket = bucket,
                    Name = photo,
                },
            };

            var searchFacesByImageRequest = new SearchFacesByImageRequest()
            {
                CollectionId = collectionId,
                Image = image,
                FaceMatchThreshold = 70F,
                MaxFaces = 2,
            };

            SearchFacesByImageResponse searchFacesByImageResponse = await
rekognitionClient.SearchFacesByImageAsync(searchFacesByImageRequest);

            Console.WriteLine("Faces matching largest face in image from " +
photo);
            searchFacesByImageResponse.FaceMatches.ForEach(face =>
            {
                Console.WriteLine($"FaceId: {face.Face.FaceId}, Similarity:
{face.Similarity}");
            });
        }
    }
```

- Pour plus de détails sur l'API, voir SearchFacesByImagela section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour rechercher les visages d’une collection qui correspondent au visage le plus grand d’une image

La commande `search-faces-by-image` suivante recherche les visages d'une collection qui correspondent au visage le plus grand de l'image spécifiée :

```
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"
}
```

Pour plus d'informations, consultez Recherche d'un visage à l'aide de son image dans le Guide du développeur Amazon Rekognition.

- Pour plus de détails sur l'API, voir SearchFacesByImagela section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

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

                Usage:    <collectionId> <sourceImage>

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

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

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

        System.out.println("Searching for a face in a collections");
        searchFacebyId(rekClient, collectionId, faceId);
```

```
        rekClient.close();
    }

    public static void searchFacebyId(RekognitionClient rekClient, String
 collectionId, String faceId) {
        try {
            SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceId(faceId)
                    .faceMatchThreshold(70F)
                    .maxFaces(2)
                    .build();

            SearchFacesResponse imageResponse =
 rekClient.searchFaces(searchFacesRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " +
 face.similarity());
                System.out.println();
            }

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

- Pour plus de détails sur l'API, voir SearchFacesByImagela section Référence des AWS SDK for Java 2.x API.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.

        :param image: The image that contains the reference face to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: A tuple. The first element is the face found in the reference
image.
                 The second element is the list of matching faces found in the
                 collection.
        """
        try:
            response = self.rekognition_client.search_faces_by_image(
                CollectionId=self.collection_id,
                Image=image.image,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            image_face = RekognitionFace(
                {
                    "BoundingBox": response["SearchedFaceBoundingBox"],
                    "Confidence": response["SearchedFaceConfidence"],
                }
            )
            collection_faces = [
                RekognitionFace(face["Face"]) for face in response["FaceMatches"]
            ]
            logger.info(
                "Found %s faces in the collection that match the largest "
                "face in %s.",
                len(collection_faces),
                image.image_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
```

```
                self.collection_id,
                image.image_name,
            )
            raise
        else:
            return image_face, collection_faces
```

- Pour plus de détails sur l'API, consultez SearchFacesByImagele AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Search for matching faces
        oo_result = lo_rek->searchfacesbyimage(
          iv_collectionid = iv_collection_id
          io_image = lo_image
          iv_facematchthreshold = iv_threshold
          iv_maxfaces = iv_max_faces ).

        DATA(lt_face_matches) = oo_result->get_facematches( ).
        DATA(lv_match_count) = lines( lt_face_matches ).
```

```
        DATA(lv_msg4) = |Face search completed: { lv_match_count } match(es)
found.|.
        MESSAGE lv_msg4 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Pour plus de détails sur l'API, consultez SearchFacesByImagela section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Scénarios d'utilisation d'Amazon Rekognition AWS Kits SDK

Les exemples de code suivants vous montrent comment implémenter des scénarios courants dans Amazon Rekognition à l'aide de kits AWS SDK. Ces scénarios vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions dans Amazon Rekognition ou en les combinant avec d'autres Services AWS. Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code.

Les scénarios ciblent un niveau d'expérience intermédiaire pour vous aider à comprendre les actions de service dans leur contexte.

Exemples

- Créez une collection Amazon Rekognition et retrouvez-y des visages à l'aide d'un AWS Kit SDK
- Création d'une application de gestion des ressources photographiques permettant aux utilisateurs de gérer les photos à l'aide d'étiquettes
- Détectez le PPE dans les images avec Amazon Rekognition à l'aide d'un AWS Kit SDK
- Détectez et affichez des éléments dans des images avec Amazon Rekognition à l'aide d'un AWS Kit SDK
- Détecter les visages dans une image à l'aide d'un AWS Kit SDK

- Détectez les informations contenues dans les vidéos à l'aide d'Amazon Rekognition et du AWS Kit SDK
- Détectez des objets dans des images avec Amazon Rekognition à l'aide d'un AWS Kit SDK
- Détectez les personnes et les objets dans une vidéo avec Amazon Rekognition à l'aide d'un AWS Kit SDK
- Enregistrez les informations EXIF et autres informations sur l'image à l'aide d'un AWS Kit SDK

Créez une collection Amazon Rekognition et retrouvez-y des visages à l'aide d'un AWS Kit SDK

L'exemple de code suivant illustre comment :

- Créer une collection Amazon Rekognition.
- Ajouter des images à la collection et détecter les visages qu'elle contient.
- Rechercher dans la collection les visages qui correspondent à une image de référence.
- Supprimer une collection.

Pour plus d'informations, veuillez consulter Recherche de visages dans une collection.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez des classes qui encapsulent les fonctions 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
```

Utilisez les classes wrapper pour créer une collection de visages à partir d’un ensemble d’images, puis recherchez des visages dans la collection.

```
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)
```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Création d'une application de gestion des ressources photographiques permettant aux utilisateurs de gérer les photos à l'aide d'étiquettes

Les exemples de code suivants montrent comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

.NET

SDK pour .NET

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK pour C++

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK pour Java 2.x

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK pour JavaScript (v3)

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK pour Kotlin

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

Kit SDK pour PHP

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK pour Rust

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur AWS Community.

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Détectez le PPE dans les images avec Amazon Rekognition à l'aide d'un AWS Kit SDK

Les exemples de code suivants montrent comment créer une application qui utilise Amazon Rekognition afin de détecter l'équipement de protection individuelle (EPI) dans les images.

Java

SDK pour Java 2.x

Montre comment créer une AWS Lambda fonction qui détecte les images à l'aide d'un équipement de protection individuelle.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Détectez et affichez des éléments dans des images avec Amazon Rekognition à l'aide d'un AWS Kit SDK

L'exemple de code suivant illustre comment :

- Détecter des éléments dans des images à l'aide d'Amazon Rekognition.
- Afficher des images et tracer des cadres autour des éléments détectés.

Pour plus d'informations, veuillez consulter Affichage des cadres de délimitation.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez des classes pour encapsuler les fonctions 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
```

Créez des fonctions d’assistance pour dessiner des cadres de délimitation et des polygones.

```
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()
```

Créer des classes pour analyser les objets renvoyés par 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
```

Utilisez les classes wrapper pour détecter des éléments dans les images et afficher leurs cadres de délimitation. Les images utilisées dans cet exemple se trouvent ici, GitHub ainsi que des instructions et du code supplémentaire.

```
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)
```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Détecter les visages dans une image à l'aide d'un AWS Kit SDK

L'exemple de code suivant illustre comment :

- Enregistrez une image dans un compartiment Amazon S3.
- Utilisez Amazon Rekognition pour détecter les détails du visage, tels que la tranche d'âge, le sexe et l'émotion (sourire, etc.).
- Affichez ces détails.

Rust

SDK pour Rust

Enregistrez l'image dans un compartiment Amazon S3 avec un préfixe uploads (chargement), utilisez Amazon Rekognition pour détecter les détails du visage, tels que la tranche d'âge, le sexe et l'émotion (sourire, etc.) et affichez ces détails.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Détectez les informations contenues dans les vidéos à l'aide d'Amazon Rekognition et du AWS Kit SDK

Les exemples de code suivants montrent comment :

- Lancer des tâches sur Amazon Rekognition pour détecter des éléments tels que des personnes, des objets et du texte dans des vidéos.
- Vérifier l'état de la tâche jusqu'à ce qu'elle soit terminée.
- Afficher la liste des éléments détectés par chaque tâche.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Obtenez des informations sur des célébrités à partir d'une vidéo située dans un compartiment 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);
        }
    }
}
```

Détectez les étiquettes dans une vidéo par une opération de détection d'étiquettes.

```
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);
        }
    }
}
```

Détectez des visages dans une vidéo stockée dans un compartiment 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);
        }
    }
}
```

Détectez un contenu inapproprié ou offensant dans une vidéo stockée dans un compartiment 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);
        }
    }
}
```

Détectez les segments de repères techniques et les segments de détection de prises de vue dans une vidéo stockée dans un compartiment 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);
        }
    }
}
```

Détectez le texte dans une vidéo stockée dans un compartiment 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);
        }
    }
}
```

Détectez des personnes dans une vidéo stockée dans un compartiment 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);
        }
    }
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK for Java 2.x .
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection

- StartTextDetection

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Détectez des visages dans une vidéo stockée dans un compartiment 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}")
        }
    }
}
```

Détectez un contenu inapproprié ou offensant dans une vidéo stockée dans un compartiment 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)
        }
    }
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la AWS Référence de l’API de SDK pour Kotlin.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition

- StartContentModeration
- StartLabelDetection
- StartPersonTracking
- StartSegmentDetection
- StartTextDetection

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Détectez des objets dans des images avec Amazon Rekognition à l'aide d'un AWS Kit SDK

Les exemples de code suivants montrent comment créer une application qui utilise Amazon Rekognition afin de détecter des objets par catégorie dans des images.

.NET

SDK pour .NET

Montre comment utiliser l'API Java Amazon Rekognition afin de créer une application qui, avec Amazon Rekognition, permet d'identifier des objets par catégorie dans des images stockées dans un compartiment Amazon Simple Storage Service (Amazon S3). L'application envoie à l'administrateur une notification par e-mail contenant les résultats à l'aide d'Amazon Simple Email Service (Amazon SES).

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK pour Java 2.x

Montre comment utiliser l'API Java Amazon Rekognition afin de créer une application qui, avec Amazon Rekognition, permet d'identifier des objets par catégorie dans des images stockées dans un compartiment Amazon Simple Storage Service (Amazon S3). L'application envoie à l'administrateur une notification par e-mail contenant les résultats à l'aide d'Amazon Simple Email Service (Amazon SES).

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK pour JavaScript (v3)

Montre comment utiliser Amazon Rekognition AWS SDK pour JavaScript pour créer une application qui utilise Amazon Rekognition pour identifier les objets par catégorie dans des images situées dans un compartiment Amazon Simple Storage Service (Amazon S3). L'application envoie à l'administrateur une notification par e-mail contenant les résultats à l'aide d'Amazon Simple Email Service (Amazon SES).

Découvrez comment :

- Créer un utilisateur non authentifié à l'aide d'Amazon Cognito.
- Analyser les images à la recherche d'objets à l'aide d'Amazon Rekognition.
- Vérifier une adresse e-mail pour Amazon SES.
- Envoyer une notification par e-mail à l'aide d'Amazon SES.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK pour Kotlin

Montre comment utiliser l'API Kotlin Amazon Rekognition afin de créer une application qui, avec Amazon Rekognition, permet d'identifier des objets par catégorie dans des images stockées dans un compartiment Amazon Simple Storage Service (Amazon S3). L'application envoie à l'administrateur une notification par e-mail contenant les résultats à l'aide d'Amazon Simple Email Service (Amazon SES).

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK pour Python (Boto3)

Vous montre comment utiliser le AWS SDK pour Python (Boto3) pour créer une application Web qui vous permet d'effectuer les opérations suivantes :

- Chargez les photos dans un compartiment Amazon Simple Storage Service (Amazon S3).
- Utilisez Amazon Rekognition pour analyser et étiqueter les photos.
- Utilisez Amazon Simple Email Service (Amazon SES) pour envoyer des rapports d'analyse d'images par e-mail.

Cet exemple contient deux composants principaux : une page Web écrite avec React et un service REST écrit en Python construit avec Flask-RESTful. JavaScript

Vous pouvez utiliser la page web React pour :

- Affichez une liste d'images stockées dans votre compartiment S3.
- Chargez des images depuis votre ordinateur dans votre compartiment S3.
- Affichez des images et des étiquettes qui identifient les éléments détectés dans l'image.
- Obtenez un rapport de toutes les images de votre compartiment S3 et envoyez un e-mail du rapport.

La page web appelle le service REST. Le service envoie des demandes à AWS pour effectuer les opérations suivantes :

- Obtenez et filtrez la liste des images de votre compartiment S3.
- Chargez des photos dans votre compartiment S3.
- Utilisez Amazon Rekognition pour analyser des photos individuelles et obtenir une liste d'étiquettes qui identifient les éléments détectés sur la photo.
- Analysez toutes les photos de votre compartiment S3 et utilisez Amazon SES pour envoyer un rapport par e-mail.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Détectez les personnes et les objets dans une vidéo avec Amazon Rekognition à l'aide d'un AWS Kit SDK

Les exemples de code suivants montrent comment détecter des personnes et des objets dans une vidéo avec Amazon Rekognition.

Java

SDK pour Java 2.x

Montre comment utiliser l'API Java Amazon Rekognition afin de créer une application qui détecte les visages et les objets dans des vidéos stockées dans un compartiment Amazon Simple Storage Service (Amazon S3). L'application envoie à l'administrateur une notification par e-mail contenant les résultats à l'aide d'Amazon Simple Email Service (Amazon SES).

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK pour Python (Boto3)

Utilisez Amazon Rekognition pour détecter des visages, des objets et des personnes dans des vidéos en démarrant des tâches de détection asynchrone. Cet exemple montre également comment configurer Amazon Rekognition pour notifier une rubrique Amazon Simple Notification Service (Amazon SNS) lorsque les tâches sont terminées et abonner une file d'attente Amazon Simple Queue Service (Amazon SQS) à la rubrique. Lorsque la file d'attente reçoit un message concernant une tâche, elle est récupérée et les résultats sont affichés.

Il est préférable de visionner cet exemple sur GitHub. Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS

- Amazon SQS

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Enregistrez les informations EXIF et autres informations sur l'image à l'aide d'un AWS Kit SDK

L'exemple de code suivant illustre comment :

- Obtenir des informations EXIF à partir d'un fichier JPG, JPEG ou PNG.
- Charger le fichier image sur un compartiment Amazon S3.
- Utiliser Amazon Rekognition pour identifier les trois principaux attributs (étiquettes) dans le fichier.
- Ajouter les informations EXIF et les étiquettes à un tableau Amazon DynamoDB dans la région.

Rust

SDK pour Rust

Obtenez les informations EXIF à partir d'un fichier JPG, JPEG ou PNG, chargez le fichier image dans un compartiment Amazon S3, utilisez Amazon Rekognition pour identifier les trois principaux attributs (étiquettes dans Amazon Rekognition) du fichier et ajoutez les informations EXIF et d'étiquettes à un tableau Amazon DynamoDB dans la région.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- DynamoDB
- Amazon Rekognition
- Amazon S3

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtiliser Rekognition avec un AWS Kit SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Référence d’API

La référence de l'API Amazon Rekognition se trouve désormais sur le site Référence API Amazon Rekognition.

Sécurité Amazon Rekognition

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

Consultez les rubriques suivantes pour apprendre à sécuriser vos ressources Amazon Rekognition.

Rubriques

- Gestion des identités et des accès pour Amazon Rekognition
- Protection des données dans Amazon Rekognition
- Utilisation d'Amazon Rekognition avec les points de terminaison d'un VPC Amazon
- Validation de la conformité pour Amazon Rekognition
- Résilience dans Amazon Rekognition
- Configuration et analyse des vulnérabilités dans Amazon Rekognition
- Cross-service prévention confuse des adjoints
- Sécurité de l'infrastructure dans Amazon Rekognition

Gestion des identités et des accès pour Amazon Rekognition

Gestion des identités et des accès AWS (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Des administrateurs IAM contrôlent les personnes qui peuvent être authentifiées (connectées) et autorisées (disposant d'autorisations) à utiliser des ressources Amazon Rekognition. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- Public ciblé
- Authentification par des identités
- Gestion de l'accès à l'aide de politiques
- Fonctionnement d'Amazon Rekognition avec IAM
- AWS politiques gérées pour Amazon Rekognition
- Bonnes pratiques en matière de politiques
- Utilisation de la console Amazon Rekognition

- Exemples d'utilisation d'Amazon Rekognition Exemples de politiques basées sur l'identité
- Exemples de politique basée sur les ressources Amazon Rekognition
- Résolution de problèmes pour identité et accès Amazon Rekognition

Public ciblé

La façon dont vous utilisez Gestion des identités et des accès AWS (IAM) varie en fonction de votre rôle :

- Utilisateur du service : demandez des autorisations à votre administrateur si vous ne pouvez pas accéder aux fonctionnalités (voir Résolution de problèmes pour identité et accès Amazon Rekognition)
- Administrateur du service : déterminez l'accès des utilisateurs et soumettez les demandes d'autorisation (voir Fonctionnement d'Amazon Rekognition avec IAM)
- Administrateur IAM : rédigez des politiques pour gérer l'accès (voir Exemples d'utilisation d'Amazon Rekognition Exemples de politiques basées sur l'identité)

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en tant qu'identité fédérée à l'aide d'informations d'identification provenant d'une source d'identité telle que AWS IAM Identity Center (IAM Identity Center), d'une authentification unique ou d'informations d'identification. Google/Facebook Pour plus d'informations sur la connexion, consultez Connexion à votre Compte AWS dans le Guide de l'utilisateur Connexion à AWS .

Pour l'accès par programmation, AWS fournit un SDK et une CLI pour signer les demandes de manière cryptographique. Pour plus d'informations, consultez Signature AWS Version 4 pour les demandes d'API dans le Guide de l'utilisateur IAM.

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité qui dispose d'autorisations spécifiques pour une seule personne ou application. Nous vous recommandons d'utiliser ces informations d'identification temporaires au

lieu des utilisateurs IAM avec des informations d'identification à long terme. Pour plus d'informations, voir Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires dans le guide de l'utilisateur IAM.

Les groupes IAM spécifient une collection d'utilisateurs IAM et permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Pour plus d'informations, consultez Cas d'utilisation pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité dotée d'autorisations spécifiques qui fournit des informations d'identification temporaires. Vous pouvez assumer un rôle en passant d'un rôle d'utilisateur à un rôle IAM (console) ou en appelant une opération d' AWS API AWS CLI ou d'API. Pour plus d'informations, consultez Méthodes pour endosser un rôle dans le Guide de l'utilisateur IAM.

Les rôles IAM sont utiles pour l'accès des utilisateurs fédérés, les autorisations temporaires des utilisateurs IAM, les accès intercompte, les accès entre services et les applications exécutées sur Amazon EC2. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Gestion de l'accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique définit les autorisations lorsqu'elles sont associées à une identité ou à une ressource. AWS évalue ces politiques lorsqu'un directeur fait une demande. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations les documents de politique JSON, consultez Vue d'ensemble des politiques JSON dans le Guide de l'utilisateur IAM.

À l'aide de politiques, les administrateurs précisent qui a accès à quoi en définissant quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Un administrateur IAM crée des politiques IAM et les ajoute aux rôles, que les utilisateurs peuvent ensuite assumer. Les politiques IAM définissent les autorisations quelle que soit la méthode que vous utilisez pour exécuter l'opération.

Utilisation de politiques basées sur l'identité

Identity-based les politiques sont des documents de politique d'autorisation JSON que vous attachez à une identité (utilisateur, groupe ou rôle). Ces politiques contrôlent les actions que peuvent exécuter ces identités, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Définition d'autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l'utilisateur IAM.

Identity-based les politiques peuvent être des politiques intégrées (intégrées directement dans une seule identité) ou des politiques gérées (politiques autonomes associées à plusieurs identités). Pour découvrir comment choisir entre des politiques gérées et en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

Utilisation de stratégies basées sur une ressource

Resource-based les politiques sont des documents de politique JSON que vous attachez à une ressource. Les exemples incluent les politiques de confiance de rôle IAM et les stratégies de compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Vous devez spécifier un principal dans une politique basée sur les ressources.

Resource-based les politiques sont des politiques intégrées qui se trouvent dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACL)

Les listes de contrôle d'accès (ACL) vérifie quels principaux (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et Amazon VPC sont des exemples de services qui prennent en charge les ACL. AWS WAF Pour en savoir plus sur les listes de contrôle d'accès, consultez Vue d'ensemble des listes de contrôle d'accès (ACL) dans le Guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge des types de politiques supplémentaires qui peuvent définir les autorisations maximales accordées par les types de politiques les plus courants :

- Limites d'autorisations : une limite des autorisations définit le nombre maximum d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM. Pour plus d'informations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.
- Politiques de contrôle des services (SCP) : spécifient les autorisations maximales pour une organisation ou une unité organisationnelle dans AWS Organizations. Pour plus d'informations, consultez Politiques de contrôle de service dans le Guide de l'utilisateur AWS Organizations .
- Politiques de contrôle des ressources (RCP) : définissent les autorisations maximales disponibles pour les ressources de votre organisation. Pour plus d'informations, consultez Politiques de contrôle des ressources (RCP) dans le Guide de l'utilisateur AWS Organizations .
- Politiques de session : politiques avancées que vous passez en tant que paramètre lorsque vous créez par programmation une session temporaire pour un rôle ou un utilisateur fédéré. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Fonctionnement d'Amazon Rekognition avec IAM

Avant d'utiliser IAM pour gérer l'accès à Amazon Rekognition, vous devez comprendre quelles sont les fonctionnalités IAM qui peuvent être utilisées avec Amazon Rekognition. Pour obtenir une vue d'ensemble de la manière dont Amazon Rekognition AWS et les autres services fonctionnent avec IAM AWS , consultez la section Services That Work with IAM dans le guide de l'utilisateur IAM.

Rubriques

- Politiques basées sur l'identité Amazon Rekognition
- Politiques basées sur les ressources Amazon Rekognition
- Rôles IAM Amazon Rekognition

Politiques basées sur l'identité Amazon Rekognition

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou

refusées. Amazon Rekognition prend en charge des actions, des ressources et des clés de condition spécifiques. Pour en savoir plus sur tous les éléments que vous utilisez dans une politique JSON, veuillez consulter Références des éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

Actions

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Les actions de politique dans Amazon Rekognition utilisent le préfixe suivant avant l'action : `rekognition:`. Par exemple, pour accorder à quelqu'un l'autorisation de détecter des objets, des scènes ou des concepts dans une image avec l'opération d'API `DetectLabels` Amazon Rekognition, vous incluez l'action `rekognition:DetectLabels` dans sa stratégie. Les déclarations de politique doivent inclure un élément `Action` ou `NotAction`. Amazon Rekognition définit son propre ensemble d'actions qui décrivent les tâches que vous pouvez effectuer avec ce service.

Pour spécifier plusieurs actions dans une seule déclaration, séparez-les par des virgules comme suit :

```
"Action": [
      "rekognition:action1",
      "rekognition:action2"
```

Vous pouvez aussi spécifier plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `Describe`, incluez l'action suivante :

```
"Action": "rekognition:Describe*"
```

Pour afficher la liste des actions Amazon Rekognition, consultez Actions définies par Amazon Rekognition dans le Guide de l'utilisateur IAM.

Ressources

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

Pour plus d'informations sur le format des ARN, consultez Amazon Resource Names (ARN) et AWS Service Namespaces.

Par exemple, pour spécifier la collection `MyCollection` dans votre instruction, utilisez l'ARN suivant :

```
"Resource": "arn:aws:rekognition:us-east-1:123456789012:collection/MyCollection"
```

Pour spécifier toutes les instances qui appartiennent à un compte spécifique, utilisez le caractère générique (*) :

```
"Resource": "arn:aws:rekognition:us-east-1:123456789012:collection/*"
```

Certaines actions Amazon Rekognition, telles que celles destinées à la création de ressources, ne peuvent pas être exécutées sur une ressource spécifique. Dans ces cas-là, vous devez utiliser le caractère générique (*).

```
"Resource": "*"
```

Pour afficher une liste des types de ressources Amazon Rekognition et de leurs ARN, veuillez consulter Ressources définies par Amazon Rekognition dans le Guide de l'utilisateur IAM. Pour savoir les actions avec lesquelles vous pouvez spécifier l'ARN de chaque ressource, consultez Actions définies par Amazon Rekognition.

Clés de condition

Amazon Rekognition ne fournit pas de clés de condition spécifiques au service, mais prend en charge l'utilisation de certaines clés de condition globales. Pour voir toutes les clés de condition AWS globales, consultez la section Clés contextuelles de condition AWS globale dans le guide de l'utilisateur IAM.

Politiques basées sur les ressources Amazon Rekognition

Amazon Rekognition ne prend en charge que les politiques basées sur les ressources pour la copie des modèles d'étiquettes personnalisées. Pour de plus amples informations, veuillez consulter Exemples de stratégies basées sur les ressources Amazon Rekognition.

D'autres services, tels qu'Amazon S3, prennent également en charge les politiques d'autorisation basées sur une ressource. Par exemple, vous pouvez attacher une politique à un compartiment S3 pour gérer les autorisations d'accès à ce compartiment.

Pour accéder à des images stockées dans un compartiment Amazon S3, vous devez être autorisé à accéder à l'objet de ce compartiment S3. Grâce à cette autorisation, Amazon Rekognition peut télécharger des images à partir du compartiment S3. L'exemple de politique suivant permet à l'utilisateur d'effectuer l'`s3:GetObject`action sur le compartiment S3 nommé amzn-s3-demo-bucket3.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket3/*"
            ]
        }
    ]
}
```

