



Guide des étiquettes personnalisées

Rekognition



Rekognition: Guide des étiquettes personnalisées

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

Les marques et la présentation commerciale d'Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n'est pas d'Amazon, d'une manière susceptible de créer une confusion parmi 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 Custom Labels ?	1
Bénéfices clés	2
Choisir d'utiliser Amazon Rekognition Custom Labels	2
Détection d'étiquettes Amazon Rekognition Image	3
Étiquettes Amazon Rekognition	3
Vous utilisez Amazon Rekognition Custom Labels ?	4
Configuration d'Étiquettes personnalisées Amazon Rekognition	5
Étape 1 : Créer un compte AWS	5
S'inscrire à un Compte AWS	6
Création d'un utilisateur administratif	6
Accès par programmation	7
Étape 2 : Configurer les autorisations de la console	9
Autorisation de l'accès à la console	10
Accès à des compartiments Amazon S3 externes	11
Attribution d'autorisations	12
Étape 3 : Créer le compartiment de la console	12
Étape 4 : Configurer AWS CLI et les kits AWS SDK	13
Installation des kits AWS SDK	13
Octroi d'un accès par programmation	7
Configuration des autorisations du kit SDK	18
Appel d'une opération	20
Étape 5 (Facultatif) : Chiffrer les fichiers d'entraînement	24
Déchiffrement de fichiers chiffrés avec AWS Key Management Service	25
Chiffrement de copies d'images d'entraînement et de test	25
Étape 6 (Facultatif) : Associer des jeux de données précédents	26
Utilisation d'un jeu de données précédent comme jeu de données de test	27
Comprendre les étiquettes personnalisées Amazon Rekognition	28
Choisissez votre type de modèle	28
Trouvez des objets, des scènes et des concepts	29
Trouver l'emplacement des objets	30
Recherchez l'emplacement des marques	30
Création d'un modèle	31
Créer un projet	31
Création de jeux de données de formation et de test	32

Entraînez votre modèle	33
Améliorez votre modèle	34
Évaluer votre modèle	34
Améliorez votre modèle	35
Démarrez votre modèle.	35
Démarrez votre modèle (console).	36
Démarrez votre modèle.	36
Analyser une image	36
Arrêter votre modèle	38
Arrêtez votre modèle (console)	38
Arrêtez votre modèle (SDK)	38
Démarrer	39
Vidéos tutorielles	39
Exemples de projets	40
Image classification	40
Classification d'images à étiquettes multiples	40
Détection de marque	41
Localisation d'objets	41
À l'aide des exemples de projets	42
Création de l'exemple de projet	42
Entraînement du modèle	43
Utilisation du modèle	43
Étapes suivantes	43
Étape 1 : Choisissez un exemple de projet	44
Étape 2 : Entraînez votre modèle	47
Étape 3 : Démarrez votre modèle	51
Étape 4 : Analyser une image avec votre modèle	52
Obtenir un exemple d'image	57
Étape 5 : arrêtez votre modèle	59
Étape 6 : étapes suivantes	61
Tutoriel : Classification d'images	63
Étape 1 : Collectez vos images	63
Étape 2 : Choisissez vos cours	65
Étape 3 : Création d'un projet	65
Étape 4 : créer des ensembles de données d'entraînement et de test	66
Étape 5 : Ajouter des étiquettes au projet	72

Étape 6 : Attribuer des étiquettes au niveau de l'image aux ensembles de données d'entraînement et de test	73
Étape 7 : Entraînez votre modèle	74
Étape 8 : Démarrez votre modèle	79
Étape 9 : Analyser une image avec votre modèle	81
Étape 10 : arrêtez votre modèle	84
Création d'un modèle	87
Crée un projet.	87
Création d'un projet (console)	87
Création d'un projet (SDK)	88
Création de jeux de données	93
Utilisation des jeux de données	94
Préparation des images	100
Création de jeux de données avec des images	101
Étiquetage des images	164
Débogage des jeux de données	175
Entraînement d'un modèle	182
Entraînement d'un modèle (console)	184
Entraînement d'un modèle (SDK)	188
Formation sur les modèles de modèle de modèle	198
Erreurs de terminal	199
Erreurs de validation de lignes JSON non terminales	201
Présentation du récapitulatif du manifeste	202
Comprendre les manifestes de résultats de validation des formations et des tests	206
Obtenir les résultats de validation	212
Corriger les erreurs d'entraînement	215
Erreurs du fichier manifeste du terminal	217
Erreurs de contenu du manifeste du terminal	219
Erreurs de validation de lignes JSON non terminales	229
Améliorer un modèle entraîné	254
Métriques pour évaluer votre modèle	254
Évaluation des performances de modèle	255
Seuil supposé	256
Précision	257
Sensibilité	257
F1	258

Utilisation de métriques	258
Accès aux mesures d'évaluation (console)	259
Accès aux métriques d'évaluation (SDK)	261
Fichier récapitulatif	262
Aperçu du manifeste d'évaluation	264
Accès au fichier récapitulatif et à l'instantané du manifeste d'évaluation (SDK)	268
Affichage de la matrice de confusion d'un modèle	269
Référence : fichier récapitulatif	276
Amélioration d'un modèle	278
non structurées	279
Réduire les faux positifs (meilleure précision)	279
Réduire les faux négatifs (meilleure mémorisation)	280
Exécution d'un modèle entraîné	281
Unités d'inférence	281
Gestion du débit à l'aide d'unités d'inférence	282
Zones de disponibilité	284
Démarrage d'un modèle	285
Démarrage ou arrêt d'un modèle (console)	286
Démarrage d'un modèle (kit SDK)	287
Arrêt d'un modèle	297
Arrêt d'un modèle (console)	297
Arrêt d'un modèle (kit SDK)	298
Rapport sur la durée et les unités d'inférence	306
Analyse d'une image	310
DetectCustomLabels demande d'opération	337
DetectCustomLabels réponse à l'opération	337
Gestion des ressources	339
Gestion d'un projet	339
Suppression d'un projet	339
Décrire un projet (SDK)	349
Création d'un projet avec AWS CloudFormation	356
Gestion des jeux de données	357
Ajouter un ensemble de données.	357
Ajouter d'autres images	367
Création d'un jeu de données à l'aide d'un jeu de données existant (SDK)	377
Décrire un ensemble de données (SDK)	386

Répertorier les entrées du jeu de données (SDK)	391
Distribution d'un jeu de données d'entraînement (SDK)	397
Supprimer un ensemble de données.	407
Gérer un modèle	415
Suppression d'un modèle	415
Marquer un modèle	424
Décrire un modèle (SDK)	432
Copier un modèle (SDK)	440
Exemples	478
Solution de feedback sur les modèles	478
Démonstration des étiquettes personnalisées Amazon Rekognition	479
Analyse vidéo	479
Analyse d'images à l'aide d'unAWS Lambda fonction	482
Étape 1 : Création d'unAWS Lambda fonction (console)	482
Étape 2 : (Facultatif) Création d'une couche (console)	485
Étape 3 : Ajouter du code Python (console)	486
Étape 4 : essayez votre fonction Lambda	489
Sécurité	494
Sécurisation des projets d'étiquettes personnalisées Amazon Rekognition	494
SécurisationDetectCustomLabels	496
Politiques gérées par AWS	496
Con	497
Régions prises en charge	497
Quotas	497
Entraînement	497
Test	498
Détection	499
Copie de modèles	499
Référence d'API	500
Entraînement votre modèle	511
Projets	511
Politiques d'un projet	511
Jeux de données	511
Modèles	512
Étiquettes	511
Utilisation votre modèle	512

Historique de document	513
Glossaire AWS	520
.....	dxxi

Qu'est-ce qu'Amazon Rekognition Custom Labels ?

Avec les étiquettes personnalisées Amazon Rekognition, vous pouvez identifier les objets, les logos et les scènes dans les images qui sont spécifiques aux besoins de votre entreprise. Par exemple, vous pouvez trouver votre logo dans des publications sur les réseaux sociaux, identifier vos produits dans les rayons des magasins, classer les pièces de machines sur une chaîne de montage, distinguer les plantes saines des plantes infectées ou détecter des personnages animés sur des images.

Le développement d'un modèle personnalisé pour analyser des images est une entreprise importante qui nécessite du temps, de l'expertise et des ressources. Il faut souvent des mois pour le terminer. En outre, des milliers ou des dizaines de milliers d'images étiquetées à la main peuvent être nécessaires pour fournir au modèle suffisamment de données pour prendre des décisions précises. La génération de ces données peut prendre des mois et nécessiter de grandes équipes d'étiqueteurs pour les préparer en vue de leur utilisation dans le cadre de l'apprentissage automatique.

Amazon Rekognition Custom Labels étend les fonctionnalités existantes d'Amazon Rekognition, qui sont déjà appliquées à des dizaines de millions d'images dans de nombreuses catégories. Au lieu de milliers d'images, vous pouvez télécharger un petit ensemble d'images d'entraînement (généralement quelques centaines d'images ou moins) spécifiques à votre cas d'utilisation. Pour ce faire, utilisez la easy-to-use console. Si vos images sont déjà étiquetées, Amazon Rekognition Custom Labels peut commencer à entraîner un modèle en peu de temps. Sinon, vous pouvez étiqueter les images directement dans l'interface d'étiquetage, ou vous pouvez utiliser Amazon SageMaker Ground Truth pour les étiqueter à votre place.

Une fois qu'Amazon Rekognition Custom Labels a commencé à s'entraîner à partir de votre ensemble d'images, il peut créer un modèle d'analyse d'images personnalisé pour vous en quelques heures seulement. Dans les coulisses, Amazon Rekognition Custom Labels charge et inspecte automatiquement les données d'entraînement, sélectionne les bons algorithmes d'apprentissage automatique, forme un modèle et fournit des mesures de performance du modèle. Vous pouvez ensuite utiliser votre modèle personnalisé via l'API Amazon Rekognition Custom Labels et l'intégrer à vos applications.

Rubriques

- [Bénéfices clés](#)
- [Choisir d'utiliser Amazon Rekognition Custom Labels](#)
- [Vous utilisez Amazon Rekognition Custom Labels ?](#)

Bénéfices clés

Étiquetage simplifié des données

La console Amazon Rekognition Custom Labels fournit une interface visuelle qui permet d'étiqueter vos images rapidement et simplement. L'interface permet d'appliquer une étiquette à l'ensemble de l'image. Vous pouvez également identifier et étiqueter des objets spécifiques dans des images à l'aide de cadres de sélection dotés d'une click-and-drag interface. Sinon, si vous disposez d'un ensemble de données volumineux, vous pouvez utiliser [Amazon SageMaker Ground Truth](#) pour étiqueter efficacement vos images à grande échelle.

Machine learning

Aucune expertise en apprentissage automatique n'est requise pour créer votre modèle personnalisé. Amazon Rekognition Custom Labels inclut des fonctionnalités d'apprentissage automatique (AutoML) qui prennent en charge l'apprentissage automatique à votre place. Lorsque les images d'apprentissage sont fournies, les étiquettes personnalisées Amazon Rekognition peuvent automatiquement charger et inspecter les données, sélectionner les bons algorithmes d'apprentissage automatique, former un modèle et fournir des mesures de performance du modèle.

Évaluation, inférence et retour d'informations simplifiés des modèles

Vous évaluez les performances de votre modèle personnalisé sur votre ensemble de tests. Pour chaque image de l'ensemble de test, vous pouvez voir la side-by-side comparaison entre la prédiction du modèle et l'étiquette attribuée. Vous pouvez également consulter des indicateurs de performance détaillés tels que la précision, le rappel, les scores F1 et les scores de confiance. Vous pouvez commencer à utiliser votre modèle immédiatement pour l'analyse d'images, ou vous pouvez itérer et réentraîner les nouvelles versions avec davantage d'images pour améliorer les performances. Une fois que vous avez commencé à utiliser votre modèle, vous suivez vos prévisions, vous corrigez les erreurs et vous utilisez les données de feedback pour adapter les nouvelles versions du modèle et améliorer les performances.

Choisir d'utiliser Amazon Rekognition Custom Labels

Amazon Rekognition propose deux fonctionnalités que vous pouvez utiliser pour rechercher des étiquettes (objets, scènes et concepts) dans des images : Amazon Rekognition Custom [Labels et Amazon Rekognition Image](#). Utilisez les informations suivantes pour déterminer la fonctionnalité à utiliser.

Détection d'étiquettes Amazon Rekognition Image

Vous pouvez utiliser la fonction de détection d'étiquettes d'Amazon Rekognition Image pour identifier, classer et rechercher des étiquettes courantes dans des images et des vidéos, à grande échelle et sans avoir à créer de modèle d'apprentissage automatique. Par exemple, vous pouvez facilement détecter des milliers d'objets courants, tels que des voitures et des camions, des tomates, des ballons de basket et des ballons de football.

Si votre application a besoin de trouver des étiquettes courantes, nous vous recommandons d'utiliser la détection d'étiquettes Amazon Rekognition Image, car vous n'avez pas besoin de former de modèle. Pour obtenir la liste des étiquettes [détectées par la détection d'étiquettes Amazon Rekognition Image](#), consultez la section [Détection d'étiquettes](#).

Si votre application a besoin de trouver des étiquettes non détectées par la détection d'étiquettes Amazon Rekognition Image, telles que des pièces de machine personnalisées sur une chaîne de montage, nous vous recommandons d'utiliser les étiquettes personnalisées Amazon Rekognition.

Étiquettes Amazon Rekognition

Vous pouvez utiliser les étiquettes personnalisées Amazon Rekognition pour créer facilement un modèle d'apprentissage automatique capable de trouver des étiquettes (objets, logos, scènes et concepts) dans des images adaptées aux besoins de votre entreprise.

Les étiquettes personnalisées Amazon Rekognition peuvent classer les images (prévisions au niveau de l'image) ou détecter l'emplacement des objets dans une image (prévisions au niveau de l'objet/du cadre de délimitation).

Les étiquettes personnalisées Amazon Rekognition offrent une plus grande flexibilité quant aux types d'objets et de scènes que vous pouvez détecter. Par exemple, vous pouvez utiliser la détection d'étiquettes Amazon Rekognition Image pour trouver des plantes et des feuilles. Pour distinguer les plantes saines, endommagées et infectées, vous devez utiliser les étiquettes personnalisées Amazon Rekognition.

Les exemples suivants illustrent la manière dont vous pouvez utiliser Amazon Rekognition Custom Labels.

- Identifiez les logos des équipes sur les maillots et les casques des joueurs
- Distinguer des pièces de machines ou des produits spécifiques sur une chaîne de montage
- Identifier des personnages de dessins animés dans une médiathèque

- Localisez les produits d'une marque spécifique dans les rayons des magasins
- Classifiez la qualité des produits agricoles (pourris, mûrs ou crus)

Note

Les étiquettes personnalisées Amazon Rekognition ne sont pas conçues pour analyser des visages, détecter du texte ou détecter du contenu d'image dangereux dans les images. Pour effectuer ces tâches, vous pouvez utiliser Amazon Rekognition Image. Pour de plus amples informations, [utilisez Amazon Rekognition](#).

Vous utilisez Amazon Rekognition Custom Labels ?

Si vous utilisez Amazon Rekognition Custom Labels pour la première fois, nous vous recommandons de lire les sections suivantes dans l'ordre :

1. [Configuration d'Étiquettes personnalisées Amazon Rekognition](#)— Dans cette section, vous définissez les détails de votre compte.
2. [Comprendre les étiquettes personnalisées Amazon Rekognition](#)— Dans cette section, vous découvrirez le processus de création d'un modèle.
3. [Démarez avec les étiquettes personnalisées Amazon Rekognition](#)— Dans cette section, vous entraînez un modèle à l'aide d'exemples de projets créés par Amazon Rekognition Custom Labels.
4. [Tutoriel : Classification d'images](#)— Dans cette section, vous apprendrez à entraîner un modèle qui classe les images à l'aide de jeux de données que vous créez.

Configuration d'Étiquettes personnalisées Amazon Rekognition

Les instructions suivantes vous expliquent comment configurer la console Étiquettes personnalisées Amazon Rekognition et le kit SDK.

Notez que vous pouvez utiliser la console Étiquettes personnalisées Amazon Rekognition avec les navigateurs suivants :

- Chrome version 21 ou ultérieure
- Firefox version 27 ou ultérieure
- Microsoft Edge version 88 ou ultérieure
- Safari version 7 ou ultérieure. En outre, vous ne pouvez pas utiliser Safari pour dessiner des cadres de délimitation avec la console Étiquettes personnalisées Amazon Rekognition. Pour plus d'informations, consultez [Étiquetage des objets à l'aide de cadres de délimitation](#).

Avant d'utiliser Étiquettes personnalisées Amazon Rekognition pour la première fois, exécutez les tâches suivantes :

Rubriques

- [Étape 1 : Créer un compte AWS](#)
- [Étape 2 : Configurer les autorisations d'accès à la console Étiquettes personnalisées Amazon Rekognition](#)
- [Étape 3 : Créer le compartiment de la console](#)
- [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#)
- [Étape 5 \(Facultatif\) : Chiffrer les fichiers d'entraînement](#)
- [Étape 6 \(Facultatif\) : Associer des jeux de données précédents à de nouveaux projets](#)

Étape 1 : Créer un compte AWS

Au cours de cette étape, vous allez créer un compte AWS ainsi qu'un utilisateur administratif, et apprendre à accorder un accès par programmation au kit AWS SDK.

Rubriques

- [S'inscrire à un Compte AWS](#)
- [Création d'un utilisateur administratif](#)
- [Accès par programmation](#)

S'inscrire à un Compte AWS

Si vous n'avez pas de compte Compte AWS, procédez comme suit pour en créer un.

Pour s'inscrire à un Compte AWS

1. Ouvrez <https://portal.aws.amazon.com/billing/signup>.
2. Suivez les instructions en ligne.

Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique et vous saisissez un code de vérification en utilisant le clavier numérique du téléphone.

Lorsque vous souscrivez à un Compte AWS, un Utilisateur racine d'un compte AWS est créé. Par défaut, seul l'utilisateur root a accès à l'ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à [attribuer un accès administratif à un utilisateur administratif](#), et à uniquement utiliser l'utilisateur root pour effectuer [tâches nécessitant un accès utilisateur root](#).

AWS vous envoie un e-mail de confirmation lorsque le processus d'inscription est terminé. Vous pouvez afficher l'activité en cours de votre compte et gérer votre compte à tout moment en accédant à <https://aws.amazon.com/> et en cliquant sur Mon compte.

Création d'un utilisateur administratif

Après vous être inscrit à un Compte AWS, sécurisez votre Utilisateur racine d'un compte AWS, activez AWS IAM Identity Center, puis créez un utilisateur administratif afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisation de votre Utilisateur racine d'un compte AWS

1. Connectez-vous à la [AWS Management Console](#) en tant que propriétaire du compte en sélectionnant Root user (utilisateur root) et en saisissant l'adresse e-mail de Compte AWS. Sur la page suivante, saisissez votre mot de passe.

Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur root, consultez [Connexion en tant qu'utilisateur root](#) dans le Guide de l'utilisateur Connexion à AWS.

2. Activez l'authentification multifactorielle (MFA) pour votre utilisateur root.

Pour obtenir des instructions, consultez [Activation d'un dispositif MFA virtuel pour l'utilisateur root de votre Compte AWS \(console\)](#) dans le Guide de l'utilisateur IAM.

Création d'un utilisateur administratif

1. Activez IAM Identity Center.

Pour obtenir des instructions, consultez [Configuration d'AWS IAM Identity Center](#) dans le guide de l'utilisateur AWS IAM Identity Center.

2. Dans IAM Identity Center, octroyez un accès administratif à un utilisateur administratif.

Pour profiter d'un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, consultez [Configuration de l'accès utilisateur avec le répertoire Répertoire IAM Identity Center par défaut](#) dans le Guide de l'utilisateur AWS IAM Identity Center.

Connexion en tant qu'utilisateur administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l'utilisateur IAM Identity Center.

Pour obtenir de l'aide pour vous connecter à l'aide d'un utilisateur IAM Identity Center, consultez [Connexion au portail d'accès AWS](#) dans le Guide de l'utilisateur Connexion à AWS.

Accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS en dehors de la AWS Management Console. La manière d'octroyer un accès par programmation dépend du type d'utilisateur qui accède à AWS.

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

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de l'interface que vous souhaitez utiliser. <ul style="list-style-type: none"> • Pour l'AWS CLI, veuillez consulter la rubrique Configuration de l'AWS CLI pour l'utilisation d'AWS IAM Identity Center dans le Guide de l'utilisateur AWS Command Line Interface. • Pour les kits AWS SDK, les outils et les API AWS, consultez Authentification IAM Identity Center dans le Guide de référence des kits SDK et des outils AWS.
IAM	Utilisez des informations d'identification temporaires pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec des ressources AWS dans le Guide de l'utilisateur IAM.
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de l'interface que vous souhaitez utiliser. <ul style="list-style-type: none"> • Pour l'AWS CLI, veuillez consulter la rubrique Authentification à l'aide des informations d'identification d'utilisateur IAM dans le

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
		<p>Guide de l'utilisateur AWS Command Line Interface.</p> <ul style="list-style-type: none">• Pour les kits SDK et les outils AWS, veuillez consulter la rubrique Authentification à l'aide d'informations d'identification à long terme dans le Guide de référence des kits SDK et des outils AWS.• Pour les API AWS, veuillez consulter la rubrique Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Étape 2 : Configurer les autorisations d'accès à la console Étiquettes personnalisées Amazon Rekognition

Pour utiliser la console Amazon Rekognition, vous devez ajouter les autorisations appropriées. Si vous souhaitez stocker vos fichiers d'entraînement dans un autre compartiment que le [compartiment de la console](#), vous avez besoin d'autorisations supplémentaires.

Rubriques

- [Autorisation de l'accès à la console](#)
- [Accès à des compartiments Amazon S3 externes](#)
- [Attribution d'autorisations](#)

Autorisation de l'accès à la console

Pour utiliser la console Amazon Rekognition Custom Labels, vous avez besoin de la politique IAM suivante qui couvre Amazon S3 SageMaker , Ground Truth et Amazon Rekognition Custom Labels. Pour plus d'informations sur l'attribution d'autorisations, consultez [Attribution d'autorisations](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "*"
    },
    {
      "Sid": "s3Policies",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:CreateBucket",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectVersion",
        "s3:GetObjectTagging",
        "s3:GetBucketVersioning",
        "s3:GetObjectVersionTagging",
        "s3:PutBucketCORS",
        "s3:PutLifecycleConfiguration",
        "s3:PutBucketPolicy",
        "s3:PutObject",
        "s3:PutObjectTagging",
        "s3:PutBucketVersioning",
        "s3:PutObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::custom-labels-console-*"
      ]
    }
  ]
}
```

```
    },
    {
      "Sid": "rekognitionPolicies",
      "Effect": "Allow",
      "Action": [
        "rekognition:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "groundTruthPolicies",
      "Effect": "Allow",
      "Action": [
        "groundtruthlabeling:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Accès à des compartiments Amazon S3 externes

Lorsque vous ouvrez la console Étiquettes personnalisées Amazon Rekognition pour la première fois dans une nouvelle région AWS, elle crée un compartiment (compartiment de la console) qui sert à stocker les fichiers du projet. Vous pouvez également utiliser votre propre compartiment Amazon S3 (compartiment externe) pour charger les images ou le fichier manifeste dans la console. Pour utiliser un compartiment externe, ajoutez le bloc de stratégie suivant à la stratégie précédente. Remplacez `my-bucket` par le nom du compartiment.

```
{
  "Sid": "s3ExternalBucketPolicies",
  "Effect": "Allow",
  "Action": [
    "s3:GetBucketAcl",
    "s3:GetBucketLocation",
    "s3:GetObject",
    "s3:GetObjectAcl",
    "s3:GetObjectVersion",
    "s3:GetObjectTagging",
    "s3:ListBucket",
    "s3:PutObject"
  ],
}
```

```
"Resource": [  
  "arn:aws:s3:::my-bucket*"  
]  
}
```

Attribution d'autorisations

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. 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.