Pour utiliser un compartiment S3 avec la gestion des versions activée, ajoutez l'action `s3:GetObjectVersion`, comme indiqué dans l'exemple suivant.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket3/*"
            ]
        }
    ]
}
```

Rôles IAM Amazon Rekognition

Un rôle IAM est une entité de votre AWS compte qui possède des autorisations spécifiques.

Utilisation d'informations d'identification temporaires avec Amazon Rekognition

Vous pouvez utiliser des informations d'identification temporaires pour vous connecter à l'aide de la fédération, endosser un rôle IAM ou encore pour endosser un rôle intercompte. Vous obtenez des informations d'identification de sécurité temporaires en appelant des opérations d' AWS STS API telles que AssumeRoleou GetFederationToken.

Amazon Rekognition prend en charge l'utilisation d'informations d'identification temporaires.

Service-linked rôles

Service-linked les rôles permettent aux AWS services d'accéder aux ressources d'autres services pour effectuer une action en votre nom. Service-linked les rôles apparaissent dans votre compte IAM et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Amazon Rekognition ne prend pas en charge les fonctions liées à un service.

Fonctions du service

Cette fonction permet à un service d'endosser une fonction du service en votre nom. Ce rôle autorise le service à accéder à des ressources d'autres services pour effectuer une action en votre nom. Les rôles de service s'affichent dans votre compte IAM et sont la propriété du compte. Cela signifie qu'un administrateur IAM peut modifier les autorisations associées à ce rôle. Toutefois, une telle action peut perturber le bon fonctionnement du service.

Amazon Rekognition prend en charge les fonctions du service.

L'utilisation d'une fonction du service peut créer un problème de sécurité dans lequel Amazon Rekognition est utilisé pour appeler un autre service et agir sur des ressources auxquelles il ne devrait pas avoir accès. Pour garantir la sécurité de votre compte, vous devez limiter l'accès d'Amazon Rekognition aux seules ressources que vous utilisez. Pour ce faire, attachez une politique de confiance à votre fonction du service IAM. Pour plus d'informations sur la procédure à utiliser, consultez Cross-service prévention confuse des adjoints.

Choix d'un rôle IAM dans Amazon Rekognition

Lorsque vous configurez Amazon Rekognition pour analyser des vidéos stockées, vous devez choisir un rôle pour autoriser Amazon Rekognition à accéder à Amazon SNS en votre nom. Si vous avez déjà créé une fonction du service ou un rôle lié à un service, Amazon Rekognition vous fournit une liste de rôles dans laquelle effectuer votre choix. Pour de plus amples informations, veuillez consulter the section called "Configuration de Vidéo Amazon Rekognition".

Exemple : Configuration d'Amazon Rekognition pour accéder aux images d'un compartiment Amazon S3

Voici un exemple de la manière dont vous pouvez configurer Amazon Rekognition pour analyser des images dans un compartiment Amazon S3. Si vous souhaitez utiliser Amazon Rekognition pour analyser des images dans un compartiment Amazon S3, vous devez effectuer les opérations suivantes :

1. Assurez-vous que votre IAM user/role (le client) est autorisé à appeler les opérations d'API Amazon Rekognition pertinentes (comme, etc.) DetectLabels DetectFaces

   Joignez une politique basée sur l'identité qui accorde les autorisations appropriées pour appeler les opérations d'API souhaitées. Par exemple, pour autoriser votre rôle à appeler `DetectLabels` et`DetectFaces`, vous devez associer à votre rôle une politique semblable à celle-ci :

JSON

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

2. Le service Amazon Rekognition a besoin d'une autorisation pour accéder à votre compartiment Amazon S3. Créez un rôle de service IAM, que vous devrez transmettre à Amazon Rekognition lorsque vous effectuez des appels d'API. Assurez-vous que le rôle de service : fait confiance au principal `s3:GetObject` du service Amazon Rekognition et dispose d'autorisations pour votre compartiment.

   La politique de confiance peut ressembler à ceci :

   JSON

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

La politique basée sur l'identité attachée au rôle de service peut ressembler à ceci :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

AWS politiques gérées pour Amazon Rekognition

Pour ajouter des autorisations aux utilisateurs, aux groupes et aux rôles, il est plus facile d'utiliser des politiques AWS gérées que de les rédiger vous-même. Il faut du temps et de l'expertise pour créer des politiques gérées par le client IAM qui ne fournissent à votre équipe que les autorisations dont elle a besoin. Pour démarrer rapidement, vous pouvez utiliser nos politiques AWS gérées. Ces politiques couvrent les cas d'utilisation courants et sont disponibles dans votre AWS compte. Pour plus d'informations sur les politiques AWS gérées, voir les politiques AWS gérées dans le guide de l'utilisateur IAM.

AWS les services maintiennent et mettent à jour les politiques AWS gérées. Vous ne pouvez pas modifier les autorisations dans les politiques AWS gérées. Les services ajoutent occasionnellement des autorisations à une politique gérée par AWS pour prendre en charge de nouvelles fonctionnalités. Ce type de mise à jour affecte toutes les identités (utilisateurs, groupes et rôles) auxquelles la politique est attachée. Les services sont très susceptibles de mettre à jour une politique gérée par AWS quand une nouvelle fonctionnalité est lancée ou quand de nouvelles opérations sont disponibles. Les services ne suppriment pas les autorisations d'une politique AWS gérée. Les mises à jour des politiques n'endommageront donc pas vos autorisations existantes.

En outre, AWS prend en charge les politiques gérées pour les fonctions professionnelles qui couvrent plusieurs services. Par exemple, la politique ReadOnlyAccess AWS gérée fournit un accès en lecture

seule à tous les AWS services et ressources. Lorsqu'un service lance une nouvelle fonctionnalité, il AWS ajoute des autorisations en lecture seule pour les nouvelles opérations et ressources. Pour obtenir la liste des politiques de fonctions professionnelles et leurs descriptions, consultez la page politiques gérées par AWS pour les fonctions de tâche dans le Guide de l'utilisateur IAM.

Politique gérée par AWS : AmazonRekognitionFullAccess

`AmazonRekognitionFullAccess` octroie un accès total aux ressources Amazon Rekognition comprenant la création et la suppression des collections.

Vous pouvez associer la politique `AmazonRekognitionFullAccess` à vos identités IAM.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

JSON

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

Politique gérée par AWS : AmazonRekognitionReadOnlyAccess

`AmazonRekognitionReadOnlyAccess` accorde un accès en lecture seule aux ressources Amazon Rekognition.

Vous pouvez associer la politique `AmazonRekognitionReadOnlyAccess` à vos identités IAM.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

JSON

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

Politique gérée par AWS : AmazonRekognitionServiceRole

`AmazonRekognitionServiceRole` permet à Amazon Rekognition d'appeler les services Amazon Kinesis Data Streams et Amazon SNS en votre nom.

Vous pouvez associer la politique `AmazonRekognitionServiceRole` à vos identités IAM.

Si vous utilisez cette fonction du service, vous devez protéger votre compte en limitant l'accès d'Amazon Rekognition aux seules ressources que vous utilisez. Pour ce faire, attachez une politique de confiance à votre fonction du service IAM. Pour plus d'informations sur la procédure à utiliser, consultez Cross-service prévention confuse des adjoints.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

JSON

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

Politique gérée par AWS : AmazonRekognitionCustomLabelsFullAccess

Cette politique s’adresse aux utilisateurs d’Étiquettes personnalisées Amazon Rekognition. Utilisez cette AmazonRekognitionCustomLabelsFullAccess politique pour accorder aux utilisateurs un accès complet à l'API Amazon Rekognition Custom Labels et un accès complet aux compartiments de console créés par la console Amazon Rekognition Custom Labels.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetObject",
```

```
                "s3:GetObjectAcl",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::*custom-labels*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:CopyProjectVersion",
                "rekognition:CreateProject",
                "rekognition:CreateProjectVersion",
                "rekognition:StartProjectVersion",
                "rekognition:StopProjectVersion",
                "rekognition:DescribeProjects",
                "rekognition:DescribeProjectVersions",
                "rekognition:DetectCustomLabels",
                "rekognition:DeleteProject",
                "rekognition:DeleteProjectVersion",
                "rekognition:TagResource",
                "rekognition:UntagResource",
                "rekognition:ListTagsForResource",
                "rekognition:CreateDataset",
                "rekognition:ListDatasetEntries",
                "rekognition:ListDatasetLabels",
                "rekognition:DescribeDataset",
                "rekognition:UpdateDatasetEntries",
                "rekognition:DistributeDatasetEntries",
                "rekognition:DeleteDataset",
                "rekognition:PutProjectPolicy",
                "rekognition:ListProjectPolicies",
                "rekognition:DeleteProjectPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Rekognition met à jour AWS stratégies gérées

Consultez les informations relatives aux mises à jour des politiques AWS gérées pour Amazon Rekognition depuis que ce service a commencé à suivre ces modifications. Pour recevoir des alertes automatiques sur les modifications apportées à cette page, abonnez-vous au flux RSS de la page Historique des documents d'Amazon Rekognition.

Modifier	Description	Date
Les actions impliquant des tâches d'analyse des médias ont été ajoutées à la politique gérée suivante : • Politique gérée par AWS : AmazonRekognitionReadOnlyAccess	Amazon Rekognition a ajouté les actions suivantes à la politique gérée par `AmazonRekognitionReadOnlyAccess` : • `GetMediaAnalysisJob` • `ListMediaAnalysisJob`	31 octobre 2023
Les actions impliquant la gestion des utilisateurs ont été ajoutées à la politique gérée suivante : • Politique gérée par AWS : AmazonRekognitionReadOnlyAccess	Amazon Rekognition a ajouté les actions suivantes à la politique gérée par `AmazonRekognitionReadOnlyAccess` : • `ListUsers` • `SearchUsers` • `SearchUsersByImage`	12 juin 2023
Les actions ProjectPolicy et la copie du modèle d'étiquettes personnalisées ont été ajoutées aux politiques gérées suivantes : • Politique gérée par AWS : AmazonRekognitionFullAccess	Amazon Rekognition a ajouté les actions suivantes aux politiques gérées par `AmazonRekognitionCustomLabelsFullAccess` et `AmazonRekognitionFullAccess` : • `CopyProjectVersion` • `PutProjectPolicy`	21 juillet 2022

Modifier	Description	Date
• Politique gérée par AWS : AmazonRekognitionCustomLabelsFullAccess	• `ListProjectPolicies` • `DeleteProjectPolicy`	
Les actions ProjectPolicy et la copie du modèle d'étiquettes personnalisées ont été ajoutées aux politiques gérées suivantes : • Politique gérée par AWS : AmazonRekognitionReadOnlyAccess	Amazon Rekognition a ajouté les actions suivantes à la politique gérée par AmazonRekognitionReadOnlyAccess : • `ListProjectPolicies`	21 juillet 2022
Mise à jour de la gestion des jeux de données pour les politiques gérées suivantes : • Politique gérée par AWS : AmazonRekognitionReadOnlyAccess • Politique gérée par AWS : AmazonRekognitionFullAccess • Politique gérée par AWS : AmazonRekognitionCustomLabelsFullAccess	Amazon Rekognition a ajouté les actions suivantes aux, et a géré les AmazonRekognitionReadOnlyAccess politiques AmazonRekognitionFullOnlyAccess AmazonRekognitionCustomLabelsFullAccess • `CreateDataset` • `ListDatasetEntries` • `ListDatasetLabels` • `DescribeDataset` • `UpdateDatasetEntries` • `DistributeDatasetEntries` • `DeleteDataset`	1er novembre 2021

Modifier	Description	Date
Mise à jour du balisage pour Politique gérée par AWS : AmazonRekognitionReadOnlyAccess et Politique gérée par AWS : AmazonRekognitionFullAccess	Amazon Rekognition a ajouté de nouvelles actions de balisage aux politique s et. AmazonRekognitionFullAccess AmazonRekognitionReadOnlyAccess	2 avril 2021
Amazon Rekognition a commencé à assurer le suivi des modifications	Amazon Rekognition a commencé à suivre les modifications apportées à ses politiques gérées. AWS	2 avril 2021

Bonnes pratiques en matière de politiques

Identity-based les politiques déterminent si quelqu'un peut créer, accéder ou supprimer des ressources Amazon Rekognition dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez politiques gérées par AWS ou politiques gérées par AWS pour les activités professionnelles dans le Guide de l'utilisateur IAM.
- Accordez les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez politiques et autorisations dans IAM dans le Guide de l'utilisateur IAM.
- Utilisez des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par

exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez Validation de politiques avec IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Sécurisation de l'accès aux API avec MFA dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.

Utilisation de la console Amazon Rekognition

À l'exception de la fonctionnalité Étiquettes personnalisées Amazon Rekognition, Amazon Rekognition ne nécessite aucune autorisation supplémentaire lors de l'utilisation de la console Amazon Rekognition. Pour plus d'informations sur les Étiquettes personnalisées Amazon Rekognition, consultez étape 5 : Configurer les autorisations de la console des étiquettes personnalisées Amazon Rekognition.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API AWS CLI ou l' AWS API. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API que vous tentez d'effectuer.

Exemples d'utilisation d'Amazon Rekognition Exemples de politiques basées sur l'identité

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou à modifier des ressources Amazon Rekognition. Ils ne peuvent pas non plus effectuer de tâches à l'aide de l' AWS API Console de gestion AWS AWS CLI, ou. Un administrateur IAM doit créer des politiques IAM autorisant les utilisateurs et les rôles à exécuter des opérations d'API spécifiques sur les ressources spécifiées dont ils ont besoin. Il doit ensuite attacher ces stratégies aux utilisateurs ou aux groupes ayant besoin de ces autorisations.

Pour savoir comment créer une politique IAM basée sur l'identité à l'aide de ces exemples de documents de politique JSON, consultez Création de politiques dans l'onglet JSON dans le Guide de l'utilisateur IAM.

Rubriques

- Exemple de politiques d'étiquettes personnalisées Amazon Rekognition
- Exemple 1 : accorder à un utilisateur un accès en lecture seule aux ressources
- Exemple 2 : Accorder à un utilisateur un accès complet aux ressources
- Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Exemple de politiques d'étiquettes personnalisées Amazon Rekognition

Vous pouvez créer des stratégies basées sur l'identité pour les étiquettes personnalisées Amazon Rekognition. Pour plus d'informations, consultez Sécurité.

Exemple 1 : accorder à un utilisateur un accès en lecture seule aux ressources

L'exemple suivant accorde un accès en lecture seule aux ressources Amazon Rekognition.

JSON

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

Exemple 2 : Accorder à un utilisateur un accès complet aux ressources

L’exemple suivant accorde un accès complet aux ressources Amazon Rekognition.

JSON

```
{
```

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

Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or 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": "*"
        }
    ]
}
```

Exemples de politique basée sur les ressources Amazon Rekognition

Les étiquettes personnalisées Amazon Rekognition utilisent des politiques basées sur les ressources, appelées politiques de projet, pour gérer les autorisations de copie pour une version de modèle.

Une politique de projet donne ou refuse l'autorisation de copier une version de modèle d'un projet source vers un projet de destination. Vous avez besoin d'une politique de projet si le projet de destination se trouve dans un autre AWS compte ou si vous souhaitez restreindre l'accès au sein d'un AWS compte. Par exemple, vous pouvez refuser les autorisations de copie pour un rôle IAM spécifique. Pour plus d'informations, consultez Copie d'un modèle.

Octroi de l'autorisation de copier une version de modèle

L'exemple suivant permet au principal `arn:aws:iam::123456789012:role/Admin` de copier la version du modèle `arn:aws:rekognition:us-east-1:123456789012:project/my_project/version/test_1/1627045542080`.

JSON

```
{
  "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"
    }
```

```
  ]
}
```

Résolution de problèmes pour identité et accès Amazon Rekognition

Utilisez les informations suivantes pour identifier et résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec Amazon Rekognition et IAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans Amazon Rekognition
- Je ne suis pas autorisé à effectuer iam : PassRole
- Je suis un administrateur et je veux autoriser d'autres utilisateurs à accéder à Amazon Rekognition
- Je souhaite autoriser des personnes extérieures à mon AWS compte pour accéder à mes ressources Amazon Rekognition

Je ne suis pas autorisé à effectuer une action dans Amazon Rekognition

S'il vous Console de gestion AWS indique que vous n'êtes pas autorisé à effectuer une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni vos informations de connexion.