Étape 3 : Créer le compartiment de la console

Vous utilisez un projet Étiquettes personnalisées Amazon Rekognition pour créer et gérer vos modèles. Lorsque vous ouvrez la console Étiquettes personnalisées Amazon Rekognition pour la première fois dans une nouvelle région AWS, elle crée un compartiment Amazon S3 (compartiment de la console) pour stocker vos projets. Notez le nom du compartiment de la console dans un endroit où vous pourrez vous y référer ultérieurement, car vous pourriez en avoir besoin pour des opérations du kit AWS SDK ou des tâches de la console (création d'un jeu de données, par exemple).

Le format du nom du compartiment est `custom-labels-console-<région>-<valeur aléatoire>`. La valeur aléatoire garantit l'absence de conflit entre les noms de compartiments.

Pour créer le compartiment de la console

1. Assurez-vous que l'utilisateur dispose des autorisations adéquates. Pour plus d'informations, consultez [Autorisation de l'accès à la console](#).
2. Connectez-vous à la AWS Management Console et ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
3. Choisissez Démarrer.
4. Si c'est la première fois que vous ouvrez la console dans cette région AWS, procédez comme suit dans la boîte de dialogue Première configuration :
 - a. Copiez le nom du compartiment Amazon S3 qui s'affiche. Vous aurez besoin de ces informations ultérieurement.
 - b. Choisissez Créer un compartiment Amazon S3 pour permettre à Étiquettes personnalisées Amazon Rekognition de créer un compartiment Amazon S3 (compartiment de la console) en votre nom.
5. Fermez la fenêtre du navigateur.

Étape 4 : Configurer AWS CLI et les kits AWS SDK

Vous pouvez utiliser Étiquettes personnalisées Amazon Rekognition avec l'AWS Command Line Interface (AWS CLI) et les kits AWS SDK. Si vous devez exécuter des opérations Étiquettes personnalisées Amazon Rekognition depuis le terminal, installez AWS CLI. Si vous créez une application, téléchargez le kit SDK AWS correspondant au langage de programmation que vous utilisez.

Rubriques

- [Installation des kits AWS SDK](#)
- [Octroi d'un accès par programmation](#)
- [Configuration des autorisations du kit SDK](#)
- [Appel d'une opération Étiquettes personnalisées Amazon Rekognition](#)

Installation des kits AWS SDK

Suivez les étapes permettant de télécharger et configurer les kits AWS SDK.

Pour configurer l’AWS CLI et les kits SDK AWS

- Téléchargez et installez [AWS CLI](#) et les kits AWS SDK que vous souhaitez utiliser. Ce guide fournit des exemples pour AWS CLI, [Java](#) et [Python](#). Pour en savoir plus sur l’installation des kits AWS SDK, consultez [Outils pour Amazon Web Services](#).

Octroi d'un accès par programmation

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

Rubriques

- [Exécuter du code sur votre ordinateur local](#)
- [Exécution de code dans des environnements AWS](#)

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 SDK AWS. Pour des informations spécifiques sur l’exécution des exemples de code AWS CLI et sur un ordinateur local, consultez [Utiliser un profil sur votre ordinateur local](#).

Les utilisateurs ont besoin d’un accès programmatique s’ils souhaitent interagir avec AWS en dehors de la AWS Management Console. La manière d’octroyer un accès par programmation dépend du type d’utilisateur qui accède à AWS.

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

Quel utilisateur a besoin d’un accès programmatique ?	Pour	Par
Identité de la main-d’œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d’identification temporaires pour signer des demandes par programmation destinées à	Suivez les instructions de l’interface que vous souhaitez utiliser.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
	l'AWS CLI, aux kits SDK AWS ou aux API AWS.	<ul style="list-style-type: none">• Pour l'AWS CLI, veuillez consulter la rubrique Configuration de l'AWS CLI pour l'utilisation d'AWS IAM Identity Center dans le Guide de l'utilisateur AWS Command Line Interface.• Pour les kits AWS SDK, les outils et les API AWS, consultez Authentification IAM Identity Center dans le Guide de référence des kits SDK et des outils AWS.
IAM	Utilisez des informations d'identification temporaires pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec des ressources AWS dans le Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
IAM	<p>(Non recommandé)</p> <p>Utilisez des informations d'identification à long terme pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.</p>	<p>Suivez les instructions de l'interface que vous souhaitez utiliser.</p> <ul style="list-style-type: none"> • Pour l'AWS CLI, veuillez consulter la rubrique Authentification à l'aide des informations d'identification d'utilisateur IAM dans le Guide de l'utilisateur AWS Command Line Interface. • Pour les kits SDK et les outils AWS, veuillez consulter la rubrique Authentification à l'aide d'informations d'identification à long terme dans le Guide de référence des kits SDK et des outils AWS. • Pour les API AWS, veuillez consulter la rubrique 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 l'AWS CLI et les exemples de code présentés dans ce guide à l'aide des informations d'identification à court terme que vous avez créées dans ce guide [Exécuter du code sur votre ordinateur local](#). Pour obtenir les informations d'identification et autres informations relatives aux paramètres, les exemples utilisent un profil nommé `custom-labels-access`. Par exemple :

```
session = boto3.Session(profile_name='custom-labels-access')
rekognition_client = session.client("rekognition")
```

L'utilisateur représenté par le profil doit être autorisé à appeler les opérations du kit SDK Étiquettes personnalisées Amazon Rekognition et les opérations du kit AWS SDK requises par les exemples. Pour plus d'informations, consultez [Configuration des autorisations du kit SDK](#). Pour attribuer des autorisations, consultez [Configuration des autorisations du kit SDK](#).

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

- 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 à 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 Python, 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 [Basculer 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](#).

Exécution de code dans des environnements AWS

Vous ne devez pas utiliser les informations d'identification utilisateur pour signer les appels du SDK AWS dans des environnements AWS tels que le code de production exécuté dans une fonction AWS Lambda. 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 :

- **Fonction AWS Lambda** : 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](#).

Configuration des autorisations du kit SDK

Pour utiliser les opérations du kit SDK Étiquettes personnalisées Amazon Rekognition, vous devez disposer d'autorisations pour accéder à l'API Étiquettes personnalisées Amazon Rekognition et au compartiment Amazon S3 utilisé pour l'entraînement des modèles.

Rubriques

- [Octroi de droits d'utilisation du kit SDK](#)
- [Mises à jour des stratégies d'utilisation du kit AWS SDK](#)
- [Attribution d'autorisations](#)

Octroi de droits d'utilisation du kit SDK

Nous vous recommandons d'accorder uniquement les autorisations nécessaires à l'exécution d'une tâche (autorisations de moindre privilège). Par exemple, pour appeler [DetectCustomLabels](#), vous avez besoin d'une autorisation pour effectuer un appel `rekognition:DetectCustomLabels`. Pour connaître les autorisations relatives à une opération, consultez le [Guide de référence des API](#).

Lorsque vous commencez à utiliser une application, il se peut que vous ne connaissiez pas les autorisations spécifiques dont vous avez besoin. C'est pourquoi vous pouvez commencer par des autorisations générales. Les stratégies gérées AWS fournissent les autorisations nécessaires pour vous aider à démarrer. Vous pouvez utiliser la stratégie `AmazonRekognitionCustomLabelsFullAccess` gérée par AWS pour bénéficier d'un accès total à l'API Étiquettes personnalisées Amazon Rekognition. Pour plus d'informations, consultez la [politique gérée par AWS : AmazonRekognitionCustomLabelsFullAccess](#). Lorsque vous connaissez les autorisations dont votre application a besoin, réduisez les autorisations en définissant des politiques gérées par le client qui sont spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez [Politiques gérées par le client](#).

Pour attribuer des autorisations, consultez [Attribution d'autorisations](#).

Mises à jour des stratégies d'utilisation du kit AWS SDK

Pour utiliser le kit AWS SDK avec la dernière version d'Étiquettes personnalisées Amazon Rekognition, vous n'avez plus besoin d'autoriser Étiquettes personnalisées Amazon Rekognition à accéder au compartiment Amazon S3 qui contient vos images d'entraînement et de test. Si vous avez déjà ajouté des autorisations, vous n'avez pas besoin de les supprimer. Si vous le souhaitez, supprimez du compartiment toute stratégie dont le service pour le principal est `rekognition.amazonaws.com`. Par exemple :

```
"Principal": {
  "Service": "rekognition.amazonaws.com"
}
```

Pour plus d'informations, consultez [Utilisation des stratégies de compartiment](#).

Attribution d'autorisations

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. 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.

Appel d'une opération Étiquettes personnalisées Amazon Rekognition

Exécutez le code suivant pour vérifier que vous pouvez appeler l'API Étiquettes personnalisées Amazon Rekognition. Le code répertorie les projets de votre compte AWS dans la région AWS actuelle. Si vous n'avez pas encore créé de projet, la réponse est vide, mais confirme que vous pouvez appeler l'opération `DescribeProjects`.

En général, l'appel d'un exemple de fonction nécessite un client de kit AWS SDK Rekognition et tous les autres paramètres requis. Le client de kit AWS SDK est déclaré dans la fonction principale.

Si le code échoue, vérifiez que l'utilisateur que vous utilisez dispose des autorisations adéquates. Vérifiez également que la région AWS que vous utilisez, car Étiquettes personnalisées Amazon Rekognition n'est pas disponible dans toutes les régions AWS.

Pour appeler une opération Étiquettes personnalisées Amazon Rekognition

1. Si vous ne l'avez pas déjà fait, installez et configurez l'AWS CLI et les kits AWS SDK. Pour plus d'informations, consultez [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez l'exemple de code suivant pour afficher vos projets.

CLI

Utilisez la commande `describe-projects` pour répertorier les projets de votre compte.


```
aws rekognition describe-projects \  
--profile custom-labels-access
```

Python

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
This example shows how to describe your Amazon Rekognition Custom Labels  
projects.  
If you haven't previously created a project in the current AWS Region,  
the response is an empty list, but does confirm that you can call an  
Amazon Rekognition Custom Labels operation.  
"""  
  
from botocore.exceptions import ClientError  
import boto3  
  
def describe_projects(rekognition_client):  
    """  
    Lists information about the projects that are in in your AWS account  
    and in the current AWS Region.  
  
    : param rekognition_client: A Boto3 Rekognition client.  
    """  
    try:  
        response = rekognition_client.describe_projects()  
        for project in response["ProjectDescriptions"]:  
            print("Status: " + project["Status"])  
            print("ARN: " + project["ProjectArn"])  
            print()  
        print("Done!")  
    except ClientError as err:  
        print(f"Couldn't describe projects. \n{err}")  
        raise  
  
def main():  
    """  
    Entrypoint for script.
```

```
"""

session = boto3.Session(profile_name='custom-labels-access')
rekognition_client = session.client("rekognition")

describe_projects(rekognition_client)

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

Java V2

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

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.DatasetMetadata;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class Hello {

    public static final Logger logger = Logger.getLogger(Hello.class.getName());

    public static void describeMyProjects(RekognitionClient rekClient) {

        DescribeProjectsRequest descProjects = null;
```

```
// If a single project name is supplied, build projectNames argument
List<String> projectNames = new ArrayList<String>();

descProjects = DescribeProjectsRequest.builder().build();

// Display useful information for each project.

DescribeProjectsResponse resp =
rekClient.describeProjects(descProjects);

for (ProjectDescription projectDescription : resp.projectDescriptions())
{
    System.out.println("ARN: " + projectDescription.projectArn());
    System.out.println("Status: " +
projectDescription.statusAsString());
    if (projectDescription.hasDatasets()) {
        for (DatasetMetadata datasetDescription :
projectDescription.datasets()) {
            System.out.println("\tdataset Type: " +
datasetDescription.datasetTypeAsString());
            System.out.println("\tdataset ARN: " +
datasetDescription.datasetArn());
            System.out.println("\tdataset Status: " +
datasetDescription.statusAsString());
        }
    }
    System.out.println();
}

}

public static void main(String[] args) {

    try {

        // Get the Rekognition client
        RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
```

```
        .build();

        // Describe projects

        describeMyProjects(rekClient);