L'exemple d'erreur suivant se produit lorsque l'utilisateur IAM `mateojackson` tente d'utiliser la console pour afficher des informations détaillées concernant un élément *`widget`* mais ne dispose pas des autorisations `rekognition:`*`GetWidget`* nécessaires.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 rekognition:GetWidget on resource: my-example-widget
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder à la ressource *`my-example-widget`* à l'aide de l'action `rekognition:`*`GetWidget`*.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez une erreur selon laquelle vous n'êtes pas autorisé à exécuter l'action `iam:PassRole`, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à Amazon Rekognition.

Certains vous Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, vous devez disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour effectuer une action dans Amazon Rekognition. Toutefois, l'action nécessite que le service ait des autorisations accordées par une fonction du service. Mary n'est pas autorisée à transmettre le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je suis un administrateur et je veux autoriser d'autres utilisateurs à accéder à Amazon Rekognition

Pour autoriser d'autres personnes à accéder à Amazon Rekognition, vous devez accorder l'autorisation aux personnes ou aux applications qui ont besoin d'y accéder. Si vous utilisez AWS IAM Identity Center pour gérer des personnes et des applications, vous attribuez des ensembles d'autorisations aux utilisateurs ou aux groupes afin de définir leur niveau d'accès. Les ensembles d'autorisations créent et attribuent automatiquement des politiques IAM aux rôles IAM associés à la personne ou à l'application. Pour plus d'informations, consultez la section Ensembles d'autorisations dans le guide de AWS IAM Identity Center l'utilisateur.

Si vous n'utilisez pas IAM Identity Center, vous devez créer des entités IAM (utilisateurs ou rôles) pour les personnes ou les applications qui ont besoin d'un accès. Vous devez ensuite attacher une politique à l'entité qui va lui accorder les autorisations nécessaires dans Amazon Rekognition. Une fois les autorisations accordées, fournissez les informations d'identification à l'utilisateur ou au développeur de l'application. Ils utiliseront ces informations d'identification pour y accéder AWS. Pour en savoir plus sur la création d'utilisateurs, de groupes, de politiques et d'autorisations IAM, consultez la section Identités, politiques et autorisations IAM dans le guide de l'utilisateur IAM.

Je souhaite autoriser des personnes extérieures à mon AWS compte pour accéder à mes ressources Amazon Rekognition

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si Amazon Rekognition est compatible avec ces fonctionnalités, veuillez consulter Fonctionnement d'Amazon Rekognition avec IAM.
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues par des tiers dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.
- Pour en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Protection des données dans Amazon Rekognition

Le modèle de responsabilité AWS partagée (modèle de) s'applique à la protection des données dans Amazon Rekognition. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez la FAQ sur la confidentialité des données et les . Pour plus d'informations sur la protection des données en Europe, consultez le Centre du règlement général sur la protection des données (RGPD).

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou Gestion des identités et des accès AWS (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section Utilisation des CloudTrail sentiers dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-3 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS disponibles, consultez Norme FIPS (Federal Information Processing Standard) 140-3.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec Rekognition ou Services AWS autre à l'aide de la console, de l'API ou des SDK. AWS CLI AWS Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Chiffrement des données

Les informations suivantes expliquent dans quels cas Amazon Rekognition utilise le chiffrement des données pour protéger vos données.

Chiffrement au repos

Image Amazon Rekognition

Images

Les images transmises aux opérations de l'API Amazon Rekognition peuvent être stockées et utilisées pour améliorer le service, sauf si vous vous désinscrivez en vous rendant sur la page de politique de désinscription des services d'intelligence artificielle et en suivant le processus qui y est expliqué. Les images stockées sont chiffrées au repos (Amazon S3) à l'aide d'AWS Key Management Service (SSE-KMS).

Collections

Pour les opérations de comparaison de visages qui stockent des informations dans une collection, l'algorithme de détection sous-jacent détecte d'abord les visages de l'image en entrée, extrait un vecteur pour chaque d'entre eux, puis stocke les vecteurs faciaux dans la collection. Amazon Rekognition utilise ces vecteurs faciaux lors de l'exécution d'une comparaison faciale. Les vecteurs faciaux sont stockés sous la forme d'un ensemble de valeurs flottantes et chiffrées au repos.

Vidéo Amazon Rekognition

Vidéos

Pour analyser une vidéo, Amazon Rekognition copie vos vidéos dans le service pour traitement. Les images peuvent être stockées et utilisées pour améliorer le service, sauf si vous vous désinscrivez en vous rendant sur la page de politique de désinscription des services d'intelligence artificielle et en suivant le processus qui y est expliqué. Les vidéos sont chiffrées au repos (Amazon S3) à l'aide d'AWS Key Management Service (SSE-KMS).

Étiquettes personnalisées Amazon Rekognition

Étiquettes personnalisées Amazon Rekognition chiffre vos données au repos.

Images

Pour entraîner votre modèle, Étiquettes personnalisées Amazon Rekognition crée une copie de vos images d'entraînement et de test source. Les images copiées sont chiffrées au repos dans Amazon Simple Storage Service (S3) à l'aide d'un chiffrement côté serveur avec AWS KMS key une clé KMS que vous fournissez ou AWS que vous possédez. Étiquettes personnalisées Amazon Rekognition prend uniquement en charge les clés KMS symétriques. Vos images sources ne sont

pas affectées. Pour de plus amples informations, veuillez consulter Entraînement d'un modèle étiquettes personnalisées Amazon Rekognition.

Modèles

Par défaut, Étiquettes personnalisées Amazon Rekognition chiffre les modèles entraînés et les fichiers de manifeste stockés dans des compartiments Amazon S3 à l'aide d'un chiffrement côté serveur avec un Clé détenue par AWS. Pour plus d'informations, consultez la section Protection des données à l'aide du Server-Side chiffrement. Les résultats de l'entraînement sont écrits dans le compartiment spécifié dans le paramètre `OutputConfig` d'entrée to CreateProjectVersion. Les résultats de l'entraînement sont chiffrés à l'aide des paramètres de chiffrement configurés pour le compartiment (`OutputConfig`).

Compartiment de la console

La console Étiquettes personnalisées Amazon Rekognition crée un compartiment Amazon S3 (compartiment de console) que vous pouvez utiliser pour gérer vos projets. Le compartiment de console est chiffré à l'aide du chiffrement par défaut d'Amazon S3. Pour plus d'informations, veuillez consulter la rubrique Chiffrement par défaut Amazon S3 pour les compartiments S3. Si vous utilisez votre propre clé KMS, configurez le compartiment de console après sa création. Pour plus d'informations, consultez la section Protection des données à l'aide du Server-Side chiffrement. Les étiquettes personnalisées Amazon Rekognition bloquent l'accès public au compartiment de console.

Rekognition Face Liveness

Toutes les données relatives à la session stockées dans le compte du service Rekognition Face Liveness sont entièrement chiffrées au repos. Par défaut, les images de référence et d'audit sont chiffrées à l'aide d'une clé AWS détenue dans le compte de service. Cependant, vous pouvez choisir de fournir vos propres AWS KMS clés pour chiffrer ces images.

Chiffrement en transit

Les points de terminaison d'API Amazon Rekognition ne prennent en charge que des connexions sécurisées sur HTTPS. Toutes les communications sont chiffrées avec Transport Layer Security (TLS).

Gestion des clés

Vous pouvez utiliser AWS Key Management Service (KMS) pour gérer les clés des images et vidéos en entrée que vous stockez dans des compartiments Amazon S3. Pour de plus amples informations, veuillez consulter Concepts d'AWS Key Management Service.

Chiffrement de clés géré par le client pour Face Liveness

L'CreateFaceLivenessSessionAPI prend en compte un `KmsKeyId` paramètre facultatif. Vous pouvez fournir l'`id` de la clé KMS que vous avez créée dans votre compte. Cette clé sera utilisée pour chiffrer les images de référence et d'audit obtenues pendant l'StartFaceLivenessSessionAPI, et pendant GetFaceLivenessSessionResultsl'API, les images seront déchiffrées à l'aide de cette clé avant de renvoyer les résultats. Si la CreateFaceLivenessSession demande inclut un OutputConfig, les images de référence et d'audit seront téléchargées sur les chemins Amazon S3 spécifiés. Nous vous recommandons d'activer le chiffrement côté serveur (SSE-S3) dans vos compartiments Amazon S3 afin que les données restent chiffrées au repos.

Lorsque vous fournissez votre propre identifiant de AWS KMS clé, le service Rekognition Face Liveness est autorisé à utiliser la clé gérée par le client au nom du principal qui appelle les API. Les principaux (utilisateurs ou rôles) utilisés pour appeler les API depuis le backend du client (API `CreateFaceLivenessSession` et `GetFaceLivenessSessionResults`) doivent avoir accès pour effectuer les opérations suivantes :

- km : DescribeKey
- km : GenerateDataKey
- kms:Decrypt

Confidentialité du trafic inter-réseau

Un point de terminaison d'Amazon Virtual Private Cloud (Amazon VPC) pour Amazon Rekognition est une entité logique au sein d'un VPC qui autorise la connectivité uniquement à Amazon Rekognition. Amazon VPC achemine les demandes vers Amazon Rekognition et les réponses en retour vers le VPC. Pour plus d'informations, consultez Points de terminaison d'un VPC dans le Guide de l'utilisateur Amazon VPC. Pour plus d'informations sur l'utilisation des points de terminaison d'un VPC avec Amazon Rekognition, consultez Utilisation d'Amazon Rekognition avec les points de terminaison d'un VPC Amazon.

Utilisation d'Amazon Rekognition avec les points de terminaison d'un VPC Amazon

Si vous utilisez Amazon Virtual Private Cloud (Amazon VPC) pour héberger vos ressources AWS, vous pouvez établir une connexion entre votre VPC et Amazon Rekognition. Vous pouvez utiliser

cette connexion pour permettre à Amazon Rekognition de communiquer avec vos ressources sur votre VPC sans passer par le réseau Internet public.

Amazon VPC est un service AWS que vous pouvez utiliser pour lancer des ressources AWS dans un réseau virtuel que vous définissez. Avec un VPC, vous contrôlez vos paramètres réseau, tels que la plage d'adresses IP, les sous-réseaux, les tables de routage et les passerelles réseau. Avec les points de terminaison d'un VPC, le réseau AWS gère le routage entre le VPC et les services AWS.

Pour connecter votre VPC à Amazon Rekognition, vous définissez un point de terminaison d'un VPC d'interface pour Amazon Rekognition. Un point de terminaison d'interface est une interface elastic network dotée d'une adresse IP privée qui sert de point d'entrée pour le trafic destiné à un AWS service pris en charge. Le point de terminaison assure une connectivité scalable et fiable à Amazon Rekognition, sans qu'une passerelle Internet, une traduction d'adresses réseau (NAT) ou une connexion VPN ne soit nécessaire. Pour de plus amples informations, veuillez consulter Qu'est-ce qu'Amazon VPC ? dans le Guide de l'utilisateur Amazon VPC.

Les points de terminaison VPC de l'interface sont activés par AWS. PrivateLink Cette technologie AWS permet une communication privée entre les services AWS en utilisant une interface réseau Elastic avec des adresses IP privées.

> **Note**
>
> Tous les points de terminaison Amazon Rekognition Federal Information Processing Standard (FIPS) sont pris en charge par AWS. PrivateLink

Création de points de terminaison d'un VPC pour Amazon Rekognition

Vous pouvez créer deux types de point de terminaison d'un VPC Amazon à utiliser avec Amazon Rekognition.

- Un point de terminaison d'un VPC à utiliser avec les opérations Amazon Rekognition. Pour la plupart des utilisateurs, ce type de point de terminaison d'un VPC est le plus approprié.
- Un point de terminaison d'un VPC pour les opérations Amazon Rekognition avec des points de terminaison conformes à la norme FIPS (Federal Information Processing Standard) 140-2 du gouvernement américain.

Pour commencer à utiliser Amazon Rekognition avec votre VPC, utilisez la console Amazon VPC pour créer un point de terminaison d'un VPC pour Amazon Rekognition. Pour obtenir des instructions, consultez la procédure « Pour créer un point de terminaison d'interface vers un service AWS à l'aide de la console » dans Création d'un point de terminaison d'interface. Notez les étapes de la procédure suivante :

- Étape 3 : pour Catégorie de service, choisissez AWS services.
- Étape 4 : pour Nom du service, choisissez l'une des options suivantes :
  - com.amazonaws.region.rekognition : crée un point de terminaison d'un VPC pour les opérations Amazon Rekognition.
  - com.amazonaws.region.rekognition-fips : crée un point de terminaison d'un VPC pour les opérations Amazon Rekognition avec des points de terminaison conformes à la norme FIPS (Federal Information Processing Standard) 140-2 du gouvernement américain.

Pour plus d'informations, consultez Mise en route dans le Guide de l'utilisateur Amazon VPC.

Créez une politique de point de terminaison d'un VPC pour Amazon Rekognition

Vous pouvez créer une politique pour les points de terminaison d'un VPC Amazon pour Amazon Rekognition pour spécifier les points suivants :

- Le principal qui peut exécuter des actions.
- Les actions qui peuvent être effectuées.
- Les ressources sur lesquelles les actions peuvent être exécutées.

Pour plus d'informations, veuillez consulter Contrôle de l'accès aux services avec des points de terminaison d'un VPC dans le Guide de l'utilisateur Amazon VPC.

L'exemple de politique suivant permet aux utilisateurs se connectant à Amazon Rekognition via le point de terminaison d'un VPC d'appeler l'opération d'API `DetectFaces`. La politique empêche les utilisateurs d'effectuer d'autres opérations d'API Amazon Rekognition via le point de terminaison d'un VPC.

Les utilisateurs peuvent toujours appeler d'autres opérations d'API Amazon Rekognition à partir de l'extérieur du VPC. Pour de plus amples informations sur la façon de refuser l'accès aux opérations

d'API Amazon Rekognition qui se trouvent en dehors du VPC, consultez Politiques basées sur l'identité Amazon Rekognition.

JSON

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

Pour modifier la politique de point de terminaison d'un VPC pour Amazon Rekognition

1. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
2. Si vous n'avez pas encore créé le point de terminaison pour Amazon Rekognition, choisissez Créer un point de terminaison. Sélectionnez ensuite com.amazonaws. *Region*.rekognition et choisissez Create endpoint.
3. Dans le panneau de navigation, choisissez Points de terminaison.
4. Sélectionnez com.amazonaws. ***Region***.rekognition endpoint et choisissez l'onglet Policy dans la moitié inférieure de l'écran.
5. Choisissez Modifier la politique, puis apportez les modifications souhaitées à la politique.

Validation de la conformité pour Amazon Rekognition

Third-party les auditeurs évaluent la sécurité et la conformité d'Amazon Rekognition dans le cadre de plusieurs programmes de conformité. AWS Il s'agit notamment des certifications SOC, PCI, FedRAMP, HIPAA et d'autres.

Pour obtenir la liste des AWS services concernés par des programmes de conformité spécifiques, consultez la section Services AWS concernés par programme de conformité. Pour obtenir des informations générales, consultez Programmes de conformitéAWS.

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, consultez Téléchargement des rapports dans AWS Artifact.

Lorsque vous utilisez Amazon Rekognition, votre responsabilité en matière de conformité dépend de la sensibilité de vos données, des objectifs de conformité de votre entreprise et des lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- Guides de démarrage rapide sur la sécurité et la conformité : ces guides de déploiement abordent les considérations architecturales et indiquent les étapes à suivre pour déployer des environnements de base axés sur la sécurité et la conformité sur. AWS
- Livre blanc sur l'architecture pour la sécurité et la conformité HIPAA — Ce livre blanc décrit comment les entreprises peuvent créer des applications. AWS HIPAA-compliant
- AWS Ressources relatives à la conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- AWS Config— Ce AWS service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- AWS Security Hub CSPM— Ce AWS service fournit une vue complète de l'état de votre sécurité interne, AWS ce qui vous permet de vérifier votre conformité aux normes et aux meilleures pratiques du secteur de la sécurité.

Résilience dans Amazon Rekognition

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section Infrastructure AWS mondiale.

Outre l'infrastructure AWS mondiale, Amazon Rekognition propose plusieurs fonctionnalités pour répondre à vos besoins en matière de résilience et de sauvegarde des données.

Configuration et analyse des vulnérabilités dans Amazon Rekognition

La configuration et les contrôles informatiques sont une responsabilité partagée entre AWS et vous, notre client. Pour plus d'informations, consultez le modèle de responsabilité AWS partagée.

Cross-service prévention confuse des adjoints

Dans AWS, l'usurpation d'identité entre services peut se produire lorsqu'un service (le service appelant) appelle un autre service (le service appelé). Le service appelant peut être manipulé pour agir sur les ressources d'un autre client, même s'il ne devrait pas avoir les autorisations appropriées, ce qui se traduit par un problème d'adjoint confus.

Pour éviter cela, AWS fournit des outils qui vous aident à protéger vos données pour tous les services auprès des principaux fournisseurs de services qui ont obtenu l'accès aux ressources de votre compte.

Nous vous recommandons d'utiliser les clés de contexte de condition globale `aws:SourceArn` ou `aws:SourceAccount` dans les politiques de ressources afin de limiter les autorisations à la ressource octroyées par Amazon Rekognition à un autre service.

Si la valeur `aws:SourceArn` ne contient pas l'ID du compte, tel qu'un ARN de compartiment Amazon S3, vous devez utiliser les deux clés pour limiter les autorisations. Si vous utilisez les deux clés et que la valeur de `aws:SourceArn` contient l'ID de compte, la valeur de `aws:SourceAccount` et le compte indiqué dans la valeur de `aws:SourceArn` doivent utiliser le même ID de compte lorsqu'il est utilisé dans la même déclaration de politique.

Utilisez `aws:SourceArn` si vous souhaitez qu'une seule ressource soit associée à l'accès entre services. Utilisez `aws:SourceAccount` si vous souhaitez autoriser l'association d'une ressource de ce compte à l'utilisation interservices.

La valeur de `aws:SourceArn` doit être l'ARN de la ressource utilisée par Rekognition, qui est spécifié au format suivant : `arn:aws:rekognition:region:account:resource`.

La valeur de `arn:User` ARN doit être l'ARN de l'utilisateur qui appellera l'opération d'analyse vidéo (l'utilisateur qui assume un rôle).

Le moyen le plus efficace de se protéger contre le problème de l'adjoint confus consiste à utiliser la clé de contexte de condition globale `aws:SourceArn` avec l'ARN complet de la ressource.

Si vous ne connaissez pas l'ARN complet de la ressource, ou si vous spécifiez plusieurs ressources, utilisez la clé de contexte de condition globale `aws:SourceArn` avec des caractères génériques (`*`) pour les parties inconnues de l'ARN. Par exemple, `arn:aws:rekognition:*:111122223333:*`.

Afin de vous protéger contre le problème d'adjoint désorienté, effectuez les tâches suivantes :

1. Dans le panneau de navigation de la console IAM, sélectionnez l'option Roles. La console affiche les rôles de votre compte actuel.
2. Choisissez le nom du rôle que vous souhaitez modifier. Le rôle que vous modifiez doit être soumis à la politique AmazonRekognitionServiceRoled'autorisation. Sélectionnez l'onglet Trust Relationships (Relations d'approbation).
3. Choisissez Edit trust policy (Modifier la politique d'approbation).
4. Sur la page Modifier la politique d'approbation, remplacez la politique JSON par défaut par une politique qui utilise une ou les deux clés de contexte de condition globale `aws:SourceArn` ou `aws:SourceAccount`. Voir les exemples de politiques suivant :
5. Choisissez Mettre à jour une politique.

L'exemple suivant montre comment utiliser les clés de contexte de condition globale `aws:SourceArn` et `aws:SourceAccount` dans Amazon Rekognition afin d'éviter le problème de l'adjoint confus.

Si vous travaillez sur des vidéos stockées et diffusées en streaming, vous pouvez utiliser une politique telle que la suivante dans votre rôle IAM :

Si vous travaillez exclusivement avec de la vidéo stockée, vous pouvez utiliser une politique telle que la suivante dans votre rôle IAM (notez que vous n'êtes pas obligé d'inclure l'argument `StringLike` qui spécifie le `streamprocessor`) :

Sécurité de l'infrastructure dans Amazon Rekognition

En tant que service géré, Amazon Rekognition est AWS protégé par la sécurité du réseau mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section Sécurité du AWS cloud. Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section Protection de l'infrastructure dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à Amazon Rekognition via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Des suites de chiffrement dotées d'un secret de transmission parfait (PFS), telles que DHE (Ephemeral) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

Surveillance d’Amazon Rekognition

La surveillance constitue une part importante de la gestion de la fiabilité, de la disponibilité et des performances d’Amazon Rekognition et de vos autres solutions AWS. AWS fournit les outils de surveillance suivants pour contrôler Rekognition, signaler les problèmes et déclencher des actions automatiques, si nécessaire :

- Amazon CloudWatch surveille vos AWS ressources et les applications que vous utilisez AWS en temps réel. Vous pouvez collecter et suivre les métriques, créer des tableaux de bord personnalisés, et définir des alarmes qui vous informent ou prennent des mesures lorsqu’une métrique spécifique atteint un seuil que vous spécifiez. Par exemple, vous pouvez CloudWatch suivre l'utilisation du processeur ou d'autres indicateurs de vos instances Amazon EC2 et lancer automatiquement de nouvelles instances en cas de besoin. Pour plus d'informations, consultez le guide de CloudWatch l'utilisateur Amazon.
- Amazon CloudWatch Logs vous permet de surveiller, de stocker et d'accéder à vos fichiers journaux à partir d'instances Amazon EC2 et d'autres sources. CloudTrail CloudWatch Les journaux peuvent surveiller les informations contenues dans les fichiers journaux et vous avertir lorsque certains seuils sont atteints. Vous pouvez également archiver vos données de journaux dans une solution de stockage hautement durable. Pour plus d'informations, consultez le guide de l'utilisateur Amazon CloudWatch Logs.
- Amazon EventBridge peut être utilisé pour automatiser vos AWS services et répondre automatiquement aux événements du système, tels que les problèmes de disponibilité des applications ou les modifications des ressources. Les événements AWS liés aux services sont diffusés EventBridge en temps quasi réel. Vous pouvez écrire des règles simples pour préciser les événements qui vous intéressent et les actions automatisées à effectuer quand un événement correspond à une règle. Pour plus d'informations, consultez le guide de EventBridge l'utilisateur Amazon.
- AWS CloudTrailcapture les appels d'API et les événements associés effectués par ou pour le compte de votre AWS compte et envoie les fichiers journaux dans un compartiment Amazon S3 que vous spécifiez. Vous pouvez identifier les utilisateurs et les comptes appelés AWS, l'adresse IP source à partir de laquelle les appels ont été effectués et la date des appels. Pour plus d’informations, consultez le Guide de l’utilisateur AWS CloudTrail.

Surveillance de la Rekognition avec Amazon CloudWatch

Vous pouvez obtenir des statistiques pour des opérations de Rekognition individuelles ou des mesures de Rekognition globales pour votre compte. Vous pouvez utiliser des métriques pour suivre l'état Rekognition-based de votre solution et configurer des alarmes pour vous avertir lorsqu'une ou plusieurs mesures dépassent un seuil défini. CloudWatch Par exemple, vous pouvez consulter les métriques pour connaître le nombre d'erreurs serveur qui se sont produites ou pour connaître le nombre de visages qui ont été détectés. Vous pouvez également consulter les métriques pour obtenir le nombre de fois où une opération Rekognition spécifique a réussi. Pour consulter les statistiques, vous pouvez utiliser Amazon CloudWatch AWS Command Line Interface, Amazon ou l'CloudWatch API.

Vous pouvez également consulter les métriques agrégées, pour une période de temps choisie, à l'aide de la console Rekognition. Pour de plus amples informations, veuillez consulter Exercice 4 :afficher les métriques agrégées (console).

Utilisation de CloudWatch métriques pour la Rekognition

Pour utiliser les métriques, vous devez spécifier les informations suivantes :

- La dimension de la métrique ou l'absence de dimension. Une dimension est une paire nom-valeur qui vous aide à identifier une métrique de façon unique. Rekognition possède une dimension nommée Opération. Elle fournit les métriques d'une opération spécifique. Si vous ne spécifiez pas de dimension, la portée de la métrique englobe toutes les opérations Rekognition au sein de votre compte.
- Le nom de la métrique, par exemple `UserErrorCount`.

Vous pouvez obtenir des données de surveillance pour Rekognition à l'aide de l'API, AWS CLI de ou de Console de gestion AWS l'API. CloudWatch Vous pouvez également utiliser l' CloudWatch API via l'un des kits de développement logiciel (SDK) Amazon AWS ou les outils d' CloudWatch API. La console affiche une série de graphiques basés sur les données brutes de l' CloudWatch API. En fonction de vos besoins, vous pouvez utiliser les graphiques affichés dans la console ou extraits de l'API.

La liste suivante présente certaines utilisations courantes des métriques. Voici quelques suggestions pour vous aider à démarrer, qui ne forment pas une liste exhaustive.

Comment... ?	Métriques pertinentes
Comment suivre le nombre de visages reconnus ?	Surveillez la statistique `Sum` de la métrique `DetectedFaceCount` .
Comment savoir si mon application a atteint le nombre maximal de demandes par seconde ?	Surveillez la statistique `Sum` de la métrique `ThrottledCount` .
Comment surveiller les erreurs de demande ?	Utilisez la statistique `Sum` de la métrique `UserErrorCount` .
Comment obtenir le nombre total de demandes ?	Utilisez les statistiques `ResponseTime` et `Data Samples` de la métrique `ResponseTime` . Celle-ci inclut les demandes qui se sont traduites par une erreur. Si vous voulez ne voir que les appels d'opération ayant réussi, utilisez la métrique `SuccessfulRequestCount` .
Comment surveiller la latences des appels de l'opération `Rekognition` ?	Utilisez la métrique `ResponseTime` .
Comment surveiller le nombre de fois où `IndexFaces` a ajouté avec succès des visages aux collections Rekognition ?	Surveillez la statistique `Sum` à l'aide de la métrique `SuccessfulRequestCount` et de l'opération `IndexFaces` . Utilisez la dimension `Operation` pour sélectionner l'opération et la métrique.

Vous devez disposer des CloudWatch autorisations appropriées pour surveiller Rekognition avec. CloudWatch Pour de plus amples informations, veuillez consulter Authentification et contrôle d'accès pour Amazon CloudWatch.

Métriques Access Rekognition

Les exemples suivants montrent comment accéder aux métriques de Rekognition à l'aide de la console, AWS CLI de l'API et CloudWatch de l'API. CloudWatch

Pour consulter les métriques (console)

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Choisissez Métriques, puis l'onglet Toutes les métriques, puis Rekognition.
3. Choisissez Métriques sans dimensions, puis choisissez une métrique.

   Par exemple, choisissez la métrique DetectedFace pour mesurer le nombre de visages détectés.
4. Choisissez une valeur pour la plage de dates. Nombre de métriques affichées dans le graphique.

Pour afficher les appels réussis de l'opération **`DetectFaces`** effectués sur une période de temps (CLI).

- Ouvrez le AWS CLI et entrez la commande suivante :

  ```
  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
  ```

  Cet exemple illustre les appels réussis de l'opération `DetectFaces` effectués sur une période de temps. Pour de plus amples informations, veuillez consulter get-metric-statistics.

Pour accéder aux métriques (CloudWatch API)

- Appelez `GetMetricStatistics`. Pour plus d'informations, consultez le Amazon CloudWatch API Reference.