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        System.exit(1);
    }
}
}
```

Étape 5 (Facultatif) : Chiffrer les fichiers d'entraînement

Vous pouvez choisir l'une des options suivantes pour chiffrer les fichiers manifestes et les fichiers image Étiquettes personnalisées Amazon Rekognition qui se trouvent dans un compartiment de la console ou un compartiment Amazon S3 externe.

- Utiliser une clé Amazon S3 (SSE-S3).
- Utiliser votre AWS KMS key.

Note

Le [principal IAM](#) qui appelle a besoin d'autorisations pour déchiffrer les fichiers. Pour plus d'informations, consultez [Déchiffrement de fichiers chiffrés avec AWS Key Management Service](#).

Pour obtenir des informations sur le chiffrement d'un compartiment Amazon S3, consultez [Définition du comportement de chiffrement côté serveur par défaut pour les compartiments Amazon S3](#).

Déchiffrement de fichiers chiffrés avec AWS Key Management Service

Si vous utilisez AWS Key Management Service (KMS) pour chiffrer vos fichiers manifestes et vos fichiers image Étiquettes personnalisées Amazon Rekognition, ajoutez le principal IAM qui appelle Étiquettes personnalisées Amazon Rekognition à la stratégie de la clé KMS. Étiquettes personnalisées Amazon Rekognition peut ainsi déchiffrer vos fichiers manifestes et vos fichiers image avant l'entraînement. Pour plus d'informations, consultez la section [Mon compartiment Amazon S3 utilise un chiffrement par défaut avec une clé AWS KMS personnalisée. Comment autoriser des utilisateurs à télécharger depuis et charger vers le compartiment ?](#)

Le principal IAM a besoin des autorisations suivantes sur la clé KMS.

- km : GenerateDataKey
- kms:Decrypt

Pour plus d'informations, consultez [Utilisation du chiffrement côté serveur avec des clés AWS KMS \(SSE-KMS\)](#).

Chiffrement de copies d'images d'entraînement et de test

Pour entraîner votre modèle, Étiquettes personnalisées Amazon Rekognition crée une copie de vos images d'entraînement et de test source. Par défaut, les images copiées sont chiffrées au repos à l'aide d'une clé détenue et gérée par AWS. Vous pouvez également choisir d'utiliser votre propre AWS KMS key. Si vous utilisez votre propre clé KMS, vous devez disposer des autorisations suivantes sur la clé KMS.

- km : CreateGrant
- km : DescribeKey

Vous pouvez éventuellement spécifier la clé KMS lorsque vous entraînez le modèle avec la console ou lorsque vous appelez l'opération `CreateProjectVersion`. La clé KMS que vous utilisez n'a pas besoin d'être identique à celle que vous utilisez pour chiffrer les fichiers manifestes et les fichiers image de votre compartiment Amazon S3. Pour plus d'informations, consultez [Étape 5 \(Facultatif\) : Chiffrer les fichiers d'entraînement](#).

Pour de plus amples informations, veuillez consulter [Concepts d'AWS Key Management Service](#). Vos images source ne sont pas affectées.

Pour plus d'informations sur l'entraînement d'un modèle, consultez [Entraînement d'un modèle Amazon Rekognition Custom Labels Custom Labels](#).

Étape 6 (Facultatif) : Associer des jeux de données précédents à de nouveaux projets

Étiquettes personnalisées Amazon Rekognition gère désormais l'association de jeux de données à des projets. Les jeux de données que vous avez créés (précédemment) sont en lecture seule et doivent être associés à un projet pour pouvoir être utilisés. Lorsque vous ouvrez la page de détails d'un projet à l'aide de la console, nous associons automatiquement à ce dernier les jeux de données ayant entraîné la dernière version du modèle du projet. Un jeu de données n'est associé automatiquement à un projet que si vous utilisez le kit AWS SDK.

Les jeux de données précédents qui ne sont pas associés n'ont jamais été utilisés pour entraîner un modèle ou ont été utilisés pour entraîner une version précédente d'un modèle. La page Jeux de données précédents affiche tous vos jeux de données, associés ou non.

Pour utiliser un jeu de données précédent qui n'est pas associé, vous devez créer un nouveau projet sur la page Jeux de données précédents. Le jeu de données devient le jeu de données d'entraînement du nouveau projet. Vous pouvez également créer un projet pour un jeu de données déjà associé, car les jeux de données précédents peuvent être associés à plusieurs projets.

Pour associer un jeu de données précédent à un nouveau projet

1. Ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
2. Dans le volet de gauche, choisissez Utiliser Custom Labels. La page d'accueil d'Étiquettes personnalisées Amazon Rekognition s'affiche.
3. Dans le volet de navigation de gauche, sélectionnez Jeux de données précédents.
4. Dans la vue Jeux de données, choisissez le jeu de données précédent que vous souhaitez associer à un projet.
5. Choisissez Créer un projet avec un jeu de données.
6. Sur la page Créer un projet, saisissez un nom pour votre nouveau projet sous Nom du projet.
7. Choisissez Créer un projet pour créer le projet. La création du projet peut prendre un certain temps.

8. Utilisez le projet. Pour plus d'informations, consultez [Comprendre les étiquettes personnalisées Amazon Rekognition](#).

Utilisation d'un jeu de données précédent comme jeu de données de test

Vous pouvez utiliser un jeu de données précédent comme jeu de données de test pour un projet existant en associant d'abord le jeu de données précédent à un nouveau projet. Vous copiez ensuite le jeu de données d'entraînement du nouveau projet dans le jeu de données de test du projet existant.

Pour utiliser un jeu de données précédent comme jeu de données de test

1. Suivez les instructions de l'[Étape 6 \(Facultatif\) : Associer des jeux de données précédents à de nouveaux projets](#) pour associer le jeu de données précédent à un nouveau projet.
2. Copiez le jeu de données d'entraînement du nouveau projet dans le jeu de données de test du projet existant. Pour plus d'informations, consultez [Jeu de données](#).
3. Suivez les instructions de [Suppression d'un projet Amazon Rekognition Custom Labels \(console\)](#) pour supprimer le nouveau projet.

Vous pouvez également créer le jeu de données de test en utilisant le fichier manifeste du jeu de données précédent. Pour plus d'informations, consultez [Création d'un fichier manifeste](#).

Comprendre les étiquettes personnalisées Amazon Rekognition

Cette section donne une vue d'ensemble du flux de travail pour former et utiliser un modèle d'étiquettes personnalisées Amazon Rekognition avec la console et leAWS SDK.

Note

Amazon Rekognition Custom Labels gère désormais les ensembles de données au sein d'un projet. Vous pouvez créer des ensembles de données pour vos projets à l'aide de la console et duAWS SDK. Si vous avez déjà utilisé les étiquettes personnalisées Amazon Rekognition, vos anciens ensembles de données devront peut-être être associés à un nouveau projet. Pour plus d'informations, voir [Étape 6 \(Facultatif\) : Associer des jeux de données précédents à de nouveaux projets](#)

Rubriques

- [Choisissez votre type de modèle](#)
- [Création d'un modèle](#)
- [Améliorez votre modèle](#)
- [Démarrez votre modèle.](#)
- [Analyser une image](#)
- [Arrêter votre modèle](#)

Choisissez votre type de modèle

Vous décidez d'abord du type de modèle que vous souhaitez former, en fonction de vos objectifs commerciaux. Par exemple, vous pouvez entraîner un mannequin à trouver votre logo sur les réseaux sociaux, à identifier vos produits dans les rayons des magasins ou à classer les pièces de machines sur une chaîne de montage.

Les étiquettes personnalisées Amazon Rekognition peuvent entraîner les types de modèles suivants :

- [Trouvez des objets, des scènes et des concepts](#)

- [Trouver l'emplacement des objets](#)
- [Recherchez l'emplacement des marques](#)

Pour vous aider à choisir le type de modèle à entraîner, Amazon Rekognition Custom Labels propose des exemples de projets que vous pouvez utiliser. Pour plus d'informations, veuillez consulter [Démarrez avec les étiquettes personnalisées Amazon Rekognition](#).

Trouvez des objets, des scènes et des concepts

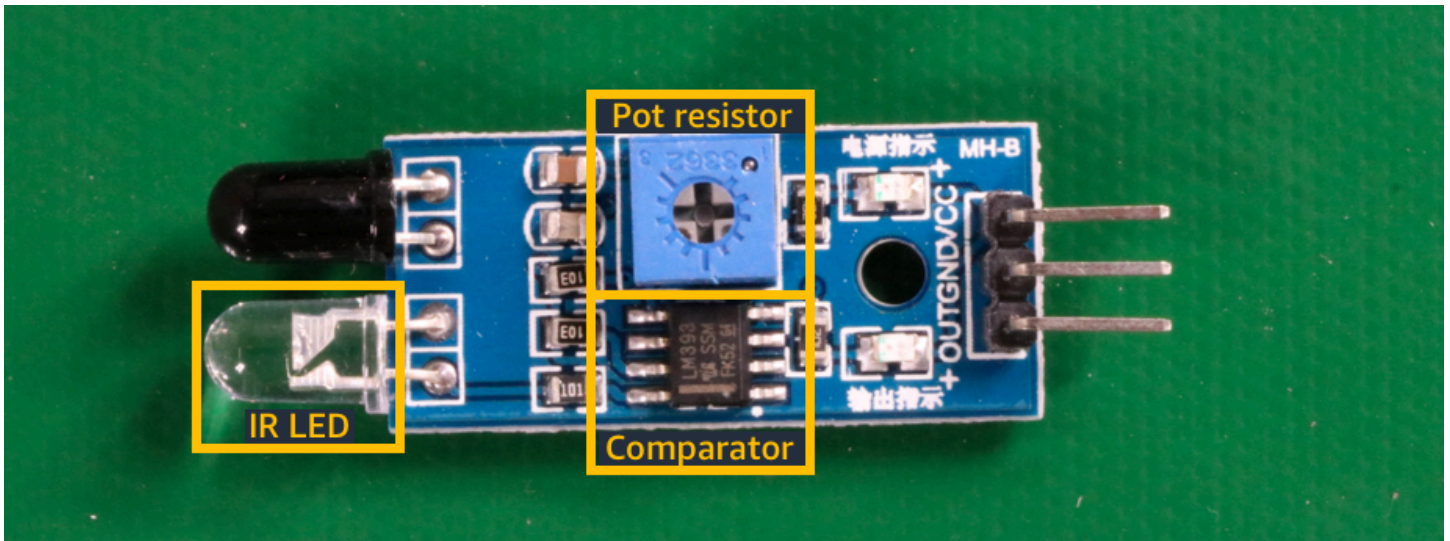
Le modèle prédit les classifications des objets, des scènes et des concepts associés à une image complète. Par exemple, vous pouvez entraîner un modèle qui détermine si une image contient une attraction touristique ou non. Pour un exemple de projet, voir [Image classification](#).



Vous pouvez également entraîner un modèle qui classe les images en plusieurs catégories. Par exemple, l'image précédente peut comporter des catégories telles que la couleur du ciel, le reflet ou le lac. Pour un exemple de projet, voir [Classification d'images à étiquettes multiples](#).

Trouver l'emplacement des objets

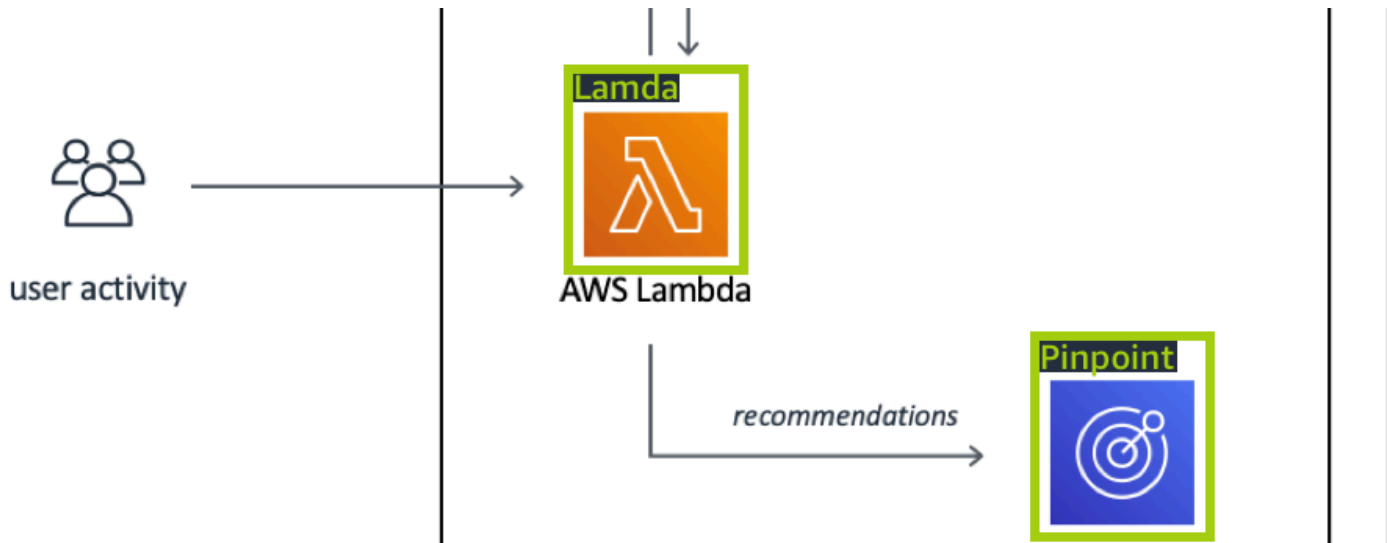
Le modèle prédit la position d'un objet sur une image. La prédiction inclut des informations sur le cadre de délimitation concernant l'emplacement de l'objet et une étiquette qui identifie l'objet dans le cadre de délimitation. Par exemple, l'image suivante montre des boîtes de délimitation situées autour de différentes parties d'une carte de circuit, telles qu'un comparateur ou une résistance en pot.



L'[Localisation d'objets](#) exemple de projet montre comment Amazon Rekognition Custom Labels utilise des zones de délimitation étiquetées pour former un modèle qui trouve l'emplacement des objets.

Recherchez l'emplacement des marques

Les étiquettes personnalisées Amazon Rekognition peuvent entraîner un modèle qui trouve l'emplacement des marques, telles que des logos, sur une image. La prédiction inclut des informations relatives au cadre de délimitation concernant l'emplacement de la marque et une étiquette identifiant l'objet à l'intérieur du cadre de délimitation. Pour un exemple de projet, voir [Détection de marque](#).



Création d'un modèle

Pour créer un modèle, vous devez créer un projet, créer des ensembles de données d'entraînement et de test et entraîner le modèle.

Créer un projet

Un projet Amazon Rekognition Custom Labels est un regroupement des ressources nécessaires à la création et à la gestion d'un modèle. Un projet gère les éléments suivants :

- Ensembles de données : images et étiquettes d'images utilisées pour entraîner un modèle. Un projet possède un jeu de données d'entraînement et un jeu de données de test.
- Modèles : les logiciels que vous formez pour trouver les concepts, les scènes et les objets propres à votre entreprise. Vous pouvez avoir plusieurs versions d'un modèle dans un projet.

Nous vous recommandons d'utiliser un projet pour un seul cas d'utilisation, par exemple pour rechercher des pièces de circuit imprimé sur une carte de circuit imprimé.

Vous pouvez créer un projet à l'aide de la console Amazon Rekognition Custom Labels et de l'[CreateProjectAPI](#). Pour plus d'informations, veuillez consulter [Crée un projet](#).

Création de jeux de données de formation et de test

Un jeu de données est un ensemble d'images et d'étiquettes qui décrivent ces images. Dans votre projet, vous créez un jeu de données d'entraînement et un autre de test qu'Amazon Rekognition Custom Labels utilise afin d'entraîner et de tester votre modèle.

Une étiquette identifie un objet, une scène, un concept ou un cadre de délimitation autour d'un objet dans une image. Les étiquettes sont attribuées à une image entière (au niveau de l'image) ou à un cadre de délimitation qui entoure un objet sur une image.

Important

La façon dont vous étiquetez les images dans vos ensembles de données détermine le type de modèle créé par Amazon Rekognition Custom Labels. Par exemple, pour entraîner un modèle qui trouve des objets, des scènes et des concepts, vous attribuez des étiquettes de niveau d'image aux images de vos ensembles de données d'entraînement et de test. Pour plus d'informations, veuillez consulter [Utilisation des jeux de données](#).

Les images doivent être au format PNG et JPEG, et vous devez suivre les recommandations relatives aux images d'entrée. Pour plus d'informations, veuillez consulter [Préparation des images](#).

Création d'ensembles de données d'entraînement et de test (console)

Vous pouvez démarrer un projet avec un seul jeu de données ou avec des ensembles de données d'entraînement et de test distincts. Si vous commencez avec un seul jeu de données, Amazon Rekognition Custom Labels divise votre jeu de données pendant l'entraînement afin de créer un jeu de données d'apprentissage (80 %) et un jeu de données de test (20 %) pour votre projet. Commencez avec un seul ensemble de données si vous souhaitez qu'Amazon Rekognition Custom Labels décide quelles images seront utilisées pour la formation et les tests. Pour un contrôle total de la formation, des tests et du réglage des performances, nous vous recommandons de démarrer votre projet avec des ensembles de données d'entraînement et de test distincts.

Pour créer les jeux de données pour un projet, vous devez importer les images de l'une des manières suivantes :

- Importez des images depuis votre ordinateur local.

- Importer des images à partir d'un compartiment S3. Les étiquettes personnalisées Amazon Rekognition peuvent étiqueter les images en utilisant les noms des dossiers qui contiennent les images.
- Importez un fichier manifeste Amazon SageMaker Ground Truth.
- Copier un jeu de données Amazon Rekognition Custom Labels existant.

Pour plus d'informations, veuillez consulter [Création de jeux de données d'entraînement et de test avec des images](#).

Selon l'endroit d'où vous importez vos images, celles-ci peuvent ne pas être étiquetées. Par exemple, les images importées depuis un ordinateur local ne sont pas étiquetées. Les images importées depuis un fichier manifeste Amazon SageMaker Ground Truth sont étiquetées. Vous pouvez utiliser la console Amazon Rekognition Custom Labels pour ajouter, modifier et attribuer des étiquettes. Pour plus d'informations, veuillez consulter [Étiquetage des images](#).

Pour créer vos ensembles de données d'entraînement et de test à l'aide de la console, consultez [Création de jeux de données d'entraînement et de test avec des images](#). Pour consulter un didacticiel qui inclut la création de jeux de données d'entraînement et de test, consultez [Tutoriel : Classification d'images](#).

Création d'ensembles de données de formation et de test (SDK)

Pour créer vos ensembles de données d'entraînement et de test, vous utilisez l'CreateDatasetAPI. Vous pouvez créer un jeu de données en utilisant un fichier manifeste au format Amazon Sagemaker ou en copiant un jeu de données Amazon Rekognition Custom Labels existant. Pour plus d'informations, voir [Création de jeux de données d'entraînement et de test \(kit SDK\)](#). Si nécessaire, vous pouvez créer votre propre fichier manifeste. Pour plus d'informations, veuillez consulter [the section called "Création d'un fichier manifeste"](#).

Entraînez votre modèle

Entraînez votre modèle à l'aide du jeu de données d'entraînement. Une nouvelle version d'un modèle est créée chaque fois qu'il est entraîné. Pendant la formation, les étiquettes personnalisées Amazon Rekognition testent les performances de votre modèle entraîné. Vous pouvez utiliser les résultats pour évaluer et améliorer votre modèle. La formation prend un certain temps. Seule une formation de modèle réussie vous est facturée. Pour plus d'informations, veuillez consulter [Entraînement d'un modèle Amazon Rekognition Custom Labels Custom Labels](#). Si la formation du modèle échoue,

Amazon Rekognition Custom Labels fournit des informations de débogage que vous pouvez utiliser. Pour plus d'informations, veuillez consulter [Déboguer un modèle d'entraînement ayant échoué](#).

Entraînez votre modèle (console)

Pour entraîner votre modèle à l'aide de la console, voir [Entraînement d'un modèle \(console\)](#).

Entraînement d'un modèle (SDK)

Vous entraînez un modèle d'étiquettes personnalisées Amazon Rekognition en appelant [CreateProjectVersion](#). Pour plus d'informations, veuillez consulter [Entraînement d'un modèle \(SDK\)](#).

Améliorez votre modèle

Pendant les tests, Amazon Rekognition Custom Labels crée des mesures d'évaluation que vous pouvez utiliser pour améliorer votre modèle entraîné.

Évaluer votre modèle

Évaluez les performances de votre modèle à l'aide des mesures de performance créées lors des tests. Les indicateurs de performance, tels que F1, précision et rappel, vous permettent de comprendre les performances de votre modèle entraîné et de décider si vous êtes prêt à l'utiliser en production. Pour plus d'informations, veuillez consulter [Métriques pour évaluer votre modèle](#).

Évaluer un modèle (console)

Pour consulter les métriques de performances, consultez [Accès aux mesures d'évaluation \(console\)](#).

Évaluer un modèle (SDK)

Pour obtenir des mesures de performance, vous appelez [DescribeProjectVersions](#) pour obtenir les résultats des tests. Pour plus d'informations, veuillez consulter [Accès aux métriques d'évaluation \(SDK\) d'Amazon Rekognition Custom Labels](#). Les résultats des tests incluent des mesures non disponibles dans la console, telles qu'une matrice de confusion pour les résultats de classification. Les résultats des tests sont renvoyés dans les formats suivants :

- Score F1 : valeur unique représentant les performances globales de précision et de rappel du modèle. Pour plus d'informations, veuillez consulter [F1](#).
- Emplacement du fichier de synthèse : le résumé des tests inclut des mesures d'évaluation agrégées pour l'ensemble de données de test et des mesures pour chaque étiquette individuelle.

DescribeProjectVersions renvoie le compartiment S3 et l'emplacement du dossier du fichier récapitulatif. Pour plus d'informations, veuillez consulter [Fichier récapitulatif](#).

- Emplacement de l'instantané du manifeste d'évaluation : l'instantané contient des détails sur les résultats des tests, notamment les cotes de confiance et les résultats des tests de classification binaire, tels que les faux positifs. DescribeProjectVersions renvoie le compartiment S3 et l'emplacement du dossier des fichiers de capture instantanée. Pour plus d'informations, veuillez consulter [Aperçu du manifeste d'évaluation](#).

Améliorez votre modèle

Si des améliorations sont nécessaires, vous pouvez ajouter d'autres images d'apprentissage ou améliorer l'étiquetage des jeux de données. Pour plus d'informations, veuillez consulter [Amélioration d'un modèle Amazon Rekognition Custom Labels](#). Vous pouvez également donner votre avis sur les prévisions faites par votre modèle et les utiliser pour apporter des améliorations à votre modèle. Pour plus d'informations, veuillez consulter [Solution de feedback sur les modèles](#).

Améliorez votre modèle (console)

Pour ajouter des images à un jeu de données, consultez [Ajouter d'autres images à un jeu de données](#). Pour ajouter ou modifier des étiquettes, voir [the section called "Étiquetage des images"](#).

Pour réentraîner votre modèle, voir [Entraînement d'un modèle \(console\)](#).

Améliorez votre modèle (SDK)

Pour ajouter des images à un jeu de données ou modifier l'étiquetage d'une image, utilisez l'UpdateDatasetEntriesAPI. UpdateDatasetEntries met à jour ou ajoute des lignes JSON à un fichier manifeste. Chaque ligne JSON contient des informations relatives à une seule image, telles que les étiquettes attribuées ou les informations relatives au cadre de délimitation. Pour plus d'informations, veuillez consulter [Ajouter d'autres images \(SDK\)](#). Pour afficher les entrées d'un jeu de données, utilisez l>ListDatasetEntriesAPI.

Pour réentraîner votre modèle, voir [Entraînement d'un modèle \(SDK\)](#).

Démarrez votre modèle.

Avant de pouvoir utiliser votre modèle, vous devez le démarrer à l'aide de la console Amazon Rekognition Custom Labels ou de l'StartProjectVersionAPI. La durée de fonctionnement de

vos modèle vous est facturée. Pour plus d'informations, veuillez consulter [Exécution d'un modèle entraîné](#).

Démarrez votre modèle (console).

Pour démarrer votre modèle à l'aide de la console, consultez [Démarrage d'un modèle Étiquettes personnalisées Amazon Rekognition \(Console\)](#).

Démarrez votre modèle.

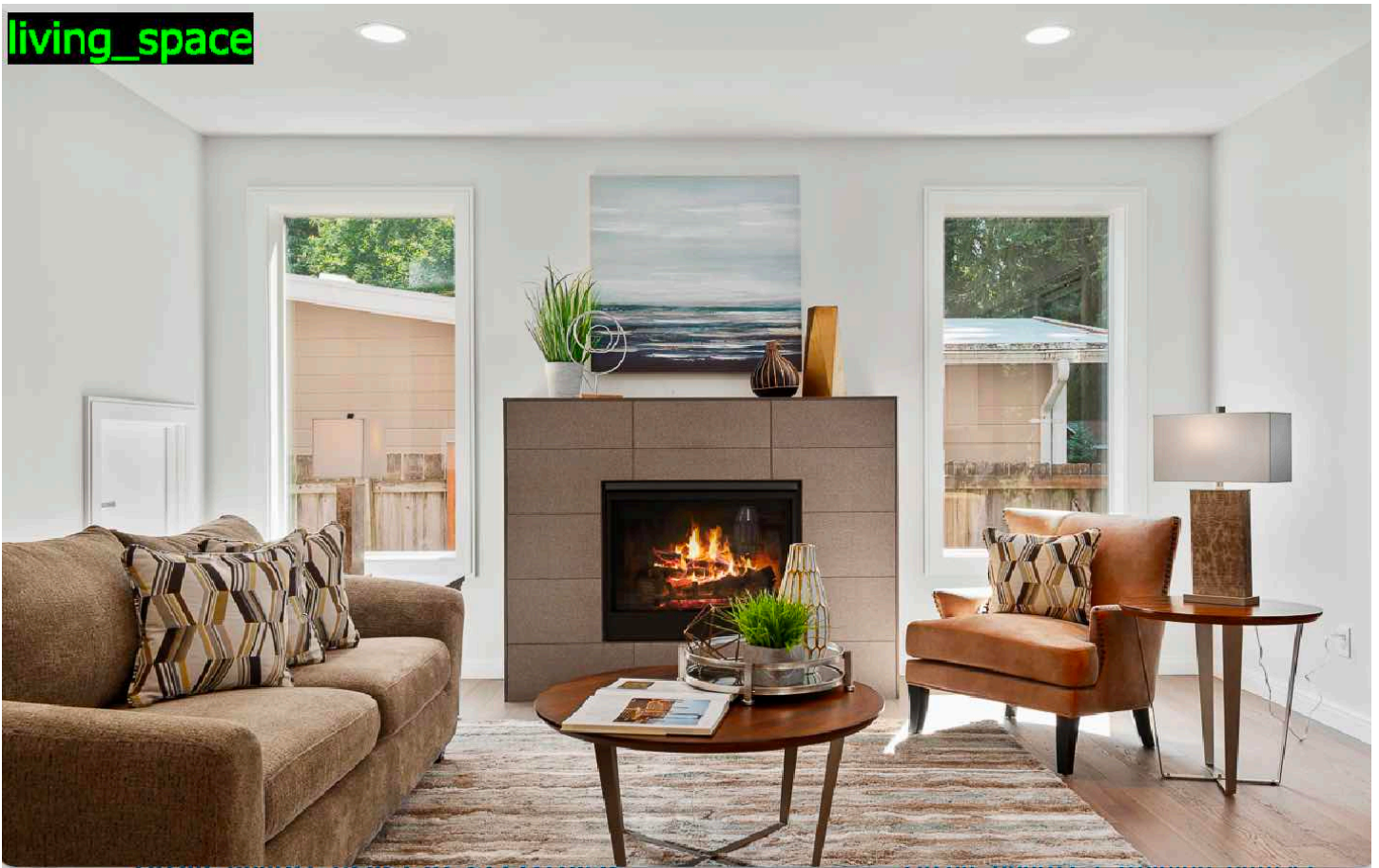
Vous commencez à appeler votre modèle [StartProjectVersion](#). Pour plus d'informations, veuillez consulter [Démarrage d'un modèle Étiquettes personnalisées Amazon Rekognition \(kit SDK\)](#).

Analyser une image