Créer une alarme

Vous pouvez créer une CloudWatch alarme qui envoie un message Amazon Simple Notification Service (Amazon SNS) lorsque l'état de l'alarme change. Une alarme surveille une seule métrique pendant une durée que vous définissez et exécute une ou plusieurs actions en fonction de la valeur de la métrique par rapport à un seuil donné pendant un certain nombre de périodes. L'action est une notification envoyée à une rubrique Amazon SNS ou à une stratégie Auto Scaling.

Les alarmes déclenchent des actions uniquement pour les changements d'état prolongés. CloudWatch les alarmes n'appellent pas d'actions simplement parce qu'elles sont dans un état particulier. L'état doit avoir changé et avoir été maintenu pendant un nombre de périodes spécifié.

Pour définir une alarme (console)

1. Connectez-vous à la CloudWatch console Console de gestion AWS et ouvrez-la à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Sélectionnez Create Alarm (Créer une alerte). L'assistant Create Alarm démarre.
3. Dans la liste Métriques sans dimensions, choisissez Métriques Rekognition, puis choisissez une métrique.

   Par exemple, choisissez DetectedFaceCount pour définir une alarme quand le nombre maximal de visages détectés est atteint.
4. Dans la zone Time Range, sélectionnez une plage de dates incluant les opérations de détection de visage que vous avez appelées. Choisissez Next (Suivant)
5. Remplissez les champs Nom et Description. Pour Lorsque, choisissez **>=** et entrez une valeur maximale de votre choix.
6. Si vous souhaitez vous CloudWatch envoyer un e-mail lorsque l'état d'alarme est atteint, pour Whenever this alarm :, choisissez State is ALARM. Afin d'envoyer des alarmes à une rubrique Amazon SNS existante, pour Envoyer une notification à : choisissez une rubrique SNS existante. Pour définir le nom et les adresses e-mail d'une nouvelle liste d'abonnement par e-mail, choisissez Créer un sujet CloudWatch pour enregistrer la liste et l'afficher dans le champ afin que vous puissiez l'utiliser pour définir de futures alarmes.

   > **Note**
   >
   > Si vous utilisez Créer une rubrique pour créer une rubrique Amazon SNS, les adresses de messagerie doivent être vérifiées avant que les destinataires prévus ne reçoivent les notifications. Amazon SNS n'envoie les e-mails que lorsque l'alarme passe à l'état d'alarme. Si ce changement d'état de l'alarme se produit avant la vérification des adresses de messagerie, les destinataires prévus ne reçoivent pas de notification.
7. Affichez un aperçu de l'alarme dans la section Aperçu de l'alarme. Sélectionnez Create Alarm (Créer une alerte).

Pour régler une alarme (AWS CLI)

- Ouvrez le AWS CLI et entrez la commande suivante. Modifiez la valeur du paramètre `alarm-actions` pour faire référence à une rubrique Amazon SNS que vous avez précédemment créée.

  ```
  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
  ```

  Cet exemple montre comment créer une alarme lorsque plus de 10 erreurs d'utilisateur se produisent en 5 minutes ou moins. Pour de plus amples informations, veuillez consulter put-metric-alarm.

Pour configurer une alarme (CloudWatch API)

- Appelez `PutMetricAlarm`. Pour plus d'informations, consultez Amazon CloudWatch API Reference.

CloudWatch métriques pour Rekognition

Cette section contient des informations sur les CloudWatch métriques Amazon et la dimension d'opération disponibles pour Amazon Rekognition.

Vous pouvez aussi afficher une vue de groupe des métriques Rekognition à partir de la console Rekognition. Pour de plus amples informations, veuillez consulter Exercice 4 :afficher les métriques agrégées (console).

CloudWatch métriques pour Rekognition

Le tableau suivant récapitule les métriques Rekognition.

Métrique	Description
SuccessfulRequestCount	Nombre de requêtes réussies. La plage de codes de réponse d'une demande réussie est comprise entre 200 et 299.

Métrique	Description
	Unité : nombre Statistiques valides : `Sum,Average`
ThrottledCount	Nombre de demandes limitées. Rekognition limite une demande quand il en reçoit plus que la limite de transactions par seconde définie pour votre compte. Si cette limite est souvent franchie, vous pouvez demander une augmentation de la limite. Pour demander une augmentation, consultez Limites de service AWS. Unité : nombre Statistiques valides : `Sum,Average`
ResponseTime	Durée en millisecondes pour que Rekognition calcule la réponse. Unités : 1. Nombre de statistiques `Data Samples` 2. Millisecondes pour les statistiques `Average` Statistiques valides : `Data Samples,Average` **Note** La métrique `ResponseTime` n'est pas incluse dans le volet des métriques Rekognition.
DetectedFaceCount	Nombre de visages détectés avec l'opération `IndexFaces` ou `DetectFaces` . Unité : nombre Statistiques valides : `Sum,Average`

Métrique	Description
DetectedLabelCount	Nombre d'étiquettes détectées avec l'opération `DetectLabels` . Unité : nombre Statistiques valides : `Sum,Average`
ServerErrorCount	Nombre d'erreurs de serveur. La plage des codes de réponse d'une erreur de serveur est comprise entre 500 et 599. Unité : nombre Statistiques valides : `Sum, Average`
UserErrorCount	Nombre d'erreurs d'utilisateur (paramètres non valides, image non valide, absence d'autorisation, etc). La plage des codes de réponse d'une erreur d'utilisateur est comprise entre 400 et 499. Unité : nombre Statistiques valides : `Sum,Average`
MinInferenceUnits	Le nombre minimum d'unités d'inférence spécifié lors de la demande `StartProjectVersion` . Unité : nombre Statistiques valides : `Average`
MaxInferenceUnits	Le nombre maximum d'unités d'inférence spécifiées lors de la demande `StartProjectVersion` . Unité : nombre Statistiques valides : `Average`

Métrique	Description
DesiredInferenceUnits	Le nombre d'unités d'inférence auxquelles Rekognition augmente ou diminue. Unité : nombre Statistiques valides : `Average`
InServiceInferenceUnits	Nombre d'unités d'inférence utilisées par le modèle. Unité : nombre Statistiques valides : `Average` Il est recommandé d'utiliser la statistique Average pour obtenir la moyenne sur 1 minute du nombre d'instances utilisées.

CloudWatch métriques pour Rekognition Streaming

Rekognition possède également un deuxième espace de noms utilisé pour les opérations de streaming, « Rekognition Streaming ». Le tableau suivant récapitule les métriques de Rekognition Streaming.

Métrique	Description
SuccessfulRequestCount	Nombre de requêtes réussies. La plage de codes de réponse d'une demande réussie est comprise entre 200 et 299. Unité : nombre Statistiques valides : `Sum,Average`
CallCount	Nombre d'opérations spécifiées effectuées dans votre compte. Statistiques valides : `Sum,Average`
ThrottledCount	Nombre de demandes limitées. Rekognition limite une demande quand il en reçoit plus que la limite de transactions par seconde définie pour votre compte. Si cette limite est souvent franchie, vous pouvez demander une

Métrique	Description
	augmentation de la limite. Pour demander une augmentation, consultez Limites de service AWS. Unité : nombre Statistiques valides : `Sum,Average`
ServerErrorCount	Nombre d'erreurs de serveur. La plage des codes de réponse d'une erreur de serveur est comprise entre 500 et 599. Unité : nombre Statistiques valides : `Sum, Average`
UserErrorCount	Nombre d'erreurs d'utilisateur (paramètres non valides, image non valide, absence d'autorisation, etc). La plage des codes de réponse d'une erreur d'utilisateur est comprise entre 400 et 499. Unité : nombre Statistiques valides : `Sum,Average`
ConcurrentSessions	Le nombre de sessions simultanées que votre compte utilise actuellem ent. Unité : nombre Statistiques valides : `Max,Average`

CloudWatch dimension pour Rekognition

Pour extraire les métriques spécifiques à une opération, utilisez l'espace de noms `Rekognition` et fournissez une dimension d'opération.

Pour plus d'informations sur les dimensions, consultez la section Dimensions du guide de CloudWatch l'utilisateur Amazon.

CloudWatch dimension pour les étiquettes personnalisées Rekognition

Le tableau suivant indique les CloudWatch dimensions disponibles pour une utilisation avec les étiquettes personnalisées Rekognition :

Dimension	Description
ProjectName	Le nom du projet Rekognition Custom Labels que vous avez créé avec `CreateProject` .
VersionName	Nom de la version du projet Rekognition Custom Labels que vous avez créée avec `CreateProjectVersion` .

Pour plus d'informations sur les dimensions, consultez la section Dimensions du guide de CloudWatch l'utilisateur Amazon.

Journalisation des appels d'API Amazon Rekognition avec AWS CloudTrail

Amazon Rekognition est AWS CloudTrail intégré à un service qui fournit un enregistrement des actions effectuées par un utilisateur, un rôle ou un AWS service dans Amazon Rekognition. CloudTrail capture tous les appels d'API pour Amazon Rekognition sous forme d'événements. Ces captures incluent les appels de la console Amazon Rekognition et les appels de code vers les opérations d'API Amazon Rekognition. Si vous créez un suivi, vous pouvez activer la diffusion continue d' CloudTrail événements vers un compartiment Amazon S3, y compris des événements pour Amazon Rekognition. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents dans la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande envoyée à Amazon Rekognition, l'adresse IP à partir de laquelle la demande a été effectuée, l'auteur de la demande, la date à laquelle elle a été faite, ainsi que des informations supplémentaires.

Pour en savoir plus CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur.

Informations sur Amazon Rekognition dans CloudTrail

CloudTrail est activé sur votre AWS compte lorsque vous le créez. Lorsqu'une activité se produit dans Amazon Rekognition, cette activité est enregistrée CloudTrail dans un événement AWS avec d'autres événements de service dans l'historique des événements. Vous pouvez consulter, rechercher et télécharger les événements récents dans votre AWS compte. Pour plus d'informations, consultez la section Affichage des événements à l'aide de l'historique des CloudTrail événements.

Pour un enregistrement continu des événements de votre AWS compte, y compris des événements liés à Amazon Rekognition, créez un historique. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un journal d'activité dans la console, il s'applique à toutes les régions AWS. Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez configurer d'autres AWS services pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence. Pour plus d'informations, consultez les ressources suivantes :

- Vue d'ensemble de la création d'un journal d'activité
- CloudTrail Services et intégrations pris en charge
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers CloudTrail journaux de plusieurs régions et réception de fichiers CloudTrail journaux de plusieurs comptes

Toutes les actions Amazon Rekognition sont CloudTrail enregistrées et documentées dans la référence de l'API Amazon Rekognition. Par exemple, les appels au`CreateCollection`, `CreateStreamProcessor` et les `DetectCustomLabels` actions génèrent des entrées dans les fichiers CloudTrail journaux.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été faite avec les informations d'identification de l'utilisateur root ou Gestion des identités et des accès AWS (IAM).
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la demande a été faite par un autre AWS service.

Pour plus d'informations, consultez la section Élément userIdentity CloudTrail .

Présentation des entrées du fichier journal d'Amazon Rekognition

Un suivi est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre une entrée de CloudTrail journal contenant des actions pour l'API suivante : `StartLabelDetection` et`DetectLabels`.

```
{
    "Records": [

        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "AssumedRole",
                "principalId": "AIDAJ45Q7YFFAREXAMPLE",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/JorgeSouza",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "sessionContext": {
                    "sessionIssuer": {
                        "type": "Role",
                        "principalId": "AIDAJ45Q7YFFAREXAMPLE",
                        "arn": "arn:aws:iam::111122223333:role/Admin",
                        "accountId": "111122223333",
                        "userName": "Admin"
                    },
                    "webIdFederationData": {},
                    "attributes": {
                        "mfaAuthenticated": "false",
                        "creationDate": "2020-06-30T20:10:09Z"
                    }
                }
            },
            "eventTime": "2020-06-30T20:42:14Z",
```

```
            "eventSource": "rekognition.amazonaws.com",
            "eventName": "StartLabelDetection",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "192.0.2.0",
            "userAgent": "aws-cli/3",
            "requestParameters": {
                "video": {
                    "s3Object": {
                        "bucket": "my-bucket",
                        "name": "my-video.mp4"
                    }
                }
            },
            "responseElements": {
                "jobId":
 "653de5a7ee03bd5083edde98ea8fce5794fcea66d077bdd4cfb39d71aff8fc25"
            },
            "requestID": "dfcef8fc-479c-4c25-bef0-d83a7f9a7240",
            "eventID": "b602e460-c134-4ecb-ae78-6d383720f29d",
            "readOnly": false,
            "eventType": "AwsApiCall",
            "recipientAccountId": "111122223333"
        },
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "AssumedRole",
                "principalId": "AIDAJ45Q7YFFAREXAMPLE",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/JorgeSouza",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "sessionContext": {
                    "sessionIssuer": {
                        "type": "Role",
                        "principalId": "AIDAJ45Q7YFFAREXAMPLE",
                        "arn": "arn:aws:iam::111122223333:role/Admin",
                        "accountId": "111122223333",
                        "userName": "Admin"
                    },
                    "webIdFederationData": {},
                    "attributes": {
                        "mfaAuthenticated": "false",
                        "creationDate": "2020-06-30T21:19:18Z"
                    }
```

```
                }
            },
            "eventTime": "2020-06-30T21:21:47Z",
            "eventSource": "rekognition.amazonaws.com",
            "eventName": "DetectLabels",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "192.0.2.0",
            "userAgent": "aws-cli/3",
            "requestParameters": {
                "image": {
                    "s3Object": {
                        "bucket": "my-bucket",
                        "name": "my-image.jpg"
                    }
                }
            },
            "responseElements": null,
            "requestID": "5a683fb2-aec0-4af4-a7df-219018be2155",
            "eventID": "b356b0fd-ea01-436f-a9df-e1186b275bfa",
            "readOnly": true,
            "eventType": "AwsApiCall",
            "recipientAccountId": "111122223333"
        }
    ]
}
```

Directives et quotas dans Amazon Rekognition

Les sections suivantes fournissent les consignes et les quotas qui s'appliquent à l'utilisation d'Amazon Rekognition. Il existe deux types de quotas. Les quotas définis tels que la taille maximale de l'image ne peuvent pas être modifiés. Les quotas par défaut répertoriés sur la page Service Quotas AWS peuvent être modifiés en suivant la procédure décrite dans la section Quotas par défaut.