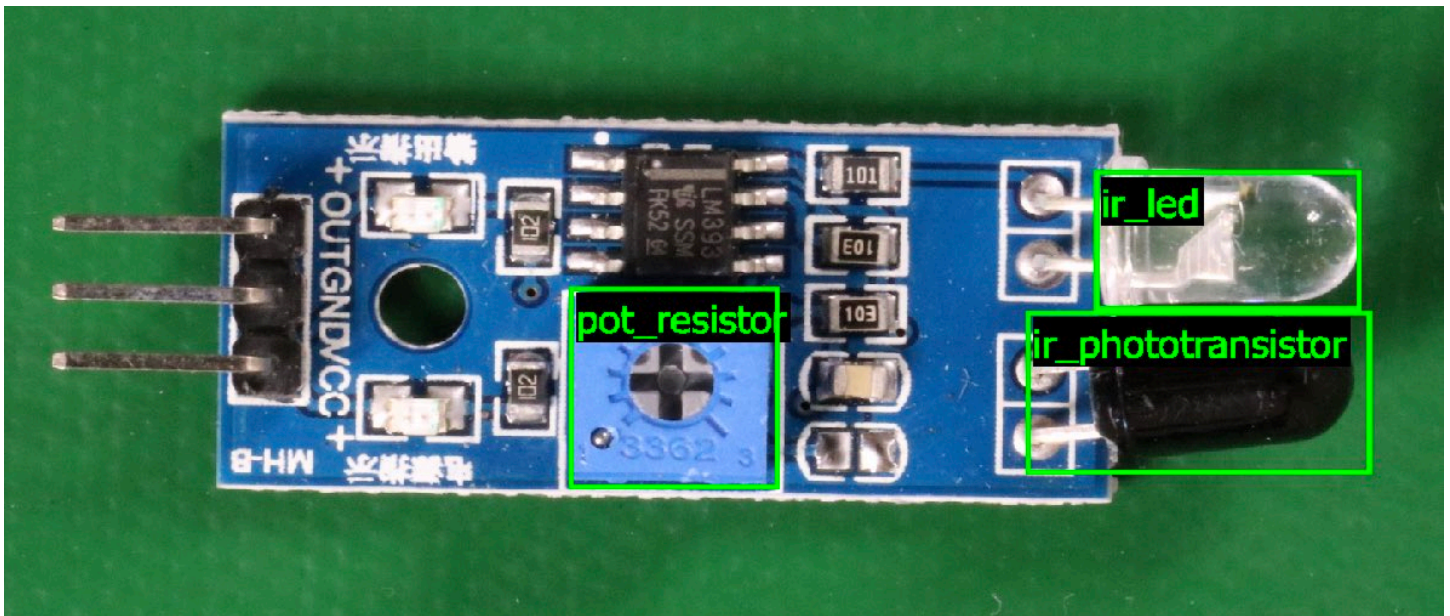
Pour analyser une image avec votre modèle, vous utilisez l'`DetectCustomLabelsAPI`. Vous pouvez spécifier une image locale ou une image stockée dans un compartiment S3. L'opération nécessite également l'Amazon Resource Name (ARN) du modèle que vous souhaitez utiliser.

Si votre modèle trouve des objets, des scènes et des concepts, la réponse inclut une liste des étiquettes au niveau de l'image présentes dans l'image. Par exemple, l'image suivante affiche les étiquettes au niveau de l'image trouvées à l'aide du projet d'exemple Rooms.

living_space



Si le modèle trouve l'emplacement des objets, la réponse inclut la liste des zones de délimitation étiquetées présentes dans l'image. Un cadre de délimitation représente l'emplacement d'un objet sur une image. Vous pouvez utiliser les informations du cadre de délimitation pour dessiner un cadre de délimitation autour d'un objet. Par exemple, l'image suivante montre des boîtes de délimitation autour de pièces de circuits imprimés trouvées à l'aide du projet d'exemple de cartes de circuits imprimés.



Pour plus d'informations, veuillez consulter [Analyse d'une image avec un modèle entraîné](#).

Arrêter votre modèle

Vous êtes facturé pour la durée pendant laquelle votre modèle fonctionne. Si vous n'utilisez plus votre modèle, arrêtez-le à l'aide de la console Amazon Rekognition Custom Labels ou de l'API `StopProjectVersion`. Pour plus d'informations, veuillez consulter [Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition](#).

Arrêtez votre modèle (console)

Pour arrêter un modèle en cours d'exécution à l'aide de la console, voir [Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition \(console\)](#).

Arrêtez votre modèle (SDK)

Pour arrêter un modèle en cours d'exécution, appelez `StopProjectVersion`. Pour plus d'informations, veuillez consulter [Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition \(kit SDK\)](#).

Démarrez avec les étiquettes personnalisées Amazon Rekognition

Avant de les démarrer, nous vous recommandons de lire [Comprendre les étiquettes personnalisées Amazon Rekognition](#).

Vous utilisez les étiquettes personnalisées Amazon Rekognition pour former un modèle de machine learning. Le modèle entraîné analyse les images pour trouver les objets, les scènes et les concepts propres aux besoins de votre entreprise. Par exemple, vous pouvez entraîner un modèle à classer des images de maisons ou à trouver l'emplacement de composants électroniques sur un circuit imprimé.

Pour vous aider à démarrer, Amazon Rekognition Custom Labels inclut des tutoriels vidéo et des exemples de projets.

Note

Pour plus d'informations sur les régions et les points de terminaison pris en charge par Amazon Rekognition Custom Labels, voir [Critères de reconnaissance et quotas](#).

Vidéos tutorielles

Les vidéos vous montrent comment utiliser les étiquettes personnalisées Amazon Rekognition pour vous entraîner et utiliser un modèle.

Pour visionner les vidéos du didacticiel

1. Connectez-vous à l'AWS Management Console et ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
2. Dans le volet de gauche, choisissez Utiliser des étiquettes personnalisées. La page d'accueil d'Amazon Rekognition Custom Labels s'affiche. Si vous ne voyez pas Utiliser des étiquettes personnalisées, vérifiez que la région AWS que vous utilisez prend en charge les étiquettes personnalisées Amazon Rekognition.
3. Dans le volet de navigation, choisissez Commencez.
4. Dans Qu'est-ce qu'Amazon Rekognition Custom Labels ?, choisissez la vidéo pour regarder la vidéo de présentation.

5. Dans le volet de navigation, choisissez Des tutoriels.
6. Sur le Des tutoriels page, choisissez les tutoriels vidéo que vous souhaitez regarder.

Exemples de projets

Amazon Rekognition Custom Labels fournit les exemples de projets suivants.

Image classification

Le projet de classification d'images (Rooms) entraîne un modèle qui trouve un ou plusieurs emplacements domestiques dans une image, par exemple cour arrière, cuisine, et patio. Les images d'entraînement et de test représentent un seul emplacement. Chaque image est étiquetée avec une seule étiquette au niveau de l'image, telle que cuisine, patio, ou espace de vie. Pour une image analysée, le modèle entraîné renvoie une ou plusieurs étiquettes correspondantes parmi l'ensemble d'étiquettes au niveau de l'image utilisé pour l'apprentissage. Par exemple, le modèle peut trouver l'étiquette espace de vie dans l'image suivante. Pour plus d'informations, veuillez consulter [Recherche d'objets, de scènes et de concepts](#).



Classification d'images à étiquettes multiples

Le projet de classification d'images à étiquettes multiples (Flowers) entraîne un modèle qui classe les images de fleurs en trois concepts (type de fleur, présence de feuilles et stade de croissance).

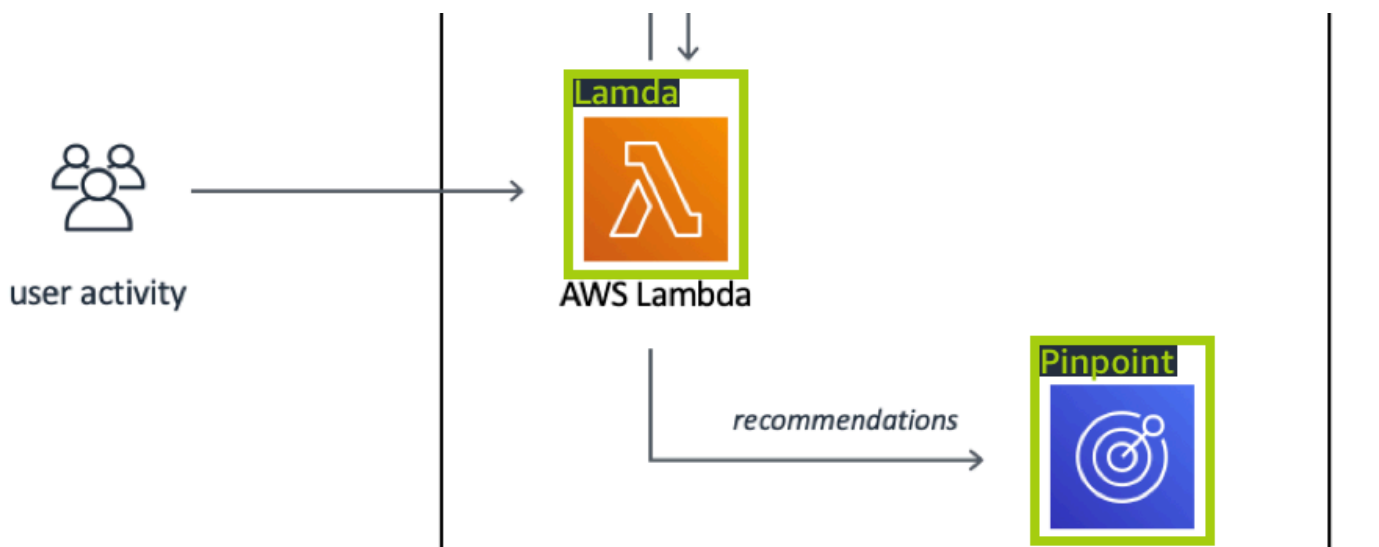
Les images de formation et de test comportent des étiquettes au niveau de l'image pour chaque concept, telles que camélia pour un type de fleur, avec_feuilles pour une fleur à feuilles, et entier_adulte pour une fleur parvenue à maturité.

Pour une image analysée, le modèle entraîné renvoie des étiquettes correspondantes à partir de l'ensemble d'étiquettes au niveau de l'image utilisé pour l'apprentissage. Par exemple, le modèle renvoie les étiquettes euphorbe méditerranéenne et avec feuilles pour l'image suivante. Pour plus d'informations, veuillez consulter [Recherche d'objets, de scènes et de concepts](#).



Détection de marque

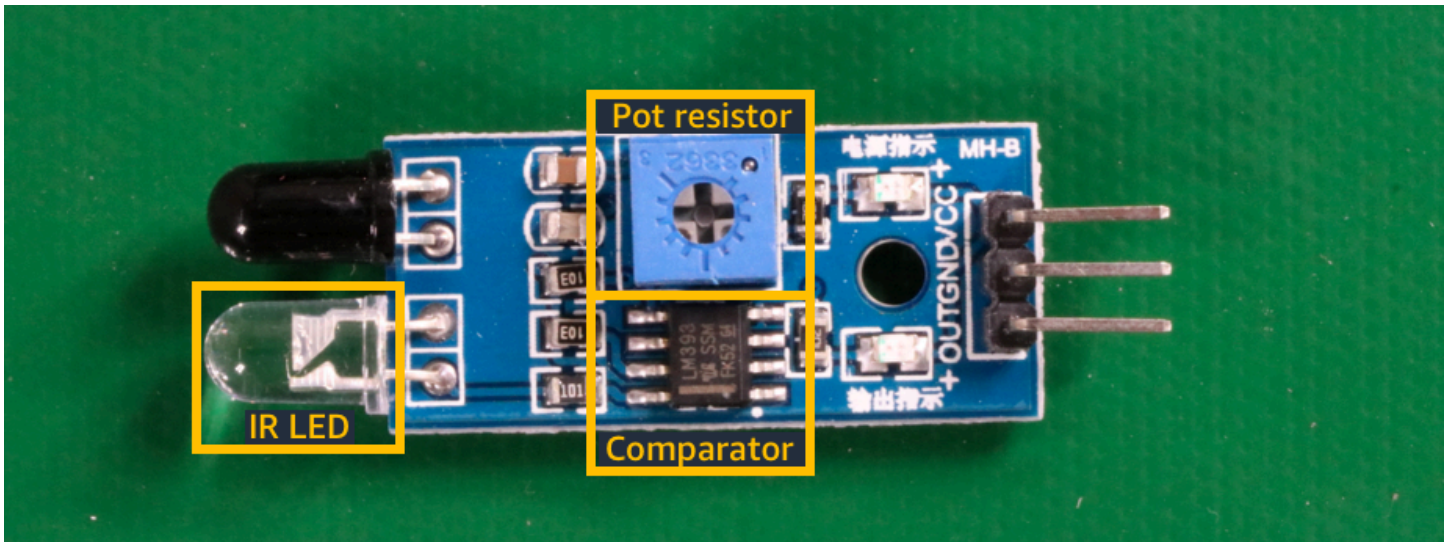
Le projet de détection de marques (Logos) entraîne un modèle qui trouve l'emplacement de certains AWS des logos tels que Extrait Amazon, et AWS Lambda. Les images de formation ne concernent que le logo et comportent une seule étiquette de niveau d'image, telle que lambda ou extrait. Il est également possible d'entraîner un modèle de détection de marque à l'aide d'images d'entraînement comportant des cadres de délimitation des emplacements des marques. Les images de test comportent des cadres de délimitation étiquetés qui représentent l'emplacement des logos dans des emplacements naturels, tels qu'un schéma architectural. Le modèle entraîné trouve les logos et renvoie un cadre de sélection étiqueté pour chaque logo trouvé. Pour plus d'informations, veuillez consulter [Recherche de l'emplacement d'une marque](#).



Localisation d'objets

Le projet de localisation d'objets (cartes de circuits imprimés) entraîne un modèle qui trouve l'emplacement des pièces sur une carte de circuit imprimé, par exemple comparateur ou

undiode électroluminescente infrarouge. Les images d'entraînement et de test incluent des boîtiers de délimitation qui entourent les parties du circuit imprimé et une étiquette identifiant la pièce à l'intérieur du boîtier de délimitation. Les noms des étiquettes sont phototransistor infrarouge, ir_led, pot_resistance, et comparateur. Le modèle entraîné trouve les pièces du circuit imprimé et renvoie une limite étiquetée pour chaque pièce de circuit trouvée. Pour plus d'informations, veuillez consulter [Recherche de l'emplacement d'objets](#).



À l'aide des exemples de projets

Ces instructions de mise en route vous montrent comment entraîner un modèle à l'aide d'exemples de projets créés pour vous par Amazon Rekognition Custom Labels. Il explique également comment démarrer le modèle et l'utiliser pour analyser une image.

Création de l'exemple de projet

Pour commencer, choisissez le projet à utiliser. Pour plus d'informations, veuillez consulter [Étape 1 : Choisissez un exemple de projet](#).

Amazon Rekognition Custom Labels utilise des ensembles de données pour entraîner et évaluer (tester) un modèle. Un jeu de données gère les images et les étiquettes qui identifient le contenu des images. Les exemples de projets incluent un jeu de données d'entraînement et un jeu de données de test dans lequel toutes les images sont étiquetées. Vous n'avez pas besoin d'apporter de modifications avant d'entraîner votre modèle. Les exemples de projets montrent les deux manières dont Amazon Rekognition Custom Labels utilise les étiquettes pour former différents types de modèles.

- au niveau de l'image— L'étiquette identifie un objet, une scène ou un concept qui représente l'ensemble de l'image.
- boîte de délimitation— L'étiquette identifie le contenu d'un cadre de délimitation. Un cadre de délimitation est un ensemble de coordonnées d'image qui entourent un objet dans une image.

Plus tard, lorsque vous créez un projet avec vos propres images, vous devrez créer des ensembles de données d'entraînement et de test, et également étiqueter vos images. Pour plus d'informations, veuillez consulter [Choisissez votre type de modèle](#).

Entraînement du modèle

Une fois qu'Amazon Rekognition Custom Labels a créé l'exemple de projet, vous pouvez entraîner le modèle. Pour plus d'informations, veuillez consulter [Étape 2 : Entraînez votre modèle](#). Une fois l'entraînement terminé, vous évaluez normalement les performances du modèle. Les images de l'exemple de jeu de données créent déjà un modèle hautes performances, et vous n'avez pas besoin d'évaluer le modèle avant de l'exécuter. Pour plus d'informations, veuillez consulter [Amélioration d'un modèle d'étiquettes personnalisées Amazon Rekognition certifié](#).

Utilisation du modèle

Ensuite, vous lancez le modèle. Pour plus d'informations, veuillez consulter [Étape 3 : Démarrez votre modèle](#).

Une fois que vous avez commencé à exécuter votre modèle, vous pouvez l'utiliser pour analyser de nouvelles images. Pour plus d'informations, veuillez consulter [Étape 4 : Analyser une image avec votre modèle](#).

Vous êtes facturé en fonction de la durée de fonctionnement de votre modèle. Lorsque vous avez fini d'utiliser le modèle d'exemple, vous devez arrêter le modèle. Pour plus d'informations, veuillez consulter [Étape 5 : arrêtez votre modèle](#).

Étapes suivantes

Lorsque vous êtes prêt, vous pouvez créer vos propres projets. Pour plus d'informations, veuillez consulter [Étape 6 : étapes suivantes](#).

Étape 1 : Choisissez un exemple de projet

Au cours de cette étape, vous allez choisir un exemple de projet. Amazon Rekognition Custom Labels crée ensuite un projet et un ensemble de données pour vous. Un projet gère les fichiers utilisés pour entraîner votre modèle. Pour plus d'informations, veuillez consulter [Gestion d'un projet Amazon Rekognition Custom Labels](#). Les jeux de données contiennent les images, les étiquettes attribuées et les cadres de délimitation que vous utilisez pour entraîner et tester un modèle. Pour plus d'informations, veuillez consulter [the section called "Gestion des jeux de données"](#).

Pour plus d'informations sur les exemples de projets, voir [Exemples de projets](#).

Choisissez un exemple de projet

1. Connectez-vous à l'AWS Management Console et ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
2. Dans le volet de gauche, choisissez Utiliser des étiquettes personnalisées. La page d'accueil d'Amazon Rekognition Custom Labels s'affiche. Si vous ne voyez pas Utiliser des étiquettes personnalisées, vérifiez que le [AWS Région](#) que vous utilisez prend en charge les étiquettes personnalisées Amazon Rekognition.
3. Sélectionnez Get started (Démarrer).

Amazon Rekognition Custom Labels



▼ Get started

Tutorials


Example projects

Projects

Datasets

4. Dans Découvrez des exemples de projets, choisissez Essayez des exemples de projets.

5. Décidez du projet que vous souhaitez utiliser et choisissez **Créer un projet »nom du projet«** dans la section des exemples. Amazon Rekognition Custom Labels crée ensuite l'exemple de projet pour vous.

 Note

Si c'est la première fois que vous ouvrez la console dans la version actuelle AWS Région, la Première configuration une boîte de dialogue s'affiche. Procédez comme suit :

1. Notez le nom du compartiment Amazon S3 qui s'affiche.
2. Choisissez Poursuivre pour permettre à Amazon Rekognition Custom Labels de créer un compartiment Amazon S3 (compartiment console) en votre nom.

Image Classification

Recommended for content categorization



Classify images as belonging to a set of predefined labels. For example, real estate companies can use Amazon Rekognition Custom Labels to categorize their images of living rooms, backyards, bedrooms, and other household locations.

Create project "Rooms"

Multi-label classification

Recommended for inventory management

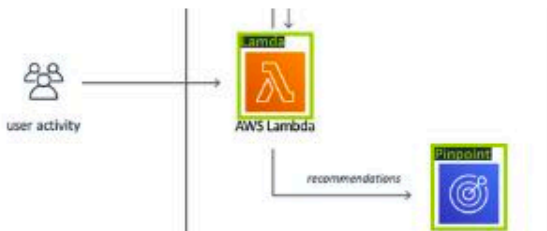


Classify images into multiple categories, such as the color, size, texture, and type of a flower. For example, plant growers can use Amazon Rekognition Custom Labels to distinguish between different types of flowers and if they are healthy, damaged, or infected.

Create project "Flowers"

Brand detection

Recommended for retail, media networks, and advertising

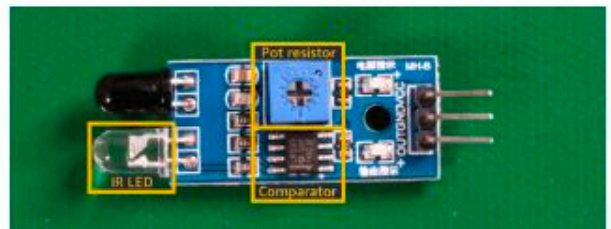


Use brand detection to find the location of commercial brands in images. For example, to report on advertiser coverage, media networks can use Amazon Rekognition Custom Labels to report on the location of sponsor logos in photographs.

Create project "Logos"

Object localization

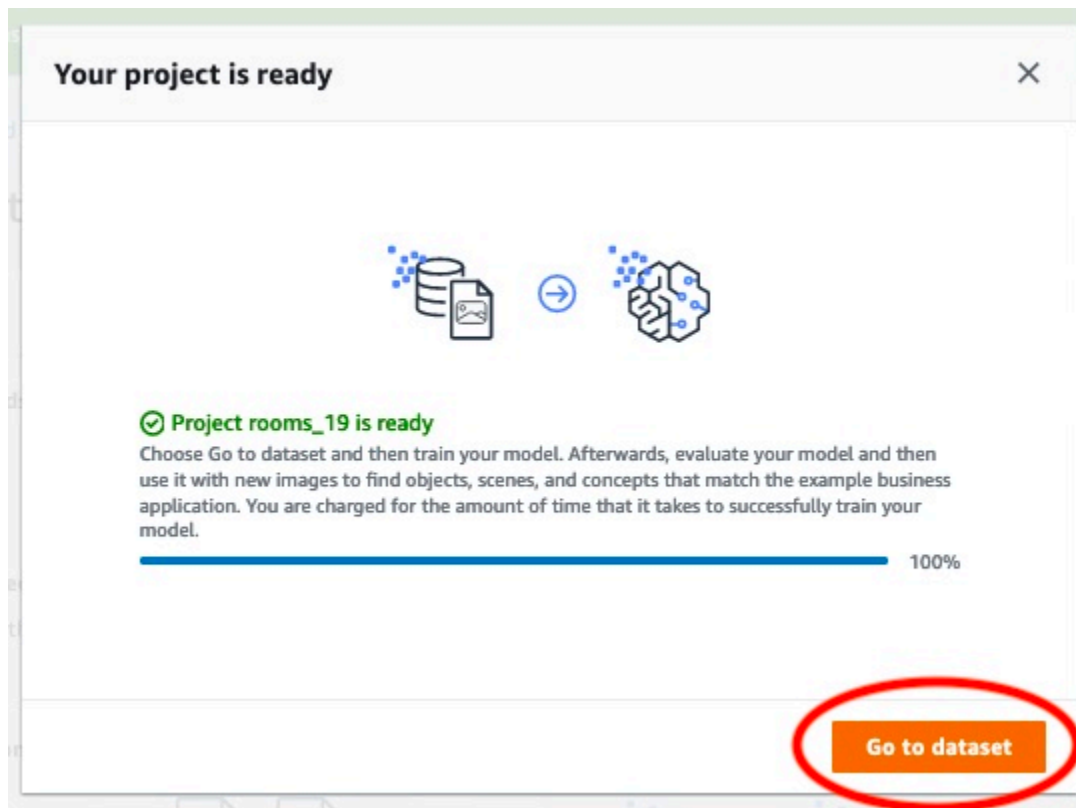
Recommended for manufacturing and production chains



Use object localization to locate parts used in production or manufacturing lines. For example, in the electronics industry, Amazon Rekognition Custom Labels can help count the number of capacitors on a circuit board.

Create project "Circuit boards"

6. Lorsque votre projet est prêt, choisissez Accéder au jeu de données.

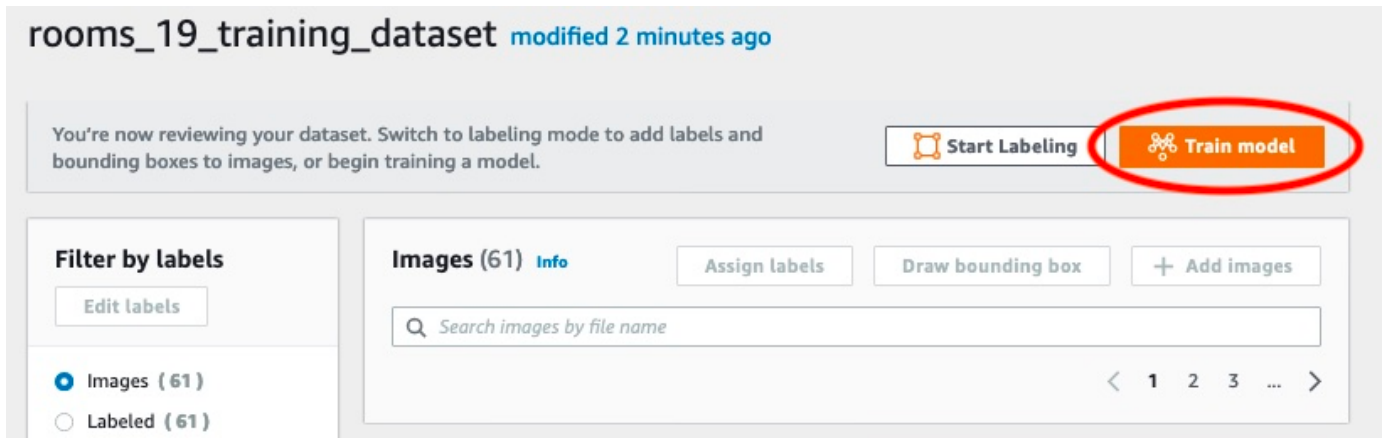


Étape 2 : Entraînez votre modèle

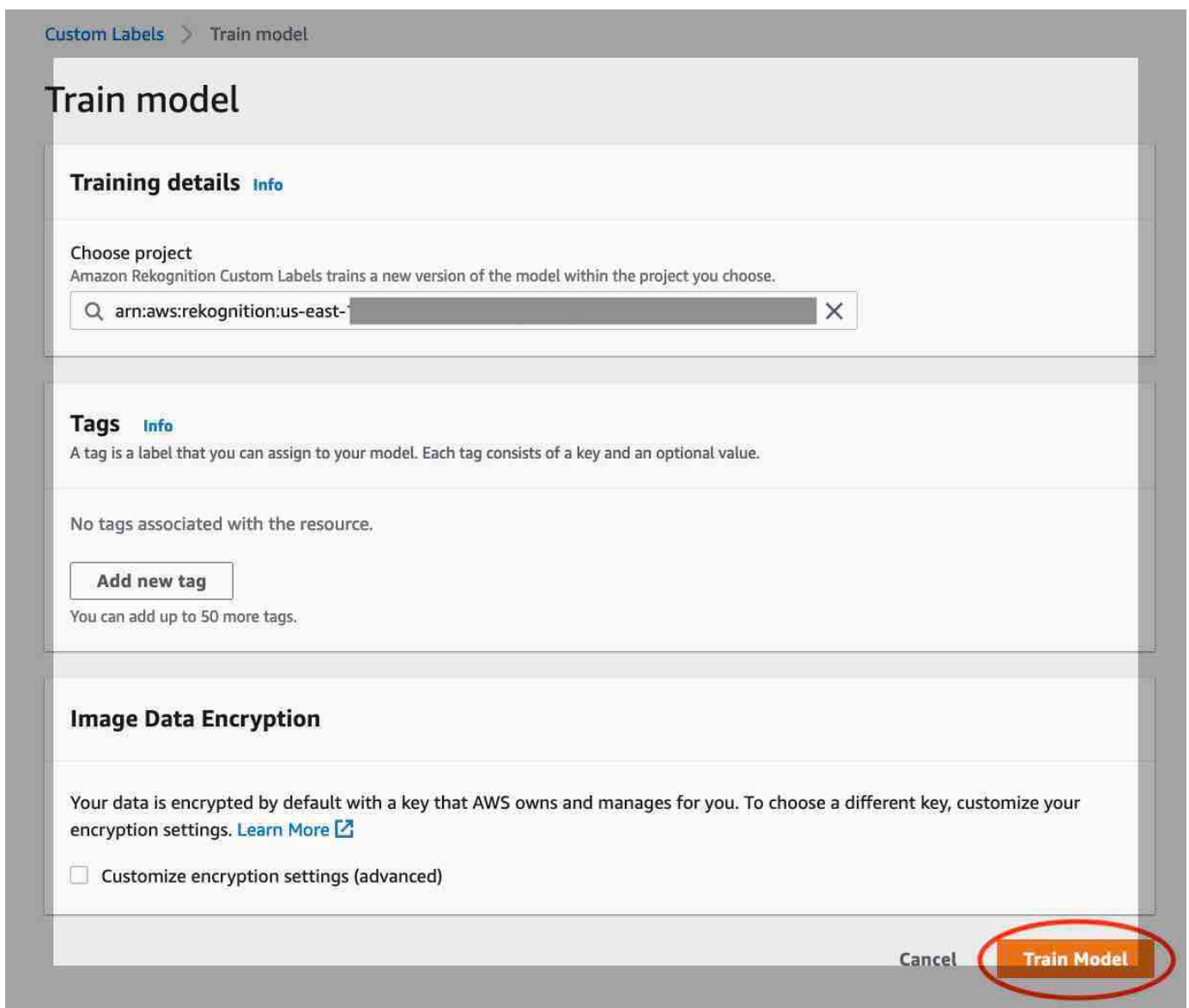
Au cours de cette étape, vous entraînez votre modèle. Les ensembles de données d'entraînement et de test sont automatiquement configurés pour vous. Une fois la formation terminée avec succès, vous pouvez voir les résultats de l'évaluation globale et les résultats de l'évaluation pour les images de test individuelles. Pour plus d'informations, veuillez consulter [Entraînement d'un modèle Amazon Rekognition Custom Labels Custom Labels](#).

Pour former votre modèle

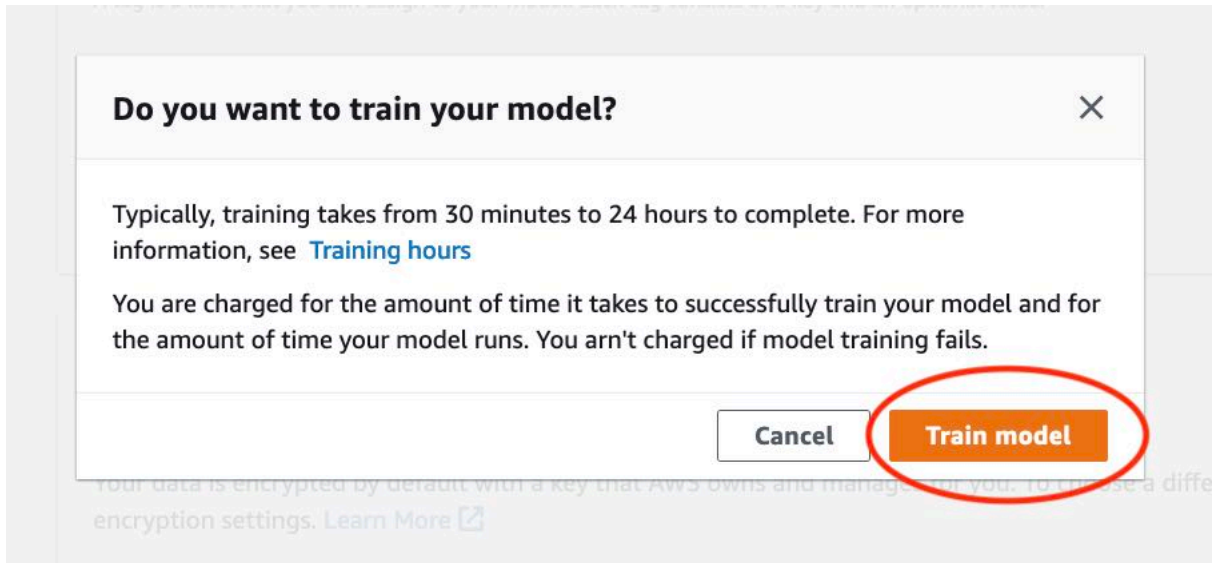
1. Sur la page du jeu de données, choisissez **Modèle de train**.



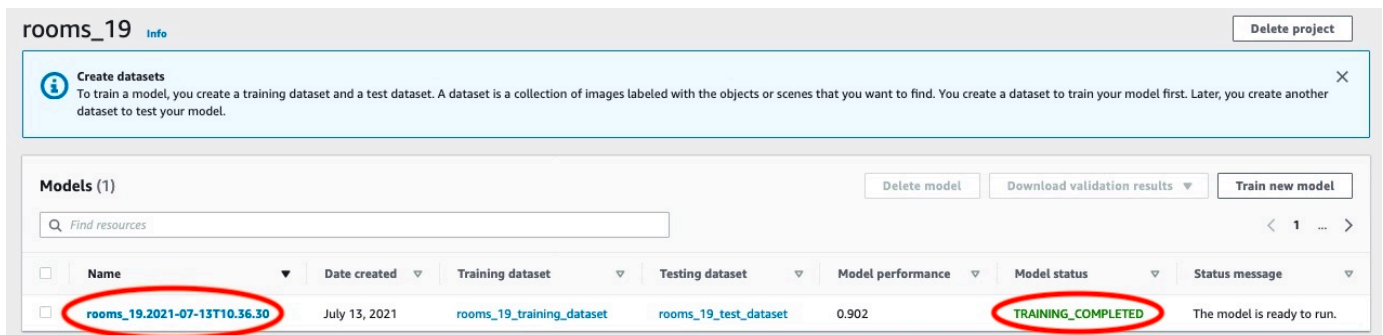
2. Sur le Modèle de trainpage, choisissezModèle de train. Le nom de ressource Amazon (ARN) de votre projet se trouve dansChoisissez un projetboîte d'édition.



3. Dans leVous souhaitez entraîner votre modèle ?boîte de dialogue, choisissezModèle de train.



4. Une fois la formation terminée, choisissez le nom du modèle. La formation est terminée lorsque l'état du modèle estFORMATION_TERMINÉE.



5. Choisissez leÉvaluerbouton pour voir les résultats de l'évaluation. Pour plus d'informations sur l'évaluation d'un modèle, voir[Amélioration d'un modèle d'étiquettes personnalisées Amazon Rekognition certifié](#).
6. ChoisissezAfficher les résultats des testspour voir les résultats des images de test individuelles.

rooms_19 [Info](#) Delete model

Evaluate | Model details | Use Model | Tags

Evaluation results View test results

F1 score Info 0.902	Average precision Info 0.893	Overall recall Info 0.928
Date completed July 13, 2021 Trained in 1.223 hours	Training dataset 10 labels, 61 images	Testing dataset 10 labels, 56 images

Per label performance (10)

Find labels < 1 >

Label name ▲	F1 score ▼	Test images ▼	Precision ▼	Recall ▼	Assumed threshold ▼
backyard	0.857	4	1.000	0.750	0.286
bathroom	0.889	9	0.889	0.889	0.185
bedroom	0.900	11	1.000	0.818	0.262
closet	1.000	2	1.000	1.000	0.169
entry_way	1.000	3	1.000	1.000	0.149
floor_plan	1.000	2	1.000	1.000	0.685

- Après avoir consulté les résultats du test, choisissez le nom du modèle pour revenir à la page du modèle.

The screenshot shows the Amazon Rekognition console interface. At the top, the breadcrumb navigation includes 'rooms_19' and 'rooms_19.2021-07-13T10.36.30', which is circled in red. Below the navigation is an 'Evaluate image' information box. The main content area is titled 'Images (56)' and contains a search bar and a list of images. Two images are displayed side-by-side:

- backyard2.jpeg**: Shows a house with a front porch. The evaluation results are:

Labels	Confidence
front_yard	30.3%
backyard	21.6%

The 'front_yard' label is marked as 'False positive' and the 'backyard' label is marked as 'False negative'.
- backyard4.jpeg**: Shows a lawn with a yellow fence. The evaluation results are:

Labels	Confidence
backyard	46.3%

The 'backyard' label is marked as 'True positive'.

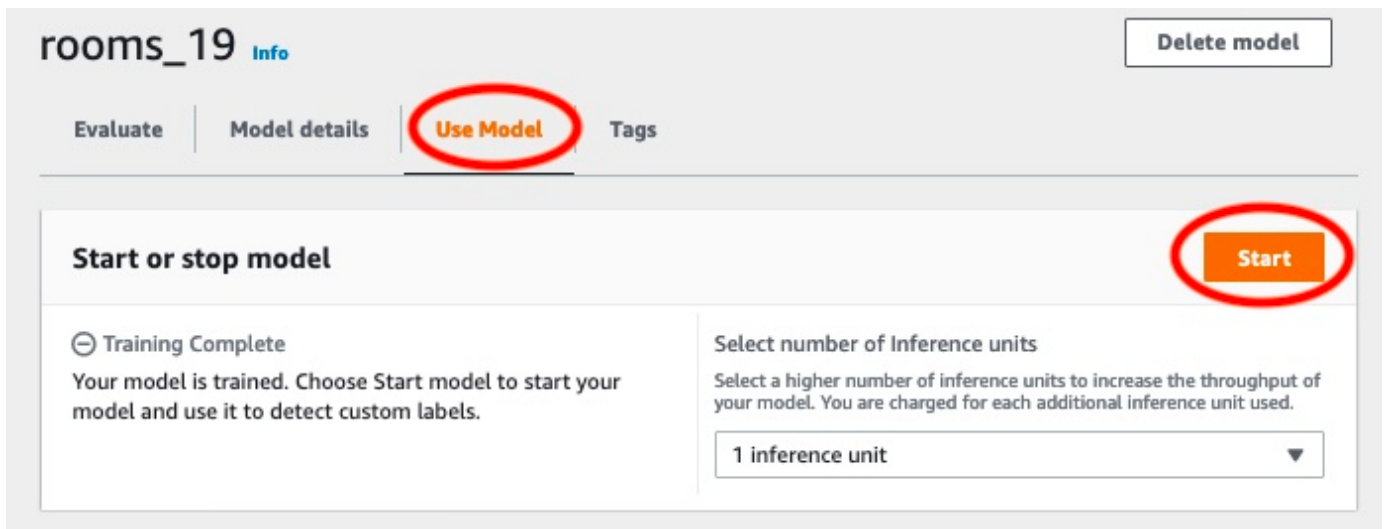
Étape 3 : Démarrez votre modèle

Au cours de cette étape, vous démarrez votre modèle. Une fois votre modèle démarré, vous pouvez l'utiliser pour analyser des images.

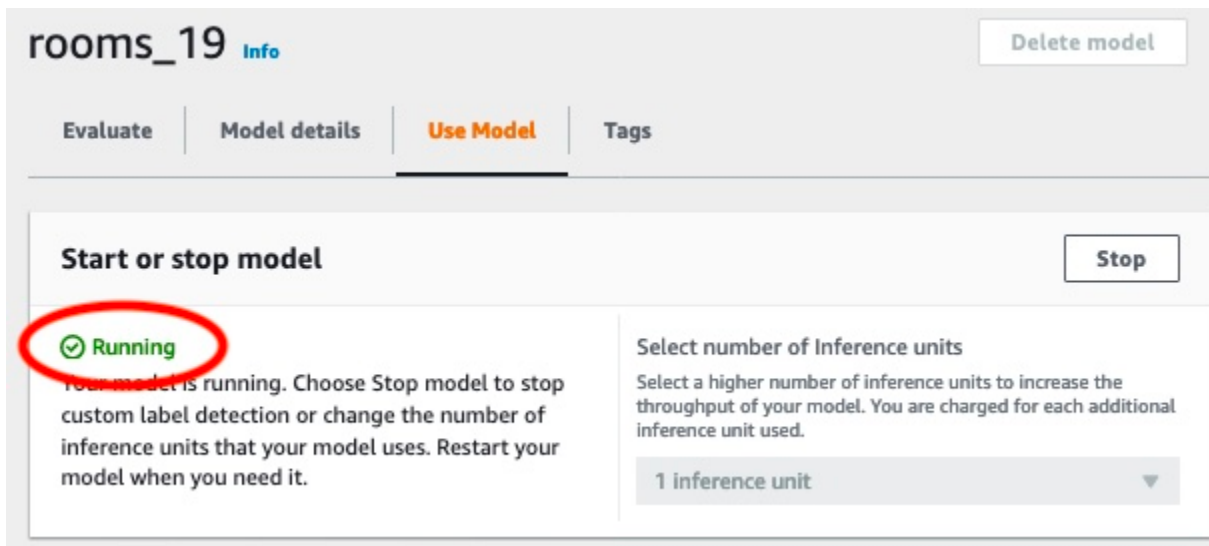
Vous êtes facturé en fonction de la durée de fonctionnement de votre modèle. Arrêtez votre modèle si vous n'avez pas besoin d'analyser les images. Vous pouvez redémarrer votre modèle ultérieurement. Pour plus d'informations, veuillez consulter [Exécution d'un modèle Étiquettes personnalisées Amazon Rekognition entraîné](#).

Pour démarrer votre modèle

1. Choisissez leModèle d'utilisationonglet sur la page du modèle.
2. Dans leModèle de démarrage ou d'arrêtsection procédez comme suit :
 - a. Sélectionnez Démarrer.
 - b. Dans leModèle de départboîte de dialogue, choisissezDémarrer.



3. Attendez que le modèle soit en cours d'exécution. Le modèle est en cours d'exécution lorsque l'état dans Modèle de démarrage ou d'arrêt la section est Courir.



4. Utilisez votre modèle pour classer les images. Pour plus d'informations, veuillez consulter [Étape 4 : Analyser une image avec votre modèle](#).

Étape 4 : Analyser une image avec votre modèle

Vous analysez une image en appelant le [DetectCustomLabels](#) API. Au cours de cette étape, vous utilisez le `detect-custom-labels` AWS Command Line Interface (AWS CLI) pour analyser un exemple d'image. Vous obtenez la commande AWS CLI depuis la console Amazon Rekognition Custom Labels. La console configure la commande AWS CLI pour utiliser votre modèle. Il vous suffit de

fournir une image stockée dans un compartiment Amazon S3. Cette rubrique fournit une image que vous pouvez utiliser pour chaque exemple de projet.

 Note

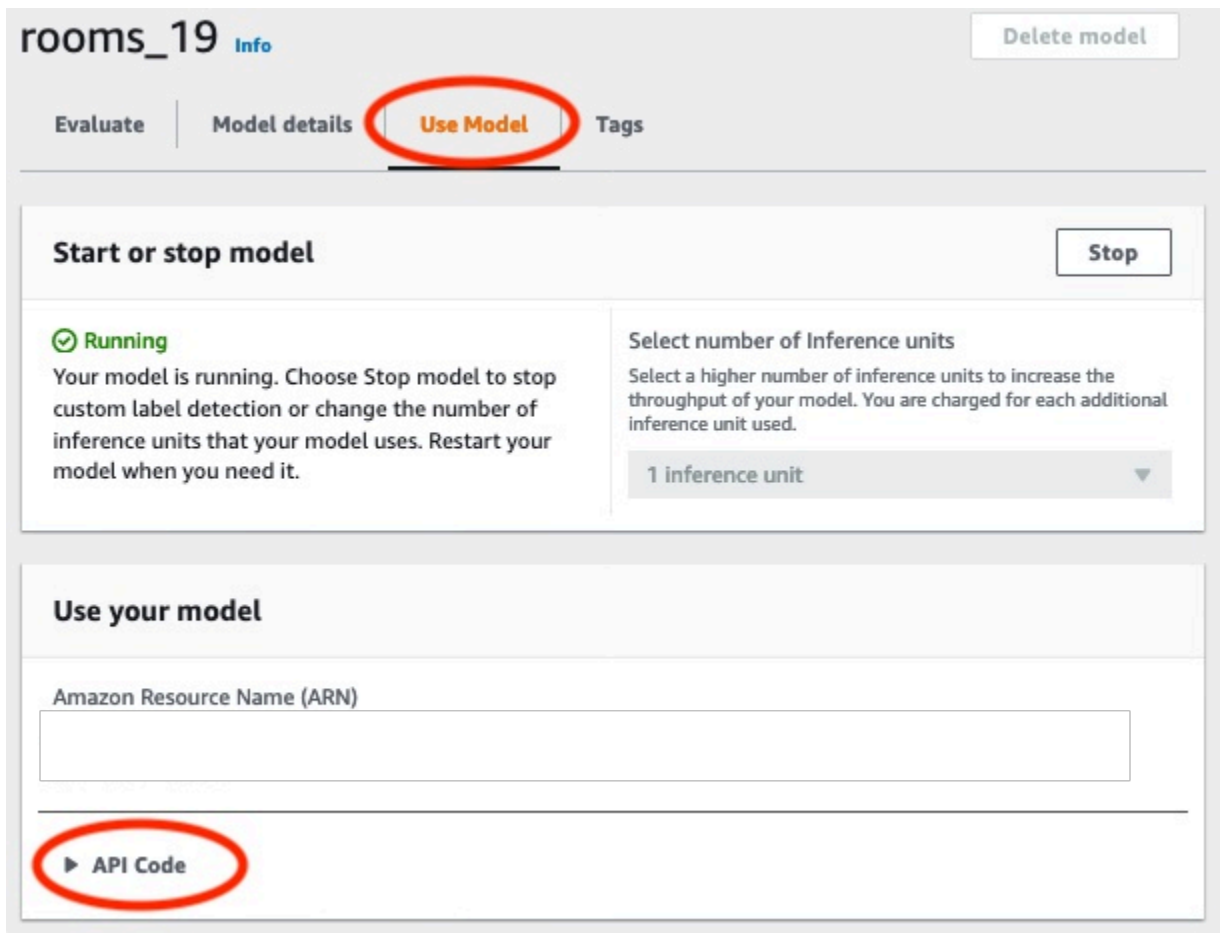
La console fournit également des exemples de code Python.

La sortie de `detect-custom-labels` inclut une liste des étiquettes présentes dans l'image, des cadres de délimitation (si le modèle trouve des emplacements d'objets) et indique le niveau de confiance du modèle dans la précision des prévisions.

Pour plus d'informations, veuillez consulter [Analyse d'une image avec un modèle entraîné](#).

Pour analyser une image (console)

1. Si ce n'est pas déjà fait, configurez AWS CLI. Pour des instructions, consultez [the section called "Étape 4 : Configurer AWS CLI et les kits AWS SDK"](#).
2. Si ce n'est pas déjà fait, commencez à exécuter votre modèle. Pour plus d'informations, veuillez consulter [Étape 3 : Démarrez votre modèle](#).
3. Choisissez le Modèle d'utilisation onglet puis choisissez Code d'API.



The screenshot displays the AWS Rekognition console interface for a custom model named 'rooms_19'. At the top, there are navigation tabs: 'Evaluate', 'Model details', 'Use Model' (highlighted with a red circle), and 'Tags'. A 'Delete model' button is located in the top right corner. The main content area is divided into two sections. The first section, 'Start or stop model', includes a 'Stop' button and a status indicator showing 'Running' with a green checkmark. Below the status, there is a dropdown menu for 'Select number of Inference units', currently set to '1 inference unit'. The second section, 'Use your model', contains a text input field for 'Amazon Resource Name (ARN)' and a button labeled '▶ API Code' (highlighted with a red circle).

4. Choisissez Commande de l'interface de ligne de commande AWS.
5. Dans leAnalyser l'imagesection, copiez leAWS CLIcommande qui appelle `detect-custom-labels`.

Use your model

Amazon Resource Name (ARN)

▼ API Code

Use your model rooms_ by calling the following AWS CLI commands or Python scripts. You can start and stop the model, and analyze custom labels in new images.

AWS CLI command

Python

Start model
Command used to start the rooms_ model.

```

1 aws rekognition start-project-version \
2   --project-version-arn "arn:aws:rekognition:us-east-1:
3   --min-inference-units 1 \
4   --region us-east-1
```

Analyze image
Command used to use analyze an image with the rooms_ model. Replace MY_BUCKET and PATH_TO_MY_IMAGE with your S3 bucket name and image path.

```

1 aws rekognition detect-custom-labels \
2   --project-version-arn "arn:aws:rekognition:us-east-1:
3   --image '{"S3Object": {"Bucket": "MY_BUCKET", "Name": "PATH_TO_MY_IMAGE"}}' \
4   --region us-east-1
```

6. Chargez un exemple d'image dans un compartiment Amazon S3. Pour des instructions, consultez [Obtenir un exemple d'image](#).
7. À l'invite de commande, entrez AWS CLI commande que vous avez copiée à l'étape précédente. Cela doit ressembler à l'exemple suivant.

La valeur de `--project-version-arn` doit être le nom de ressource Amazon (ARN) de votre modèle. La valeur de `--region` devrait être le AWS Région dans laquelle vous avez créé le modèle.

Changez `MY_BUCKET` et `PATH_TO_MY_IMAGE` au compartiment Amazon S3 et à l'image que vous avez utilisés à l'étape précédente.

Si vous utilisez le [custom-labels-access](#) profil pour obtenir des informations d'identification, ajoutez le `--profile custom-labels-access` paramètre.

```
aws rekognition detect-custom-labels \  
  --project-version-arn "model_arn" \  
  --image '{"S3Object": {"Bucket": "MY_BUCKET", "Name": "PATH_TO_MY_IMAGE"}}' \  
  --region us-east-1 \  
  --profile custom-labels-access
```

Si le modèle trouve des objets, des scènes et des concepts, la sortie JSON de l'AWS CLI commande doit ressembler à la suivante. `Name` est le nom de l'étiquette au niveau de l'image trouvée par le modèle. `Confidence` (0-100) est la confiance du modèle dans l'exactitude de la prédiction.

```
{  
  "CustomLabels": [  
    {  
      "Name": "living_space",  
      "Confidence": 83.41299819946289  
    }  
  ]  
}
```

Si le modèle trouve l'emplacement d'un objet ou trouve une marque, des cadres de délimitation étiquetés sont renvoyés. `BoundingBox` contient l'emplacement d'une boîte qui entoure l'objet. `Name` est l'objet que le modèle a trouvé dans le cadre de sélection. `Confidence` est la certitude du modèle quant au fait que le cadre de sélection contient l'objet.