Rubriques

- Régions prises en charge
- Définition des quotas.
- Quotas par défaut

Régions prises en charge

Pour obtenir la liste des AWS régions dans lesquelles Amazon Rekognition est disponible, consultez la section Régions et points de terminaison AWS dans le manuel Amazon Web Services General Reference.

Définition des quotas.

Voici une liste de limitations pour Amazon Rekognition qui ne peuvent pas être modifiées. Pour plus d'informations sur les limites pouvant être modifiées comme les transactions par seconde (TPS), consultez Quotas par défaut.

Pour connaître les limites des étiquettes personnalisées Amazon Rekognition, consultez les directives et quotas dans les étiquettes personnalisées Amazon Rekognition.

Image Amazon Rekognition

- La taille maximale de l'image stockée sous la forme d'un objet Amazon S3 est limitée à 15 Mo.
- La taille maximale de l'image `DetectModerationLabels` est de 10 000 pixels en largeur et en hauteur.
- La taille maximale de l'image `DetectLabels` est de 10 000 pixels en largeur et en hauteur.

- Pour qu'un visage soit détecté, sa taille doit être supérieure à 40 x 40 pixels dans une image de 1920 x 1080 pixels. Des images avec des dimensions supérieures à 1920 x 1080 pixels auront besoin d'une taille de visage minimum plus grande proportionnellement.
- La taille d'image minimale est de 80 pixels pour la hauteur et la largeur. La taille d'image minimale de `DetectProtectiveEquipment` est de 64 pixels pour la hauteur et la largeur.
- La taille maximale de l'image de `DetectProtectiveEquipment` est de 4 096 pixels en largeur et en hauteur.
- Pour qu'elle soit détectée par `DetectProtectiveEquipment`, une personne ne doit pas être inférieure à 100x100 pixels dans une image de 800x1 300. Des images avec des dimensions supérieures à 800x1 300 pixels auront besoin d'une taille de personne minimum plus grande proportionnellement.
- La taille maximale des images en tant qu'octets bruts transmis comme paramètre à une API est de 5 Mo. La limite est de 4 Mo pour l'API `DetectProtectiveEquipment`.
- Amazon Rekognition prend en charge les formats d'image PNG et JPEG. Ainsi, les images que vous fournissez comme entrées des différentes opérations d'API, telles que `DetectLabels` et `IndexFaces`, doivent être dans l'un des formats pris en charge.
- Le nombre maximal de vecteurs de visages que vous pouvez stocker dans une même collection de visages est de 20 millions.
- Le nombre maximal de vecteurs d'utilisateurs que vous pouvez stocker dans une même collection de visages est de 10 millions.
- Le nombre maximal de correspondances de vecteurs de visages que l'API de recherche renvoie est 4 096.
- Le nombre maximal de correspondances de vecteurs d'utilisateurs que l'API de recherche renvoie est 4 096.
- `DetectText` peut détecter jusqu'à 100 mots dans une image.
- `DetectProtectiveEquipment` peut détecter les équipements de protection individuelle sur un maximum de 15 personnes.

Pour plus d'informations sur les bonnes pratiques en matière d'images et de comparaison faciale, consultez Bonnes pratiques pour les capteurs, images d'entrée et vidéos.

Analyse globale des images Amazon Rekognition

- Amazon Rekognition Image Bulk Analysis peut analyser des lots d'images d'une taille maximale de 10 000 images.
- Amazon Rekognition Image Bulk Analysis prend en charge les manifestes d'entrée d'une taille maximale de 50 Mo.

Vidéo stockée par Vidéo Amazon Rekognition

- Vidéo Amazon Rekognition peut analyser des vidéos stockées d’une taille maximale de 10 Go.
- Vidéo Amazon Rekognition peut analyser des vidéos stockées d’une longueur de 6 heures maximum.
- Vidéo Amazon Rekognition prend en charge au maximum 20 tâches simultanées par compte.
- Les vidéos enregistrées doivent être encodées à l'aide du H.264 codec. Les formats de fichier pris en charge sont MPEG-4 et MOV.
- Toute API Vidéo Amazon Rekognition qui analyse les données audio ne prend en charge que les codecs audio AAC.
- La durée de vie (TTL) pour les jetons de pagination est de 24 heures. Les jetons de pagination sont dans le champ `NextToken` renvoyé par les opérations Get telles que `GetLabeldetection`.

Vidéo en streaming Vidéo Amazon Rekognition

- Un flux d’entrée vidéo Kinesis ne peut être associé qu’à un seul processeur de flux Vidéo Amazon Rekognition.
- Un flux de sortie de données Kinesis ne peut être associé qu’à un seul processeur de flux Vidéo Amazon Rekognition.
- Le flux d’entrée vidéo Kinesis et le flux de sortie de données Kinesis associés à un processeur de flux Vidéo Amazon Rekognition ne peuvent pas être partagés par plusieurs processeurs.
- Toute API Vidéo Amazon Rekognition qui analyse les données audio ne prend en charge que les codecs audio ACC.

Quotas par défaut

La liste des quotas par défaut est disponible sur Service Quotas AWS. Ces limites par défaut peuvent être modifiées. Pour demander une augmentation de votre limite, créez une demande. Pour connaître vos limites de quotas actuelles (valeurs de quota appliquées), consultez Amazon Rekognition Service Quotas. Pour consulter l'historique de votre utilisation du TPS pour les API Image Amazon Rekognition, consultez la page Amazon Rekognition Service Quotas et choisissez une opération d'API spécifique pour voir l'historique de cette opération.

Rubriques

- Calculer la modification de quota TPS
- Bonnes pratiques relatives aux quotas TPS
- Créez un dossier pour modifier les quotas TPS

Calculer la modification de quota TPS

Quelle est la nouvelle limite que vous demandez ? Les transactions par seconde (TPS) sont particulièrement pertinentes au pic de charge de travail attendu. Il est important de comprendre le nombre maximal d'appels d'API simultanés au pic d'une charge de travail et le temps de réponse (5 à 15 secondes). Veuillez noter que 5 secondes devraient être le minimum. En voici deux exemples ci-dessous :

- Exemple 1 : Le nombre maximum d'utilisateurs simultanés de l'authentification faciale (CompareFaces API) que j'attends au début de mon heure de pointe est de 1 000. Ces réponses sont réparties sur une période de 10 secondes. Par conséquent, le TPS requis est de 100 (1000/10) pour l' CompareFaces API dans ma région concernée.
- Exemple 2 : Le nombre maximum d'appels de détection d'objets (DetectLabels API) simultanés attendus au début de mon heure de pointe est de 250. Ces réponses sont réparties sur une période de 5 secondes. Par conséquent, le TPS requis est de 50 (250/5) pour l' DetectLabels API dans ma région concernée.

Bonnes pratiques relatives aux quotas TPS

Les bonnes pratiques recommandées pour les transactions par seconde (TPS) incluent la réduction du trafic en pointe, la configuration des nouvelles tentatives et la configuration du backoff exponentiel et de l'instabilité de l'ombre.

1. Circulation fluide et pointue. Les pics de trafic affectent le débit. Pour obtenir un débit maximal pour les transactions allouées par seconde (TPS), utilisez une architecture sans serveur de mise en file d'attente ou un autre mécanisme pour « fluidifier » le trafic afin qu'il soit plus cohérent. Pour obtenir des exemples de code et des références pour le traitement d'images et de vidéos à grande échelle sans serveur avec Rekognition, consultez la section Traitement d'images et de vidéos à grande échelle avec Amazon Rekognition.
2. Configurez de nouvelles tentatives. Suivez les instructions de la section the section called "Gestion des erreurs" pour configurer des erreurs qui les autorisent.
3. Backoff exponentiel et instabilité de l'ombre. La configuration d'un backoff exponentiel et d'une instabilité de l'ombre lors de la configuration des nouvelles tentatives vous permet d'améliorer le débit réalisable. Reportez-vous à la section Rétentatives d'erreur et recul exponentiel. AWS

Créez un dossier pour modifier les quotas TPS

Pour créer un dossier, rendez-vous sur Créer une demande et répondez aux questions suivantes :

- Avez-vous mis en œuvre le the section called "Bonnes pratiques relatives aux quotas TPS" pour atténuer vos pics de trafic et configurer les nouvelles tentatives, le backoff exponentiel et l'instabilité de l'ombre ?
- Avez-vous calculé le changement de quota TPS dont vous avez besoin ? Si ce n'est pas le cas, voyez the section called "Calculer la modification de quota TPS".
- Avez-vous vérifié l'historique de votre utilisation du TPS afin de prévoir avec plus de précision vos besoins futurs ? Pour consulter l'historique de votre utilisation du TPS, consultez la page Amazon Rekognition Service Quotas.
- Quel est votre cas d'utilisation ?
- Quelles API prévoyez-vous d'utiliser ?
- Dans quelles régions prévoyez-vous d'utiliser ces API ?
- Êtes-vous en mesure de répartir la charge sur plusieurs régions ?
- Combien d'images traitez-vous par jour ?
- Combien de temps comptez-vous maintenir ce volume (s'agit-il d'un pic ponctuel ou continu) ?
- Comment êtes-vous bloqué par la limite par défaut ? Consultez le tableau d'exceptions suivant pour confirmer le scénario que vous rencontrez.

Code d'erreur	Exception	Message	Qu'est-ce que cela signifie ?	Est-il possible de réessayer ?
Code d'état HTTP 400	ProvisionedThroughputExceededException	Provisioned Rate exceeded.	Indique une limitation. Vous pouvez réessayer ou évaluer une demande d'augmentation de limite.	Oui
Code d'état HTTP 400	ThrottlingException	Ralentir ; augmentation soudaine du taux de demandes.	Il se peut que vous envoyiez un trafic important et que vous soyez confronté à une limitation. Vous devez façonner le trafic et le rendre plus fluide et plus cohérent. Configurez ensuite des tentatives supplémentaires. Consultez les bonnes pratiques.	Oui

Code d'erreur	Exception	Message	Qu'est-ce que cela signifie ?	Est-il possible de réessayer ?
Code d'état HTTP 5xx	ThrottlingException (HTTP 500)	Service non disponible	Indique que le backend augmente pour prendre en charge l'action. Vous devez réessayer la demande.	Oui

Pour une compréhension détaillée des codes d'erreur, consultez the section called "Gestion des erreurs".

> **Note**
>
> Ces limites dépendent de la région dans laquelle vous vous trouvez. Le fait de présenter un argument en faveur de la modification d'une limite a une incidence sur l'opération d'API que vous demandez, dans la région où vous la demandez. Les autres opérations et régions de l'API ne sont pas affectées.

Historique du document pour Amazon Rekognition

Le tableau suivant décrit les modifications importantes dans chaque édition du Guide du développeur Amazon Rekognition. Pour recevoir les notifications de mise à jour de cette documentation, abonnez-vous à un flux RSS.

- Dernière date de mise à jour de la documentation : 15 juin 2023

Modification	Description	Date
Amazon Rekognition Streaming Video and Bulk Image Analysis n'est plus ouvert aux nouveaux clients.	Amazon Rekognition Streaming Video and Bulk Image Analysis n'est plus disponible pour les nouveaux clients. Les comptes existants qui ont utilisé ces fonctionn alités au cours des 12 derniers mois peuvent continuer à utiliser le service normaleme nt. Pour plus d'informations, consultez la section Modificat ions relatives à la disponibi lité des fonctionnalités de Rekognition.	30 avril 2026
Modification de la disponibilité des vidéos en streaming et de l'analyse d'images en masse d'Amazon Rekognition	Amazon Rekognition Streaming Video and Bulk Image Analysis ne sera plus ouvert aux nouveaux clients à compter du 30 avril 2026. Pour utiliser ces fonctionnalités avec de nouveaux comptes, veuillez le faire avant cette date. Les comptes existants qui ont utilisé ces fonctionn alités au cours des 12 derniers	31 mars 2026

	mois peuvent continuer à utiliser le service normalement. Pour plus d'informations, consultez la section Modifications relatives à la disponibilité des fonctionnalités de Rekognition.	
Amazon Rekognition prend désormais en charge les nouvelles étiquettes de modération et améliore la précision de la modération du contenu des images	La fonctionnalité de modération du contenu d'Amazon Rekognition a été améliorée pour améliorer la précision, la détection de nouvelles étiquettes et la capacité d'identifier le contenu illustré animé and/or .	1er février 2024
Amazon Rekognition prend désormais en charge l'analyse d'images en masse	Amazon Rekognition prend désormais en charge le traitement d'une grande collection d'images de manière asynchrone en utilisant un fichier manifeste avec l'opération. StartMediaAnalysisJob	23 octobre 2023
Amazon Rekognition prend désormais en charge la modération de contenu personnalisée avec des adaptateurs	Amazon Rekognition permet désormais d'améliorer la précision de l'API en utilisant des adaptateurs qui étendent les fonctionnalités DetectModerationLabels des modèles d'apprentissage profond de Rekognition existants.	12 octobre 2023

Rekognition prend désormais en charge les vecteurs utilisateur avec des collections	Les collections de visages Rekognition prennent désormais en charge la création de vecteurs utilisateur. Les vecteurs utilisateur regroupent plusieurs vecteurs faciaux d'un même utilisateur, ce qui améliore la précision grâce à des représentations plus robustes d'un utilisateur.	12 juin 2023
Des actions impliquant la gestion des utilisateurs ont été ajoutées aux politiques gérées suivantes : AmazonRekognitionReadOnlyAccess	Amazon Rekognition a ajouté les actions suivantes aux politiques gérées `AmazonRekognitionReadOnlyAccess` : `ListUsers`, `SearchUsers`, `SearchUsersByImage`	12 juin 2023
Image Amazon Rekognition peut désormais déduire la direction du regard	Des améliorations ont été apportées aux opérations de détection des visages d'Image Amazon Rekognition, qui peuvent désormais déduire la direction du regard d'un visage détecté.	31 mai 2023