```
{  
  "CustomLabels": [  
    {  
      "Name": "textextract",  
      "Confidence": 87.7729721069336,  
      "Geometry": {  
        "BoundingBox": {  
          "Width": 0.198987677693367,  
          "Height": 0.31296101212501526,  
          "Left": 0.07924537360668182,  
          "Top": 0.4037395715713501  
        }  
      }  
    }  
  ]  
}
```

}

8. Continuez à utiliser le modèle pour analyser d'autres images. Arrêtez le modèle si vous ne l'utilisez plus. Pour plus d'informations, veuillez consulter [Étape 5 : arrêtez votre modèle](#).

Obtenir un exemple d'image

Vous pouvez utiliser les images suivantes avec `DetectCustomLabel` opération. Il y a une image pour chaque projet. Pour utiliser les images, vous les chargez dans un compartiment S3.

Pour utiliser un exemple d'image

1. Cliquez avec le bouton droit sur l'image suivante qui correspond à l'exemple de projet que vous utilisez. Ensuite, choisissez `Enregistrer l'image` pour enregistrer l'image sur votre ordinateur. L'option du menu peut être différente selon le navigateur que vous utilisez.
2. Importez l'image dans un compartiment Amazon S3 qui appartient à votre `AWS` compte et se trouve dans le même `AWS` région dans laquelle vous utilisez les étiquettes personnalisées Amazon Rekognition.

Pour obtenir des instructions, voir [Chargement d'objets dans Amazon S3](#) dans le Guide de l'utilisateur d'Amazon Simple Storage Service.

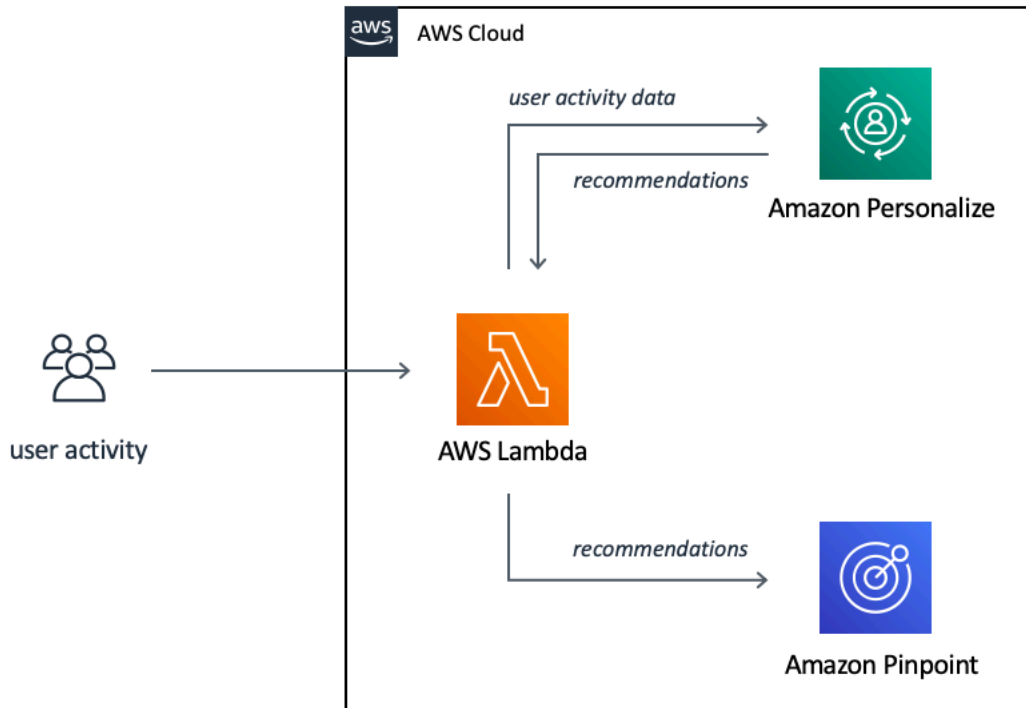
Image classification



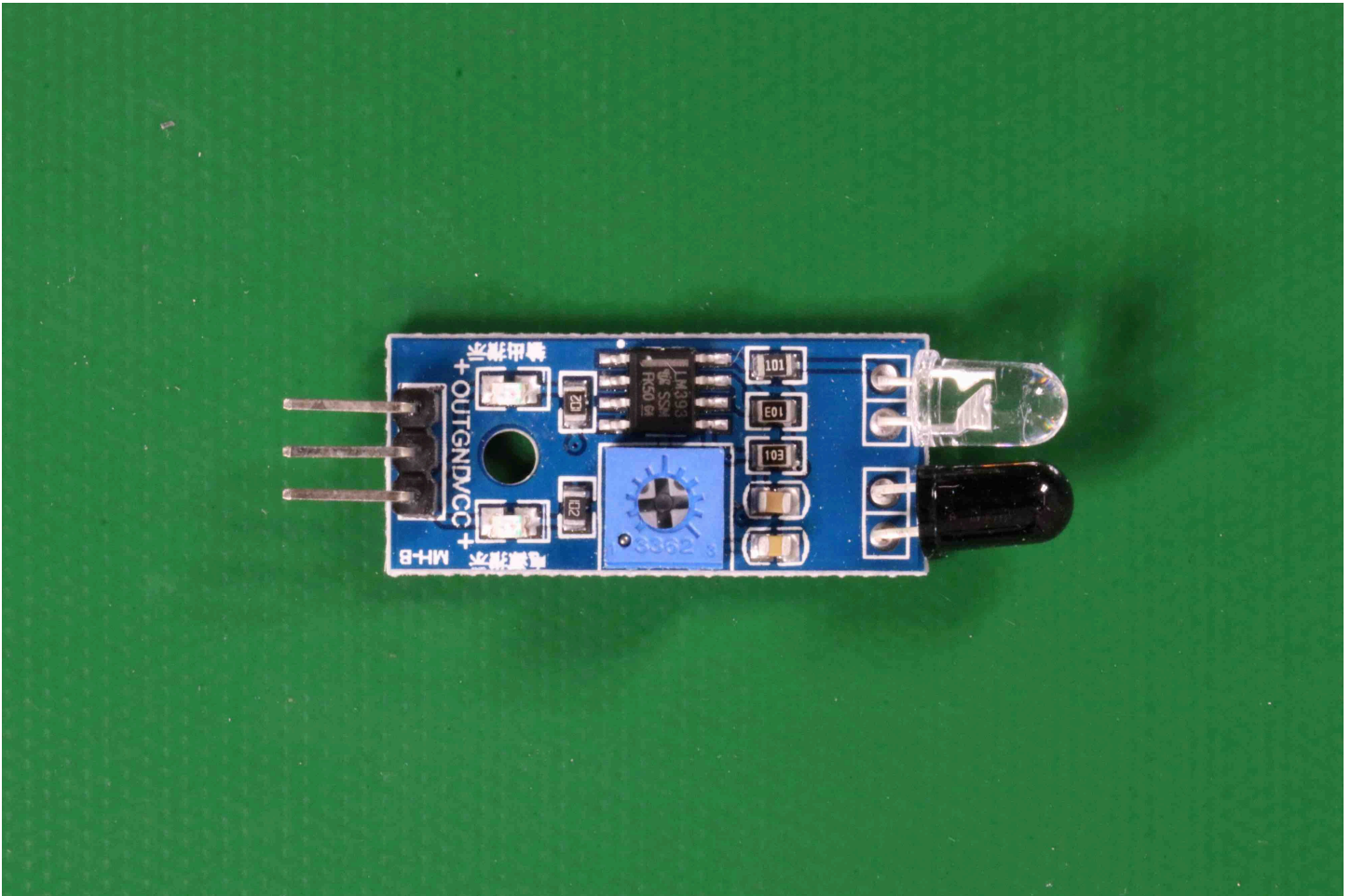
Classification multilabel



Détection de marque



Localisation d'objets

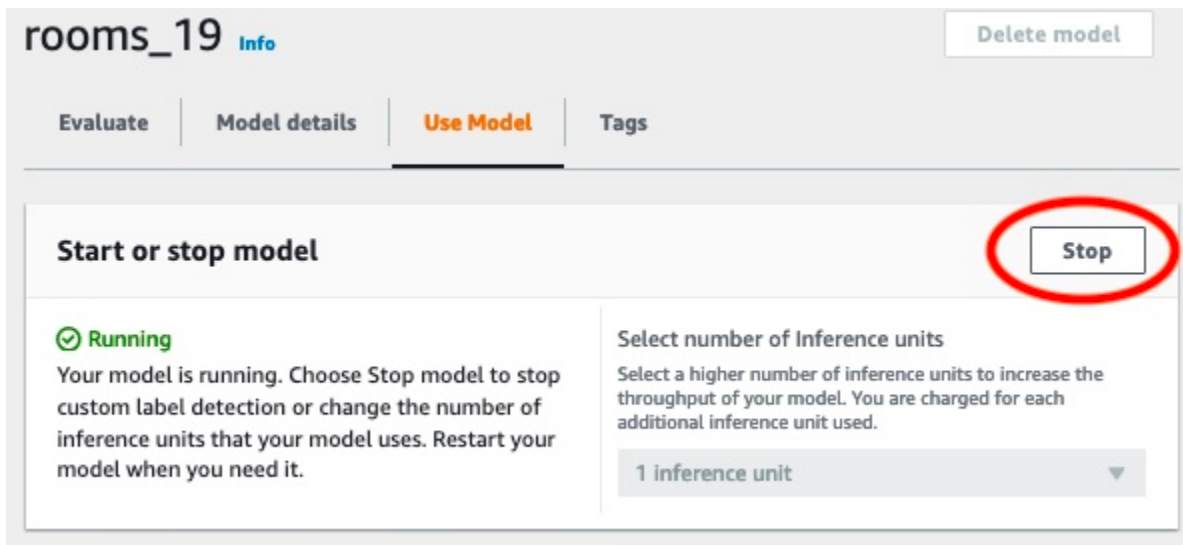


Étape 5 : arrêtez votre modèle

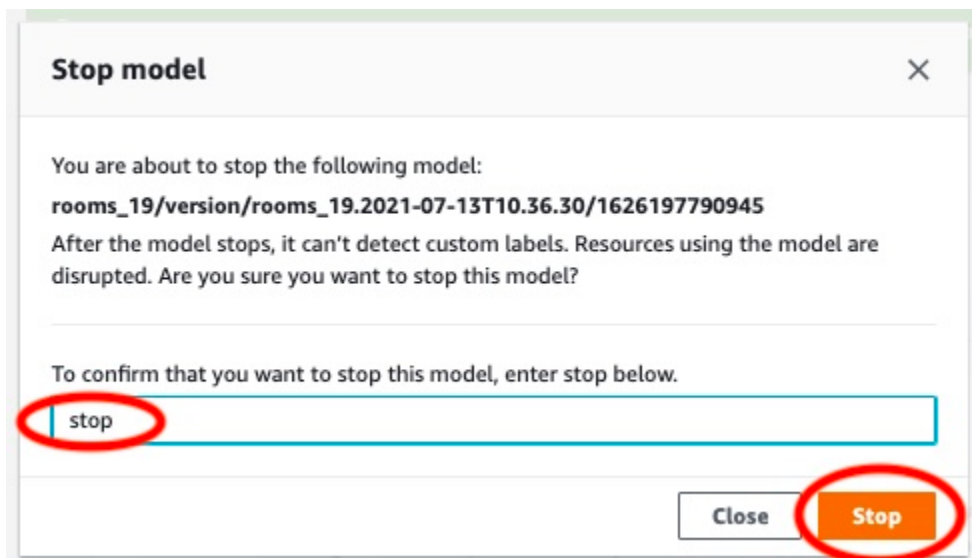
Au cours de cette étape, vous arrêtez d'exécuter votre modèle. Vous êtes facturé en fonction de la durée de fonctionnement de votre modèle. Si vous avez fini d'utiliser le modèle, vous devez l'arrêter.

Pour arrêter votre modèle

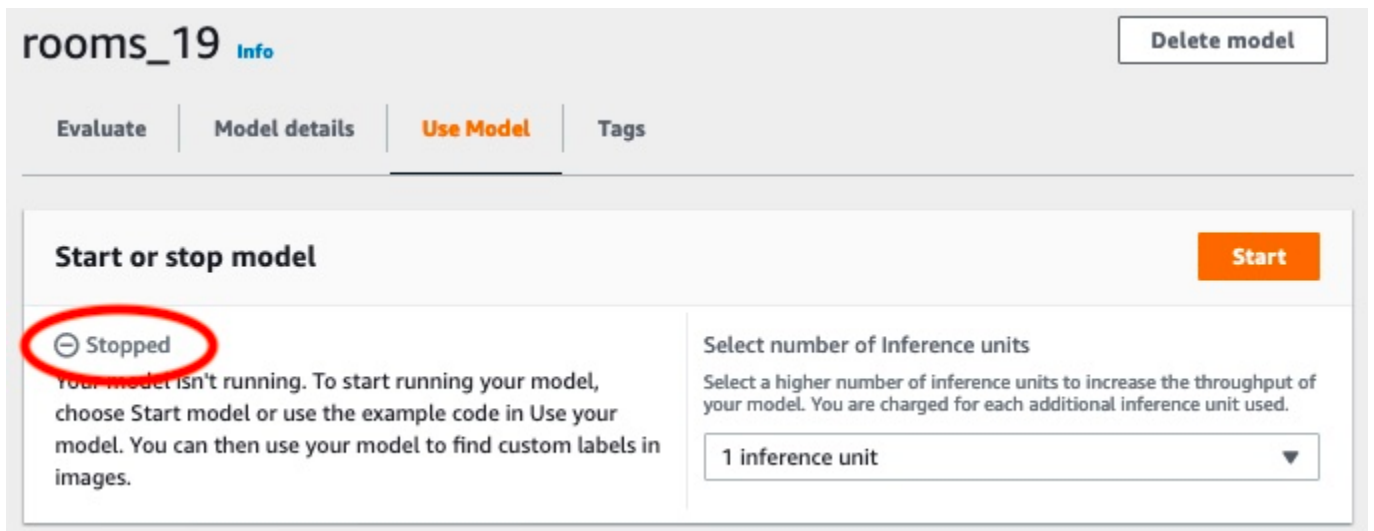
1. Dans le Modèle de démarrage ou d'arrêtsélection de la sectionArrête.



2. Dans le Modèle Stop boîte de dialogue, entrez arrête pour confirmer que vous souhaitez arrêter le modèle.



3. Choisissez Arrête pour arrêter votre modèle. Le modèle s'est arrêté lorsque l'état indiqué dans Modèle de démarrage ou d'arrêt la section est Arrêté.



Étape 6 : étapes suivantes

Après avoir essayé les exemples de projets, vous pouvez utiliser vos propres images et jeux de données pour créer votre propre modèle. Pour plus d'informations, veuillez consulter [Comprendre les étiquettes personnalisées Amazon Rekognition](#).

Utilisez les informations d'étiquetage du tableau suivant pour entraîner des modèles similaires aux exemples de projets.

Exemple	Images d'entraînement	Images de test
Classification des images (chambres)	1 étiquette au niveau de l'image par image	1 étiquette au niveau de l'image par image
Classification multiétiquette (fleurs)	Plusieurs étiquettes au niveau de l'image par image	Plusieurs étiquettes au niveau de l'image par image
Détection de marques (logos)	étiquettes de niveau d'image (vous pouvez également utiliser des cadres de délimitation étiquetés)	Cadres de délimitation étiquetés
Localisation d'images (circuits imprimés)	Cadres de délimitation étiquetés	Cadres de délimitation étiquetés

Le [Tutoriel : Classification d'images](#) explique comment créer un projet, des ensembles de données et des modèles pour un modèle de classification d'images.

Pour des informations détaillées sur la création de jeux de données et de modèles d'apprentissage, voir [Création d'un modèle d'étiquettes personnalisées Amazon Rekognition](#).

Tutoriel : Classification d'images

Ce didacticiel explique comment créer le projet et les jeux de données pour un modèle qui classe les objets, les scènes et les concepts présents dans une image. Le modèle classe l'ensemble de l'image. Par exemple, en suivant ce didacticiel, vous pouvez entraîner un modèle à reconnaître des emplacements domestiques tels qu'un salon ou une cuisine. Le didacticiel explique également comment utiliser le modèle pour analyser des images.

Avant de commencer le didacticiel, nous vous recommandons de lire [Comprendre les étiquettes personnalisées Amazon Rekognition](#).

Dans ce didacticiel, vous allez créer les ensembles de données d'entraînement et de test en téléchargeant des images depuis votre ordinateur local. Vous attribuerez ensuite des étiquettes au niveau de l'image aux images de vos ensembles de données d'entraînement et de test.

Le modèle que vous créez classe les images comme appartenant à l'ensemble d'étiquettes au niveau de l'image que vous attribuez aux images du jeu de données d'entraînement. Par exemple, si l'ensemble d'étiquettes au niveau de l'image de votre jeu de données d'entraînement est `kitchen,living_room,patio,etbackyard`, le modèle peut potentiellement trouver toutes ces étiquettes au niveau de l'image dans une seule image.

Note

Vous pouvez créer des modèles à différentes fins, par exemple pour trouver l'emplacement d'objets sur une image. Pour plus d'informations, veuillez consulter [Choisissez votre type de modèle](#).

Étape 1 : Collectez vos images

Vous avez besoin de deux séries d'images. Un ensemble à ajouter à votre jeu de données d'entraînement. Un autre ensemble à ajouter à votre jeu de données de test. Les images doivent représenter les objets, les scènes et les concepts que vous souhaitez classer dans votre modèle. Les images doivent être au format PNG ou JPEG. Pour plus d'informations, veuillez consulter [Préparation des images](#).

Vous devez disposer d'au moins 10 images pour votre jeu de données d'entraînement et 10 images pour votre jeu de données de test.

Si vous n'avez pas encore d'images, utilisez les images duChambresexemple de projet de classification. Une fois le projet créé, les images de formation et de test se trouvent aux emplacements des compartiments Amazon S3 suivants :

- Images de formation —s3://custom-labels-console-*region-numbers*/assets/rooms_*version number*_test_dataset/
- Images de test —s3://custom-labels-console-*region-numbers*/assets/rooms_*version number*_test_dataset/

region est leAWSRégion dans laquelle vous utilisez la console Amazon Rekognition Custom Labels.*number* est une valeur que la console attribue au nom du bucket.*Version number* est le numéro de version de l'exemple de projet, commençant à 1.

La procédure suivante permet de stocker les images du projet Rooms dans des dossiers locaux de votre ordinateur nommé*training*et*test*.

Pour télécharger les exemples de fichiers image du projet Rooms

1. Créez le projet Rooms. Pour plus d'informations, veuillez consulter [Étape 1 : Choisissez un exemple de projet](#).
2. Ouvrez l'invite de commande et entrez la commande suivante pour télécharger les images d'entraînement.

```
aws s3 cp s3://custom-labels-console-region-numbers/assets/rooms_version number_training_dataset/ training --recursive
```

3. À l'invite de recommandation, entrez la commande suivante pour télécharger les images de test.

```
aws s3 cp s3://custom-labels-console-region-numbers/assets/rooms_version number_test_dataset/ test --recursive
```

4. Déplacez deux des images du dossier de formation vers un dossier distinct de votre choix. Vous allez utiliser les images pour essayer votre modèle entraîné dans [Étape 9 : Analyser une image avec votre modèle](#).

Étape 2 : Choisissez vos cours

Dressez la liste des classes que vous souhaitez que votre modèle trouve. Par exemple, si vous apprenez à un mannequin à reconnaître les pièces d'une maison, vous pouvez classer l'image suivante comme `living_room`.



Chaque classe correspond à une étiquette au niveau de l'image. Vous attribuerez ensuite des étiquettes au niveau de l'image aux images de vos ensembles de données d'entraînement et de test.

Si vous utilisez les images de l'exemple de projet Rooms, les étiquettes au niveau de l'image sont `cour arrière`, `salle de bain`, `chambre`, `placard`, `entry_way`, `plan_d'étage`, `front_yard`, `cuisine`, `espace de vie`, `etpatio`.

Étape 3 : Création d'un projet

Pour gérer vos ensembles de données et vos modèles, vous créez un projet. Chaque projet doit porter sur un cas d'utilisation unique, tel que la reconnaissance des pièces d'une maison.

Pour créer un projet (console)

1. Si ce n'est pas déjà fait, configurez la console Amazon Rekognition Custom Labels. Pour plus d'informations, veuillez consulter [Configuration d'Étiquettes personnalisées Amazon Rekognition](#).
2. Connectez-vous au AWS Management Console et ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
3. Dans le volet de gauche, choisissez Utiliser des étiquettes personnalisées. La page d'accueil d'Amazon Rekognition Custom Labels s'affiche.

4. Sur la page d'accueil d'Amazon Rekognition Custom Labels, choisissez Commencez.
5. Dans le volet de navigation de gauche, choisissez Projets.
6. Sur la page des projets, choisissez Créer un projet.
7. Dans Project name (Nom de projet), saisissez un nom pour votre projet.
8. Choisissez Création d'un projet pour créer votre projet.

Custom Labels > Create project

Create project [Info](#)

Project details

Project name

The project name can't be more than 63 characters. It can only contain alphanumeric characters, with no spaces or special characters.

Cancel **Create project**

Étape 4 : créer des ensembles de données d'entraînement et de test

Au cours de cette étape, vous allez créer un jeu de données d'entraînement et un jeu de données de test en téléchargeant des images depuis votre ordinateur local. Vous pouvez télécharger jusqu'à 30 images à la fois. Si vous avez un grand nombre d'images à charger, pensez à créer les ensembles de données en les important depuis un compartiment Amazon S3. Pour plus d'informations, veuillez consulter [Compartiment Amazon S3](#).

Pour plus d'informations sur les ensembles de données, consultez [Gestion des jeux de données](#).

Pour créer un jeu de données à l'aide d'images sur un ordinateur local (console)

1. Sur la page des détails du projet, choisissez Créer un jeu de données.

How it works

Creating your dataset

1. Create dataset
A dataset is a collection of images, and image labels, that you use to train or test a model.

2. Label images
Labels identify objects, scenes, or concepts on an entire image, or they identify object locations on an image.

3. Train model
Depending on the training dataset, the training model finds image-level scenes and concepts, or it finds object locations.

4. Check performance metrics
Performance metrics tell you if your model needs additional training before you can use it.

Project details

2. Dans la Configuration de démarrage section, choisissez Commencez par un jeu de données d'entraînement et un jeu de données de test.
3. Dans le Détails du jeu de données d'entraînement section, choisissez Téléchargez des images depuis votre ordinateur.
4. Dans le Détails du jeu de données de test section, choisissez Téléchargez des images depuis votre ordinateur.
5. Choisissez Création de jeux de données.

Create dataset Info

Starting configuration

Configuration options

Start with a single dataset
When you train your model, the dataset is split to create the training dataset (80%) and test dataset (20%) for your project.

Start with a training dataset and a test dataset
Recommended for most users. Start with the highest control over training, testing, and performance tuning.

What are training datasets and test datasets?

- A training dataset teaches your model to identify scenes or objects in images.
- A test dataset evaluates the performance of your trained model.

Training dataset details

Import images Info

Import images from one of the sources below.

Import images from S3 bucket
Use images from an existing S3 bucket by entering the S3 bucket URL. You can automatically add labels based on your S3 bucket folder names.

Upload images from your computer
Add images by uploading files from your local computer. You're limited to uploading 50 images at one time.

Copy an existing Amazon Rekognition Custom Labels dataset
Use an existing dataset as a starting point for your new dataset. Your original dataset will remain unchanged.

Import images labeled by SageMaker Ground Truth
Provide the location of your manifest file. If you have a labeled datasets in a different format, convert them to a manifest format.

Test dataset details

Import images Info

Import images from one of the sources below.

Import images from S3 bucket
Use images from an existing S3 bucket by entering the S3 bucket URL. You can automatically add labels based on your S3 bucket folder names.

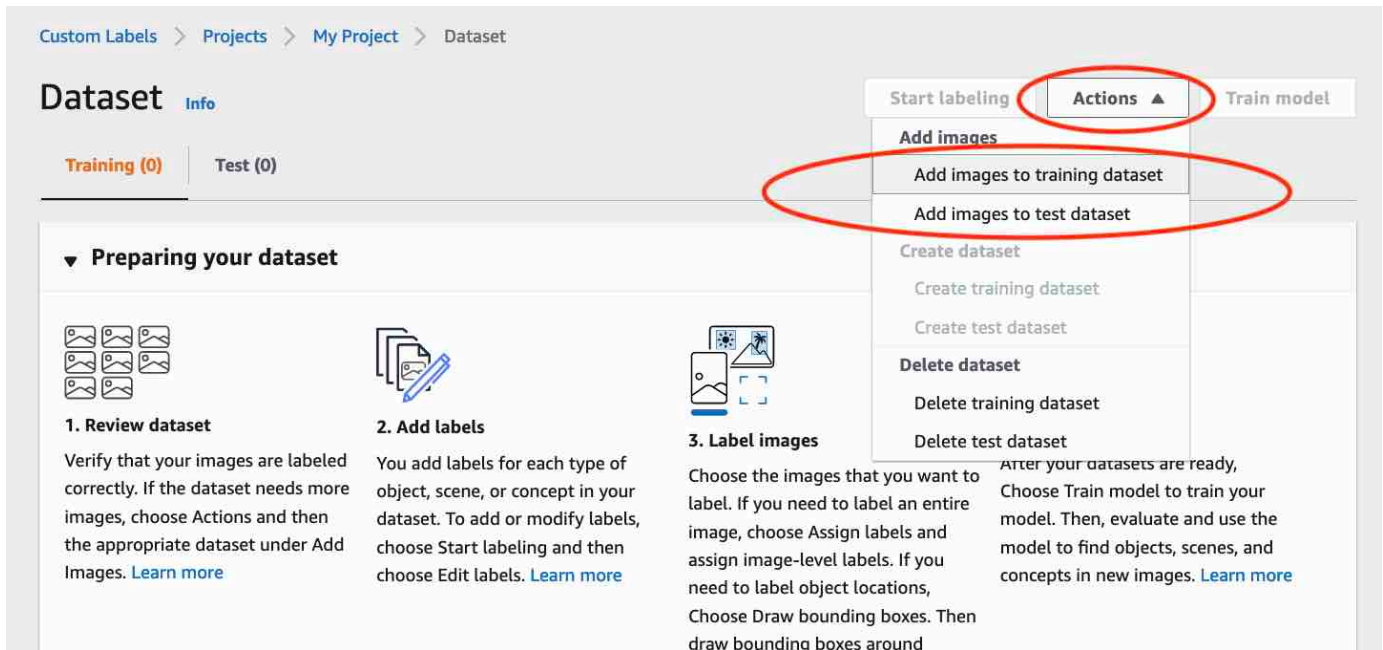
Upload images from your computer
Add images by uploading files from your local computer. You're limited to uploading 50 images at one time.

Copy an existing Amazon Rekognition Custom Labels dataset
Use an existing dataset as a starting point for your new dataset. Your original dataset will remain unchanged.

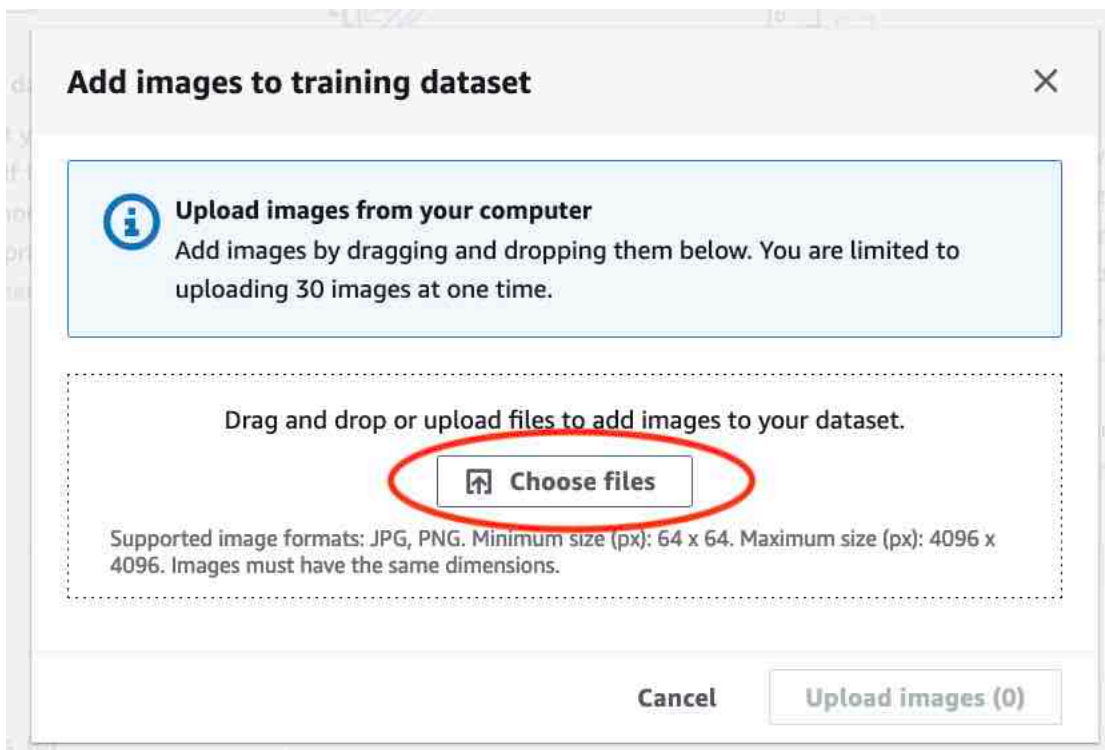
Import images labeled by SageMaker Ground Truth
Provide the location of your manifest file. If you have a labeled datasets in a different format, convert them to a manifest format.

Cancel

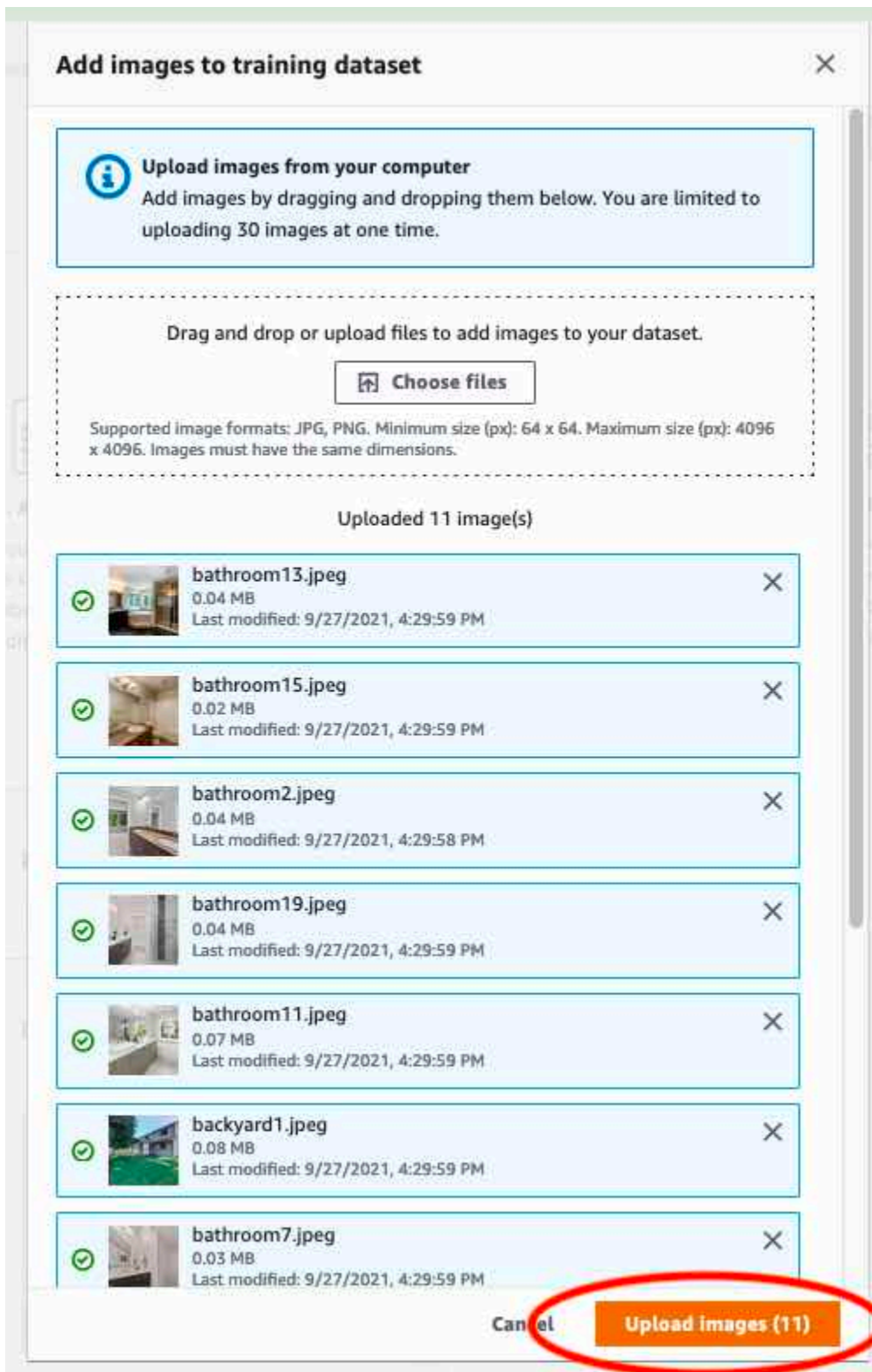
6. Une page de jeu de données s'affiche avec **Entraînement** onglet et **aTest** onglet pour les ensembles de données respectifs.
7. Sur la page du jeu de données, choisissez **Entraînement** onglet.
8. Choisissez **Actions** puis choisissez **Ajouter des images** au jeu de données d'entraînement.



9. Dans le **Ajouter des images** au jeu de données d'entraînement boîte de dialogue, choisissez **Choisissez des fichiers**.



10. Choisissez les images que vous souhaitez charger dans le jeu de données. Vous pouvez télécharger jusqu'à 30 images à la fois.
11. Choisissez Charger des images. L'ajout des images au jeu de données par Amazon Rekognition Custom Labels peut prendre quelques secondes.



12. Si vous avez d'autres images à ajouter au jeu de données d'entraînement, répétez les étapes 9 à 12.

13. Choisissez l'onglet Test.

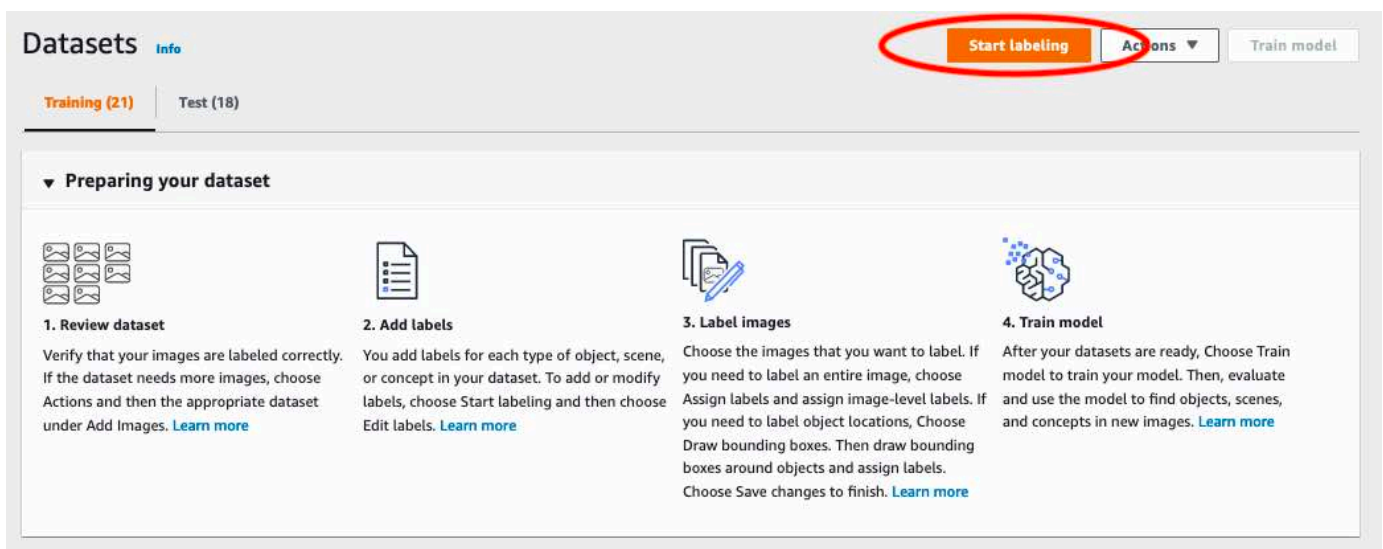
14. Répétez les étapes 8 à 12 pour ajouter des images au jeu de données de test. Pour l'étape 8, choisissez **Actions** puis choisissez **Ajouter des images au jeu de données de test**.

Étape 5 : Ajouter des étiquettes au projet

Au cours de cette étape, vous ajoutez une étiquette au projet pour chacune des classes que vous avez identifiées à l'étape 1. [Étape 2 : Choisissez vos cours](#).

Pour ajouter une nouvelle étiquette (console)

1. Sur la page de la galerie de jeux de données, choisissez **Commencez à étiqueter** pour passer en mode étiquetage.



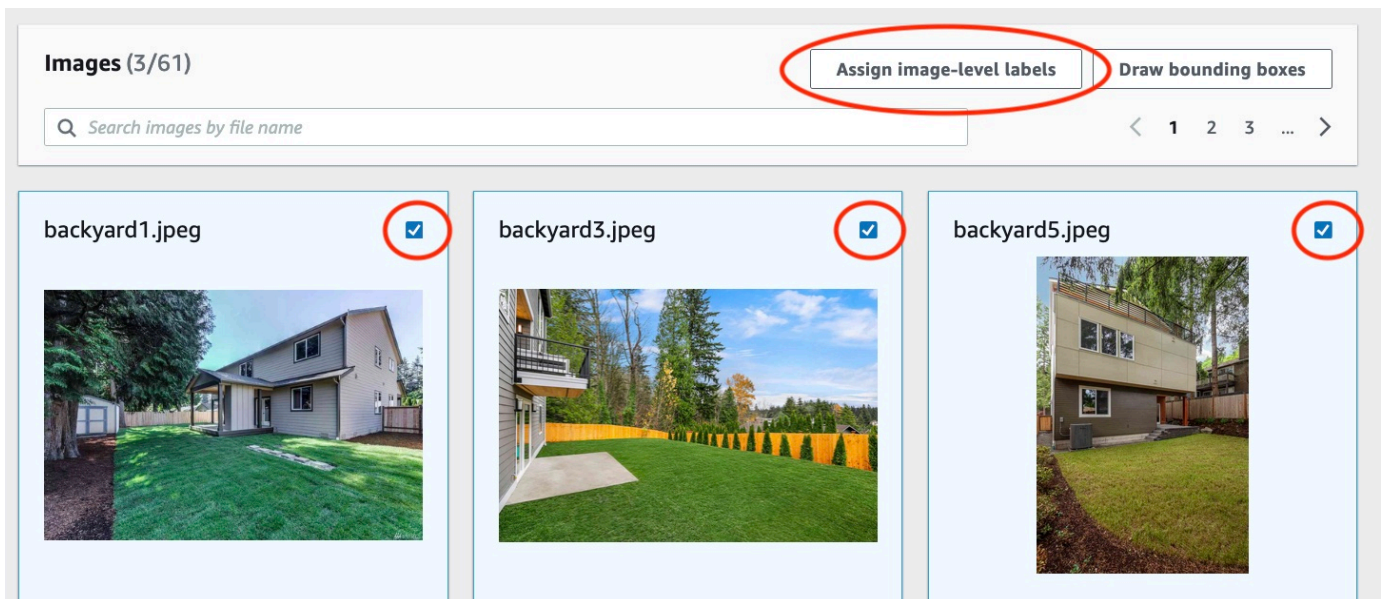
2. Dans le **Étiquettes** section de la galerie de jeux de données, choisissez **Modifier les étiquettes** pour ouvrir le **Gérer les étiquettes** boîte de dialogue.
3. Dans la zone d'édition, entrez un nouveau nom d'étiquette.
4. Choisissez **Ajouter une étiquette**.
5. Répétez les étapes 3 et 4 jusqu'à ce que vous ayez créé toutes les étiquettes dont vous avez besoin.
6. Choisissez **Enregistrer** pour enregistrer les étiquettes que vous avez ajoutées.

Étape 6 : Attribuer des étiquettes au niveau de l'image aux ensembles de données d'entraînement et de test

Au cours de cette étape, vous attribuez un seul niveau d'image à chaque image de vos ensembles de données d'entraînement et de test. L'étiquette au niveau de l'image est la classe que chaque image représente.

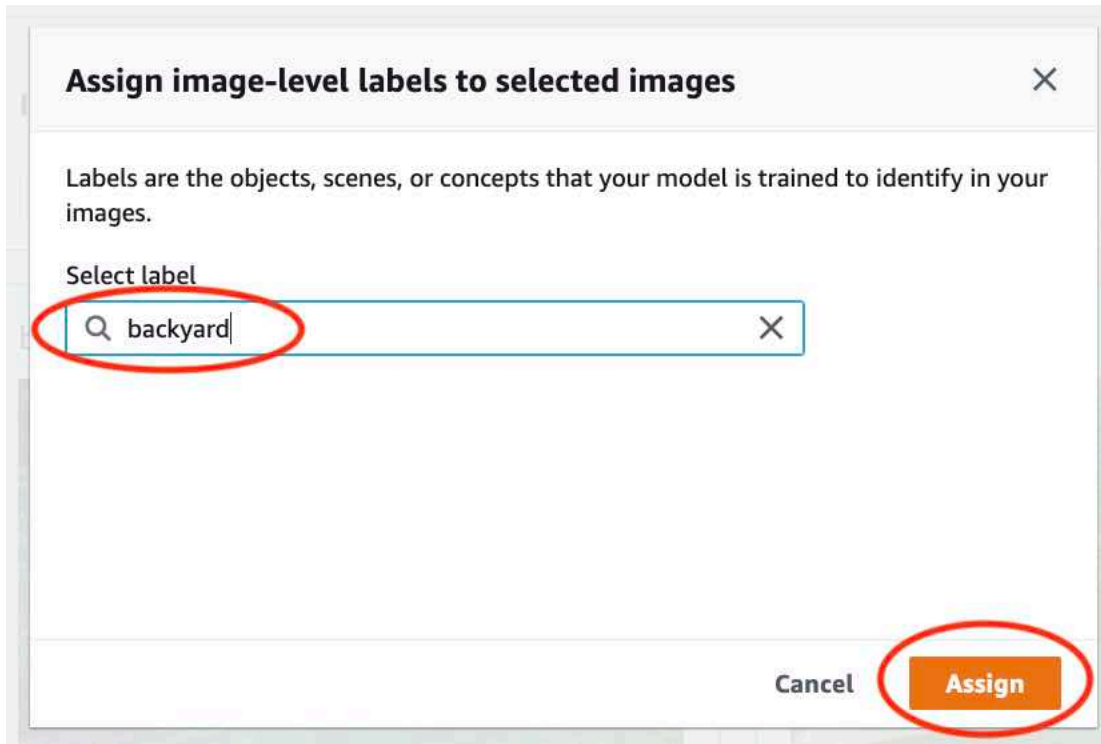
Pour attribuer des étiquettes au niveau de l'image à une image (console)

1. Sur le **Ensembles de données** page, choisissez **Entraînement** onglet.
2. Choisissez **Commencez à étiqueter** pour passer en mode étiquetage.
3. Sélectionnez une ou plusieurs images auxquelles vous souhaitez ajouter des étiquettes. Vous ne pouvez sélectionner des images que sur une seule page à la fois. Pour sélectionner une plage contiguë d'images sur une page :
 - a. Sélectionnez la première image.
 - b. Appuyez sur la touche Shift et maintenez-la enfoncée.
 - c. Sélectionnez la deuxième image. Les images situées entre la première et la seconde image sont également sélectionnées.
 - d. Relâchez la touche Shift.
4. Choisissez **Attribuer des étiquettes au niveau de l'image**.



5. Dans **Attribuer des étiquettes au niveau de l'image aux images sélectionnées** boîte de dialogue, sélectionnez le libellé que vous souhaitez attribuer à l'image ou aux images.

6. Choisissez **Attribuer** pour attribuer une étiquette à l'image.



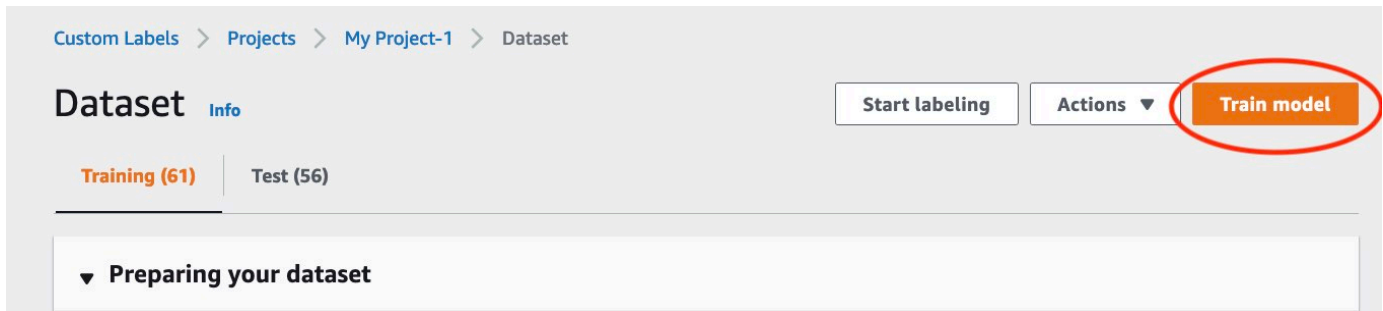
7. Répétez l'étiquetage jusqu'à ce que chaque image soit annotée avec les étiquettes requises.
8. Choisissez l'onglet Test.
9. Répétez les étapes pour attribuer des étiquettes de niveau d'image aux images du jeu de données de test.

Étape 7 : Entraînez votre modèle

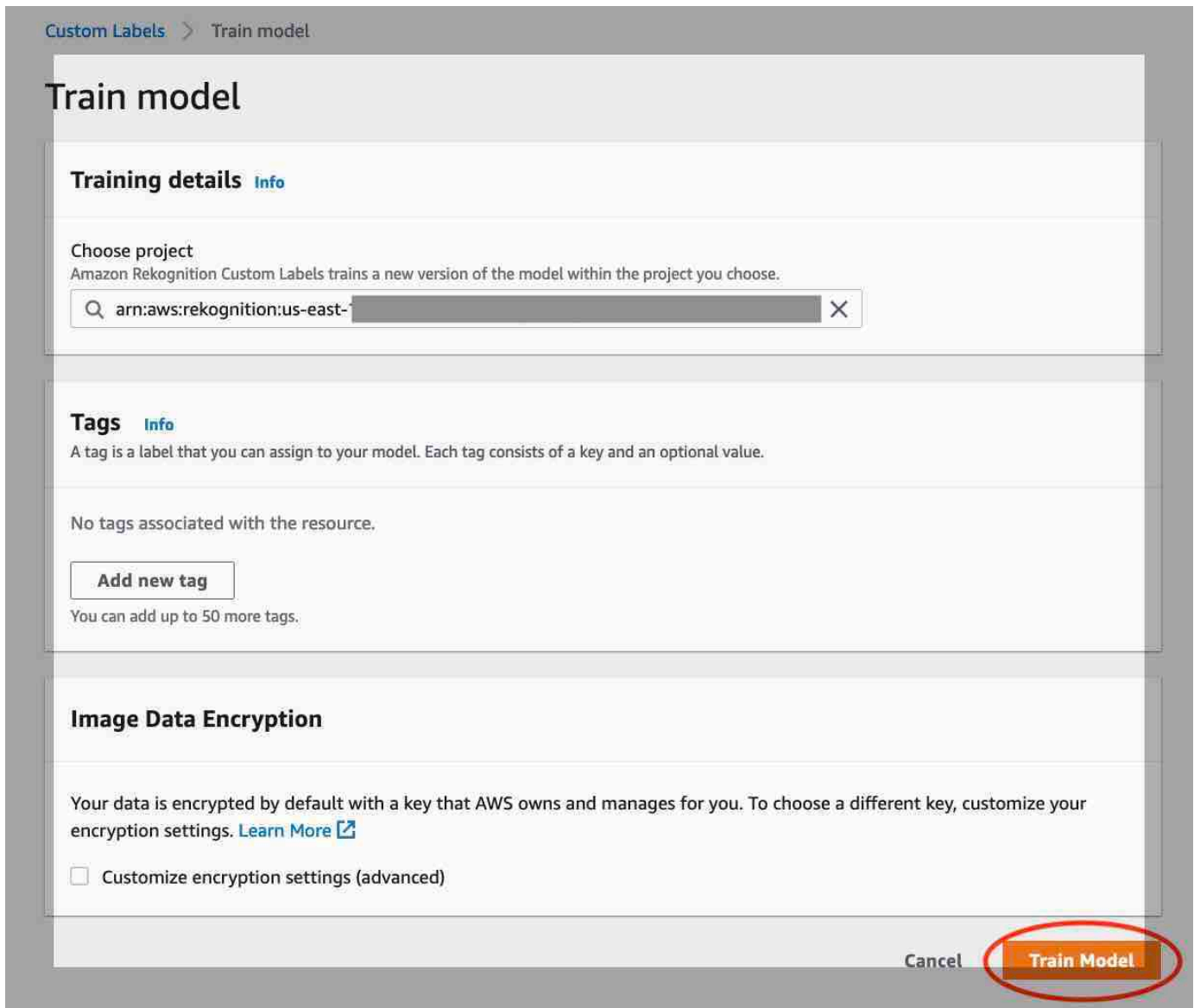
Procédez comme suit pour entraîner votre modèle. Pour plus d'informations, veuillez consulter [Entraînement d'un modèle Amazon Rekognition Custom Labels Custom Labels](#).

Pour entraîner votre modèle (console)

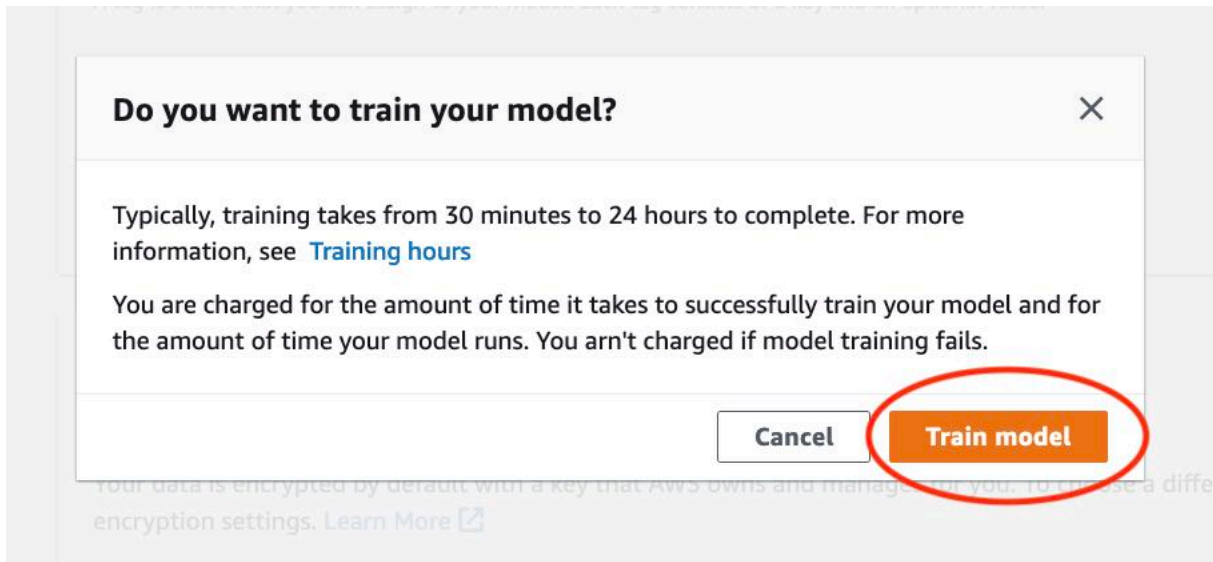
1. Sur le **Ensemble de données** page, choisissez **Modèle de train**.



2. Sur leModèle de trainpage, choisissezModèle de train. Le nom de ressource Amazon (ARN) de votre projet se trouve dansChoisissez un projetboîte d'édition.



3. Dans leVous souhaitez entraîner votre modèle ?boîte de dialogue, choisissezModèle de train.




4. Dans leModèlesdans la section de la page du projet, vous pouvez voir que la formation est en cours. Vous pouvez vérifier l'état actuel en consultantMode1 Statuscolonne pour la version du modèle. La formation d'un modèle prend un certain temps.

Custom Labels > Projects > My-Project-1

My-Project-1 Info

▼ How it works

Creating your dataset




1. Create dataset
A dataset is a collection of images, and image labels, that you use to train or test a model.

✔ Created

2. Label images
Labels identify objects, scenes, or concepts on an entire image, or they identify object locations on an image.

Label images


Training your model



3. Train model
Depending on the training dataset, the training model finds image-level scenes and concepts, or it finds object locations.

Train model

Evaluating your model



4. Check performance metrics
Performance metrics tell you if your model needs additional training before you can use it.

Check metrics

Project details

Project name My-Project-1	Created October 04, 2021 at 13:05:06 (UTC-07:00)	Dataset ↻	Models 1
------------------------------	---	--------------	-------------

Models (1) Delete model Download validation results ▼

<input type="checkbox"/>	Name	Date created	Training dataset	Test dataset	Model performance (F1 score)	Model status	Status message
<input type="checkbox"/>	My-Project-1.2021-10-04T13.52.53	October 04, 2021			N/A	TRAINING_IN_PROGRESS	The model is being trained.

5. Une fois la formation terminée, choisissez le nom du modèle. La formation est terminée lorsque l'état du modèle est `FORMATION_TERMINÉE`.

rooms_19 Info Delete project

Create datasets
To train a model, you create a training dataset and a test dataset. A dataset is a collection of images labeled with the objects or scenes that you want to find. You create a dataset to train your model first. Later, you create another dataset to test your model.

Models (1) Delete model Download validation results ▼ Train new model

<input type="checkbox"/>	Name	Date created	Training dataset	Testing dataset	Model performance	Model status	Status message
<input type="checkbox"/>	rooms_19.2021-07-13T10.36.30	July 13, 2021	rooms_19_training_dataset	rooms_19_test_dataset	0.902	TRAINING_COMPLETED	The model is ready to run.

6. Choisissez le `Évaluer` bouton pour voir les résultats de l'évaluation. Pour plus d'informations sur l'évaluation d'un modèle, voir [Amélioration d'un modèle d'étiquettes personnalisées Amazon Rekognition certifié](#).
7. Choisissez `Afficher les résultats des tests` pour voir les résultats des images de test individuelles. Pour plus d'informations, veuillez consulter [Métriques pour évaluer votre modèle](#).

rooms_19 Info Delete model

Evaluate | Model details | Use Model | Tags

Evaluation results View test results

F1 score <small>Info</small> 0.902	Average precision <small>Info</small> 0.893	Overall recall <small>Info</small> 0.928
Date completed July 13, 2021 Trained in 1.223 hours	Training dataset 10 labels, 61 images	Testing dataset 10 labels, 56 images

Per label performance (10)

Find labels < 1 >

Label name ▲	F1 score ▼	Test images ▼	Precision ▼	Recall ▼	Assumed threshold ▼
backyard	0.857	4	1.000	0.750	0.286
bathroom	0.889	9	0.889	0.889	0.185
bedroom	0.900	11	1.000	0.818	0.262
closet	1.000	2	1.000	1.000	0.169
entry_way	1.000	3	1.000	1.000	0.149
floor_plan	1.000	2	1.000	1.000	0.685

- Après avoir consulté les résultats du test, choisissez le nom du modèle pour revenir à la page du modèle.

The screenshot shows the Amazon Rekognition console interface. At the top, the breadcrumb navigation includes 'rooms_19' and 'rooms_19.2021-07-13T10.36.30', which is circled in red. Below the navigation is an 'Evaluate image' information box. The main content area is divided into a 'Filter by label' sidebar on the left and a main image display area on the right. The sidebar has a search box for labels and three checkboxes: 'True positive', 'False positive', and 'False negative'. The main area shows two image cards. The first card, 'backyard2.jpeg', shows a house with a front porch. Below the image, a table lists predicted labels: 'front_yard' (30.3% confidence, False positive) and 'backyard' (21.6% confidence, False negative). The second card, 'backyard4.jpeg', shows a lawn with a fence. Below the image, a table lists the predicted label: 'backyard' (46.3% confidence, True positive).

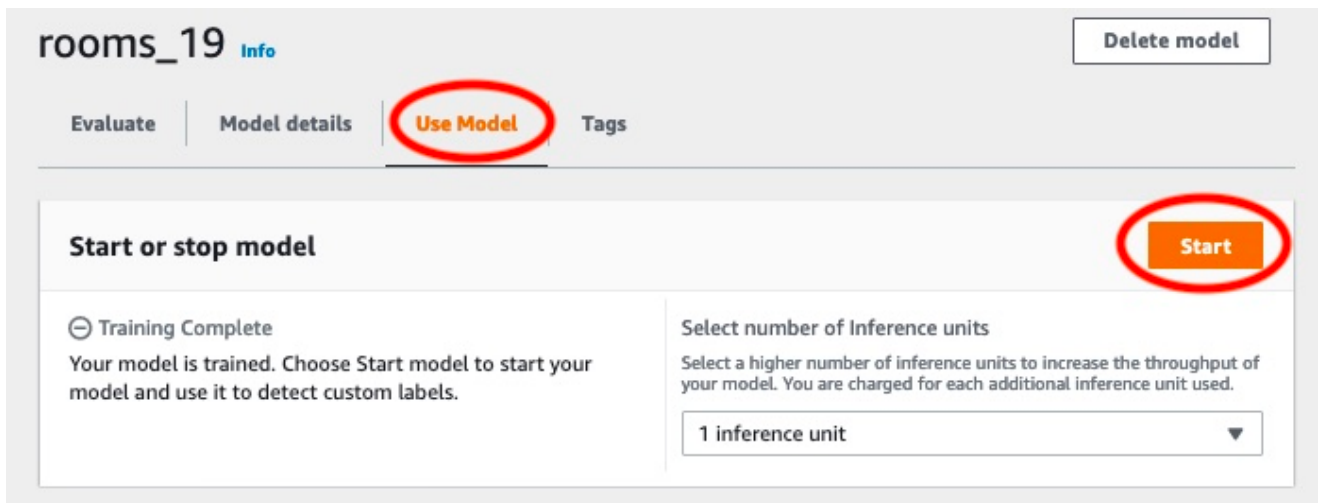
Étape 8 : Démarrez votre modèle

Au cours de cette étape, vous démarrez votre modèle. Une fois votre modèle démarré, vous pouvez l'utiliser pour analyser des images.

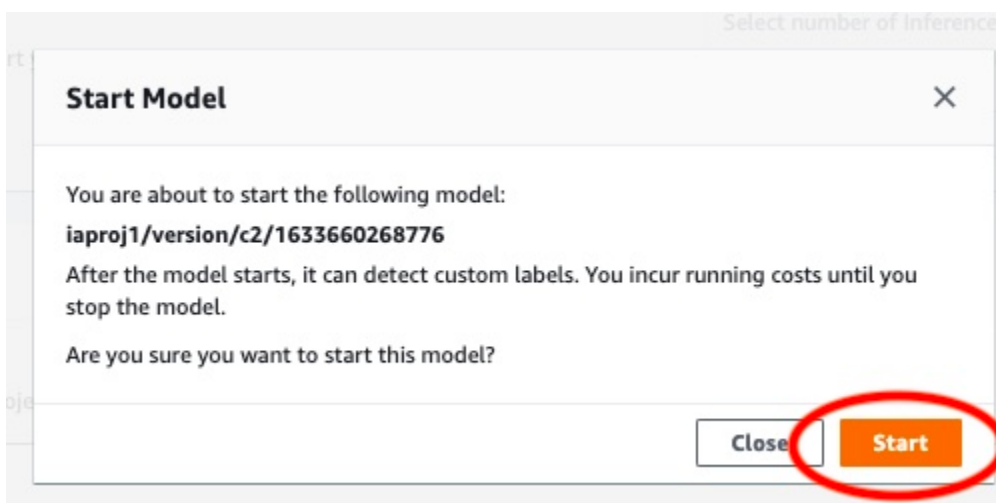
Vous êtes facturé en fonction de la durée de fonctionnement de votre modèle. Arrêtez votre modèle si vous n'avez pas besoin d'analyser les images. Vous pouvez redémarrer votre modèle ultérieurement. Pour plus d'informations, veuillez consulter [Exécution d'un modèle Étiquettes personnalisées Amazon Rekognition entraîné](#).

Pour démarrer votre modèle

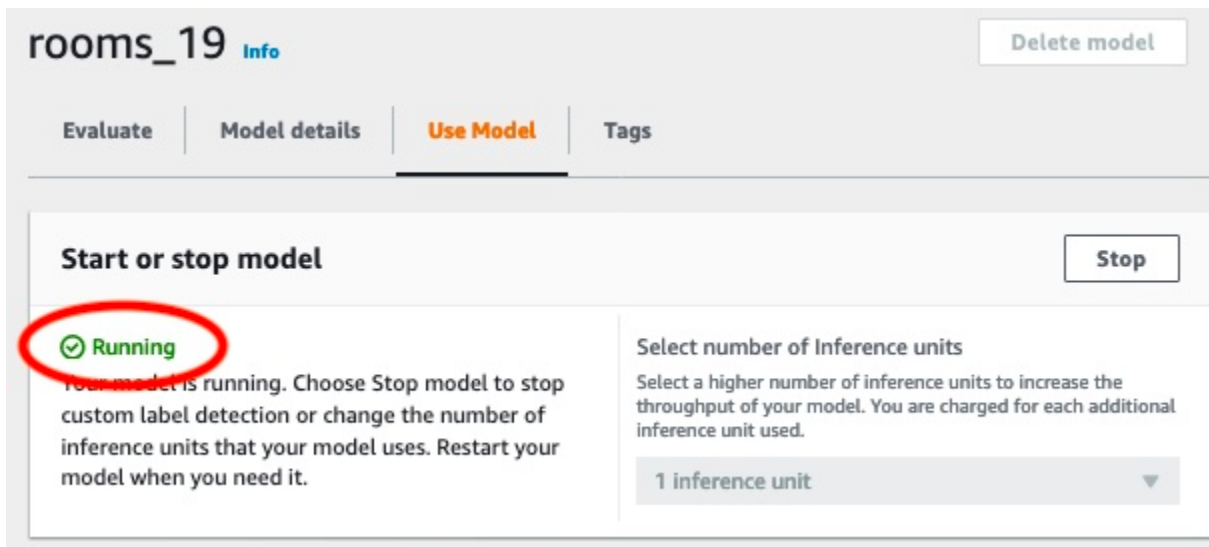
1. Choisissez leModèle d'utilisationonglet sur la page du modèle.
2. Dans leModèle de démarrage ou d'arrêtsection procédez comme suit :
 - a. Sélectionnez Démarrer.



- b. Dans le Modèle de départboîte de dialogue, choisissez Démarrer.



3. Attendez que le modèle soit en cours d'exécution. Le modèle est en cours d'exécution lorsque l'état dans Modèle de démarrage ou d'arrêt la section est Courir.



Étape 9 : Analyser une image avec votre modèle

Vous analysez une image en appelant le [DetectCustomLabels](#) API. Au cours de cette étape, vous utilisez `detect-custom-labels` AWS Command Line Interface (AWS CLI) pour analyser un exemple d'image. Vous obtenez la commande AWS CLI depuis la console Amazon Rekognition Custom Labels. La console configure la commande AWS CLI pour utiliser votre modèle. Il vous suffit de fournir une image stockée dans un compartiment Amazon S3.

Note

La console fournit également des exemples de code Python.

La sortie de `detect-custom-labels` inclut une liste des étiquettes présentes dans l'image, des cadres de délimitation (si le modèle trouve des emplacements d'objets) et indique le niveau de confiance du modèle dans la précision des prévisions.

Pour plus d'informations, veuillez consulter [Analyse d'une image avec un modèle entraîné](#).

Pour analyser une image (console)

1. Si ce n'est pas déjà fait, configurez le AWS CLI. Pour des instructions, consultez [the section called "Étape 4 : Configurer AWS CLI et les kits AWS SDK"](#).
2. Choisissez le Modèle d'utilisation onglet, puis choisissez Code d'API.

The screenshot displays the AWS Rekognition console interface for a custom model named 'rooms_19'. At the top, there are navigation tabs: 'Evaluate', 'Model details', 'Use Model' (highlighted with a red circle), and 'Tags'. A 'Delete model' button is located in the top right corner. Below the tabs, the 'Start or stop model' section features a 'Stop' button and a status indicator showing 'Running' with a green checkmark. A message states: 'Your model is running. Choose Stop model to stop custom label detection or change the number of inference units that your model uses. Restart your model when you need it.' To the right, there is a section for 'Select number of Inference units' with a dropdown menu currently set to '1 inference unit'. Below this, the 'Use your model' section contains an empty text input field for the 'Amazon Resource Name (ARN)'. At the bottom of this section, a link labeled 'API Code' is highlighted with a red circle.

3. Choisissez Commande de l'interface de ligne de commande AWS.
4. Dans leAnalyser l'imagesection, copiez leAWS CLIcommande qui appelle `detect-custom-labels`.

Use your model

Amazon Resource Name (ARN)

▼ API Code

Use your model rooms_ [] by calling the following AWS CLI commands or Python scripts. You can start and stop the model, and analyze custom labels in new images.

AWS CLI command

Python

Start model
Command used to start the rooms_ [] model.