API de modération du contenu Rekognition améliorée	Rekognition a amélioré le modèle de modération du contenu pour la modératio n des images et des vidéos. Cette amélioration étend considérablement la détection des contenus explicites, violents et suggestifs. Les clients peuvent désormais détecter les contenus explicite s et violents avec une plus grande précision afin d'amélior er l'expérience de l'utilisateur final, de protéger l'identité de leur marque et de s'assurer que tous les contenus sont conformes aux réglement ations et politiques de leur secteur d'activité.	9 mai 2023
Image Amazon Rekognition peut désormais détecter les visages occultés	Image Amazon Rekogniti on peut désormais détecter l'occlusion des visages. Un nouvel FaceOccluded attribut est renvoyé par les API et Amazon Rekogniti on DetectFaces Image, qui indique si le visage d'une image IndexFaces est partielle ment capturé ou s'il n'est pas entièrement visible en raison du chevauchement d'objets, de vêtements ou de parties du corps.	5 mai 2023

Rekognition peut désormais détecter la vivacité du visage	Vidéo Amazon Rekogniti on peut désormais être utilisé pour détecter la vivacité d'une vidéo, en vérifiant la présence physique d'un utilisateur devant une caméra. Le détecteur Face Liveness détecte également les attaques frauduleuses présentées à une caméra ou qui tentent de contourner une caméra.	11 avril 2023
Mise à jour de Vidéo Amazon Rekognition.	Vidéo Amazon Rekogniti on peut désormais détecter davantage d'étiquettes et renvoyer davantage d'informa tions sur les attributs des images et des étiquettes. L' GetLabelDetection API renvoie désormais des informations sur les alias et les catégorie s. Les informations d'étiquet te renvoyées peuvent être filtrées à l'aide d'options de filtrage inclusives et exclusive s. Les résultats peuvent être agrégés par horodatage ou par segments vidéo.	7 décembre 2022

Mise à jour d'Image Amazon Rekognition.	Image Amazon Rekogniti on peut désormais détecter davantage d'étiquettes et renvoie désormais davantage d'informations sur les attributs des images et des étiquettes. L' DetectLabels API renvoie désormais des informations sur les alias, les catégories et les propriétés des images, telles que les couleurs dominantes. Les informati ons d'étiquette renvoyées peuvent être filtrées à l'aide d'options de filtrage inclusives et exclusives.	11 novembre 2022
Les actions ProjectPolicy et la copie du modèle d'étiquet tes personnalisées ont été ajoutées aux politiques gérées suivantes : AmazonRek ognitionReadOnlyAccess	Amazon Rekognition a ajouté les actions suivantes aux politiques gérées `AmazonRek ognitionReadOnlyAc cess` : `ListProje ctPolicies`	21 juillet 2022
Les actions ProjectPo licy et la copie du modèle d'étiquettes personnal isées ont été ajoutées aux politiques gérées suivantes : AmazonRekognitionF ullAccess, AmazonRek ognitionCustomLabelsFullAcc ess	Amazon Rekognition a ajouté les actions suivantes aux politiques gérées `AmazonRek ognitionCustomLabe lsFullAccess` et `AmazonRekognitionF ullAccess` : `CopyProje ctVersion` , `PutProjec tPolicy` , `ListProje ctPolicies` , `DeletePro jectPolicy`	21 juillet 2022

Vidéo Amazon Rekognition peut désormais détecter les étiquettes dans les vidéos en streaming	Vidéo Amazon Rekognition peut détecter des étiquettes telles que colis et animaux domestiques dans les vidéos en streaming. Cette action s'effectue à l'aide des paramètres `ConnectedHome` des processeurs de flux créés lors de l'opération `CreateStreamProcessor` .	28 avril 2022
La référence d'API a été supprimée du guide du développeur Amazon Rekognition	La référence de l'API Amazon Rekognition est désormais disponible sur Référence de l'API Amazon Rekognition.	24 février 2022
Mise à jour de la gestion des jeux de données pour les politiques gérées suivantes :Politique gérée par AWS : AmazonRekognitionReadOnlyAccess,Politique gérée par AWS : AmazonRekognitionFullAccess, Politique gérée par AWS : AmazonRekognitionCustomLabelsFullAccess	Amazon Rekognition a ajouté les actions suivantes `AmazonRekognitionReadOnlyAccess` aux politiques et a géré `AmazonRekognitionFullOnlyAccess` les `CreateDataset` politiques `ListDatasetEntries` : `AmazonRekognitionCustomLabelsFullAccess` `ListDatasetLabels` ,,,,, `DescribeDataset` `UpdateDatasetEntries` `DistributeDatasetEntries` DeleteDataset	1er novembre 2021

Un nouveau nœud dans la table des matières présente des exemples d'Amazon Rekognition hébergés sur GitHub	Les exemples de code mis à jour provenant AWS du référentiel d'exemples de code apparaissent désormais dans un nœud distinct du guide du développeur Amazon Rekognition pour un accès plus facile.	22 octobre 2021
Amazon Rekognition peut détecter les cadres noirs et le contenu principal du programme dans les segments vidéo	Amazon Rekognition peut identifier les cadres noirs, les barres de couleur, les crédits d'ouverture, les crédits de fin, les logos des studios et le contenu principal du programme en tant qu'indices technique s dans une vidéo à l'aide des opérations `StartSegmentDetection` et `GetSegmentDetection` .	7 juin 2021
Mise à jour de la gestion des ensembles de données pour les politiques gérées suivantes :	Vous pouvez utiliser l'opération Amazon Rekognition `DetectText` pour détecter jusqu'à 100 mots dans une image.	21 mai 2021
Mise à jour du balisage pour et AmazonRekognitionReadOnlyAccessAmazonRekognitionFullAccess	Rekognition a ajouté de nouvelles actions de marquage aux politiques `AmazonRekognitionFullAccess` et `AmazonRekognitionReadOnlyAccess` .	2 avril 2021

Amazon Rekognition prend désormais en charge le balisage	Vous pouvez désormais utiliser des balises pour identifier, organiser, recherche r et filtrer les collections Amazon Rekognition, les processeurs de flux et les modèles d'étiquettes personnalisées.	25 mars 2021
Amazon Rekognition peut désormais détecter les équipements de protection individuelle	Amazon Rekognition peut désormais détecter les couvre-mains, les couvre-visages et les couvre-chefs sur les personnes figurant sur une image.	15 octobre 2020
Amazon Rekognition propose de nouvelles catégories de modération de contenu	Les catégories de modération du contenu Amazon Rekogniti on incluent désormais 6 nouvelles catégories : drogues, tabac, alcool, jeux de hasard, gestes grossiers et symboles haineux.	12 octobre 2020
Nouveau didacticiel pour afficher localement les résultats de Vidéo Amazon Rekognition à partir de Kinesis Video Streams	Vous pouvez afficher le résultat de Vidéo Amazon Rekognition à partir d'une vidéo en streaming dans Kinesis Video Streams dans un flux vidéo local.	20 juillet 2020

Nouveau didacticiel Amazon Rekognition pour l'utilisation de Gstreamer	À l'aide de Gstreamer, vous pouvez intégrer une vidéo diffusée en direct depuis la source de la caméra d'un appareil vers Vidéo Amazon Rekognition via Kinesis Video Streams.	17 juillet 2020
Amazon Rekognition prend désormais en charge la segmentation des vidéos stockées	Avec l'API de segmentation asynchrone Vidéo Amazon Rekognition, vous pouvez détecter les cadres noirs, les barres de couleur, les génériques de fin et les prises de vue dans les vidéos stockées.	22 juin 2020
Amazon Rekognition prend désormais en charge les stratégies de point de terminaison d'un VPC Amazon	En spécifiant une stratégie , vous pouvez restreindre l'accès à un point de terminaison d'un VPC Amazon.	3 mars 2020
Amazon Rekognition prend désormais en charge la détection de texte dans les vidéos stockées	Vous pouvez utiliser l'API Vidéo Amazon Rekognition pour détecter de manière asynchrone le texte d'une vidéo enregistrée.	17 février 2020

Amazon Rekognition prend désormais en charge l'IA augmentée (version préliminaire) et les Étiquettes personnalisées Amazon Rekognition	Avec les étiquettes personnalisées Amazon Rekognition, vous pouvez détecter des objets, des scènes et des concepts spécialisés dans les images en créant votre propre modèle d'apprentissage automatique. DetectModerationLabels prend désormais en charge Amazon Augmented AI (version préliminaire).	3 décembre 2019
Amazon Rekognition prend désormais en charge AWS PrivateLink	Avec AWS, PrivateLink vous pouvez établir une connexion privée entre votre VPC et Amazon Rekognition.	12 septembre 2019
Filtrage du visage Amazon Rekognition	Amazon Rekognition ajoute une prise en charge améliorée du filtrage des visages au fonctionnement de l'API et introduit le filtrage des visages IndexFaces pour les opérations de l'API et de l'API. CompareFaces SearchFacesByImage	12 septembre 2019
Exemples de vidéos Vidéo Amazon Rekognition mis à jour	Exemple de code Vidéo Amazon Rekognition mis à jour pour créer et configurer la rubrique Amazon SNS et la file d'attente Amazon SQS.	5 septembre 2019

Ruby et Node.js exemples ajoutés	Amazon Rekognition Image Node.js Ruby et exemples ajoutés pour la détection synchrone d'étiquettes et de visages.	19 août 2019
Mise à jour de la détection de contenu inapproprié	La détection de contenu inapproprié Amazon Rekogniti on peut désormais détecter du contenu violent.	9 août 2019
GetContentModeration opération mise à jour	GetContentModeration renvoie désormais la version du modèle de détection de modération utilisé pour détecter les contenus non sécurisés.	13 février 2019
GetLabelDetection et DetectModerationLabels opérations mises à jour	GetLabelDetection renvoie désormais des informations de cadre pour les objets courants et une taxonomie hiérarchi que des étiquettes détectées . La version du modèle utilisé pour la détection des étiquette s est désormais renvoyée. DetectModerationLabels renvoie désormais la version du modèle utilisé pour détecter les contenus non sécurisés.	le 17 janvier 2019

DetectFaces et IndexFaces fonctionnement mis à jour	Cette version met à jour le IndexFaces fonctionnement DetectFaces et. Lorsque le paramètre d'entrée Attributes est défini sur ALL, les repères de localisation des visages incluent 5 nouveaux points de repère : upperJawlineLeft, midJawlineLeft, chinBotto m, midJawlineRight, upper. JawlineRight	19 novembre 2018
DetectLabels opération mise à jour	Des cadres de limitation sont désormais renvoyés pour certains objets. Une taxonomie hiérarchique est désormais disponible pour les étiquette s. Vous pouvez désormais obtenir la version du modèle de détection utilisé pour la détection.	1er novembre 2018
IndexFaces opération mise à jour	Avec, IndexFaces vous pouvez désormais utiliser le paramètre QualityFilter d'entrée pour filtrer les visages détectés avec une faible qualité. Vous pouvez également utiliser le paramètre MaxFaces d'entrée pour réduire le nombre de visages renvoyés en fonction de la qualité de la détection des visages et de la taille du visage détecté.	18 septembre 2018

DescribeCollection opération ajoutée	Vous pouvez désormais obtenir des informations sur une collection existante en appelant l' DescribeCollection opération.	22 août 2018
Nouveaux exemples Python	Des exemples Python ont été ajoutés au contenu vidéo Vidéo Amazon Rekognition avec des réorganisations de contenu.	26 juin 2018
Disposition du contenu mise à jour	Le contenu d'Image Amazon Rekognition a été réorganisé avec de nouveaux exemples Python et C#.	29 mai 2018
Amazon Rekognition prend en charge AWS CloudTrail	Amazon Rekognition est AWS CloudTrail intégré à un service qui fournit un enregistrement des actions effectuées par un utilisateur, un rôle ou un AWS service dans Amazon Rekognition. Pour plus d'informations, consultez la section Journalisation des appels d'API Amazon Rekognition avec AWS. CloudTrail	6 avril 2018

Analysez les vidéos stockées et en streaming. Nouvelle table des matières	Pour plus d'informations sur l'analyse des vidéos stockées, consultez Utilisation des vidéos stockées. Pour plus d'informations sur l'analyse des vidéos en streaming , consultez Utilisation des vidéos streaming. La table des matières de la documentation Amazon Rekognition a été réorganisée afin d'inclure les opérations relatives aux images et aux vidéos.	29 novembre 2017
Modèles de détection de texte dans les images et de détection des visages	Amazon Rekognition peut désormais détecter le texte dans les images. Pour plus d'informations, consultez Détection de texte. Amazon Rekognition présente la gestion des versions pour le modèle de deep learning de détection des visages. Pour plus d'informations, consultez Gestion des versions.	21 novembre 2017
Reconnaissance de célébrités	Amazon Rekognition peut désormais analyser des images afin de rechercher la présence de célébrités. Pour plus d'informations, consultez Reconnaissance de célébrités.	8 juin 2017

Modération des images	Amazon Rekognition peut désormais déterminer si une image contient un contenu pour adulte explicite ou suggestif. Pour plus d'informations, consultez Détection de contenu inapproprié.	19 avril 2017
Tranches d'âge pour la détection des visages. Volet Métriques de reconnaissance agrégées	Amazon Rekognition retourne désormais la tranche d'âge estimée, en années, pour les visages détectés par l'API Rekognition. Pour de plus amples informations, veuillez consulter AgeRange. La console Rekognition dispose désormais d'un volet de statistiques présentant des graphiques d'activité pour un agrégat de CloudWatch métriques Amazon relatives à Rekognition sur une période donnée. Pour plus d'informations, consultez Exercice 4 : Afficher les métriques agrégées (console).	9 février 2017
Nouveau guide et service	Il s'agit de la version initiale du service d'analyse d'image, Amazon Rekognition, et du Manuel du développeur Amazon Rekognition.	30 novembre 2016

AWS Glossaire

Pour la AWS terminologie la plus récente, consultez le AWS glossaire dans la Glossaire AWS référence.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.