```
1 aws rekognition start-project-version \  
2 --project-version-arn "arn:aws:rekognition:us-east-1:[ ]" \  
3 --min-inference-units 1 \  
4 --region us-east-1
```

Analyze image
Command used to use analyze an image with the rooms_ [] model. Replace MY_BUCKET and PATH_TO_MY_IMAGE with your S3 bucket name and image path.

```
1 aws rekognition detect-custom-labels \  
2 --project-version-arn "arn:aws:rekognition:us-east-1:[ ]" \  
3 --image '{"S3Object": {"Bucket": "MY_BUCKET", "Name": "PATH_TO_MY_IMAG" \  
4 --region us-east-1
```

5. Chargez une image dans un compartiment Amazon S3. Pour obtenir des instructions, voir [Chargement d'objets dans Amazon S3](#) dans le Guide de l'utilisateur d'Amazon Simple Storage Service. Si vous utilisez des images du projet Rooms, utilisez l'une des images que vous avez déplacées vers un dossier distinct dans [Étape 1 : Collectez vos images](#).
6. À l'invite de commande, entrez l'AWS CLI commande que vous avez copiée à l'étape précédente. Cela doit ressembler à l'exemple suivant.

La valeur de `--project-version-arn` doit être le nom de ressource Amazon (ARN) de votre modèle. La valeur de `--region` devrait être le AWS Région dans laquelle vous avez créé le modèle.

Changez `MY_BUCKET` et `PATH_TO_MY_IMAGE` au compartiment Amazon S3 et à l'image que vous avez utilisés à l'étape précédente.

Si vous utilisez [custom-labels-access](#) profil pour obtenir des informations d'identification, ajoutez le `--profile custom-labels-access` paramètre.

```
aws rekognition detect-custom-labels \
  --project-version-arn "model_arn" \
  --image '{"S3Object": {"Bucket": "MY_BUCKET", "Name": "PATH_TO_MY_IMAGE"}}' \
  --region us-east-1 \
  --profile custom-labels-access
```

La sortie JSON de l'AWS CLI La commande doit ressembler à la suivante. `Name` est le nom de l'étiquette au niveau de l'image trouvée par le modèle. `Confidence` (0-100) est la confiance du modèle dans l'exactitude de la prédiction.

```
{
  "CustomLabels": [
    {
      "Name": "living_space",
      "Confidence": 83.41299819946289
    }
  ]
}
```

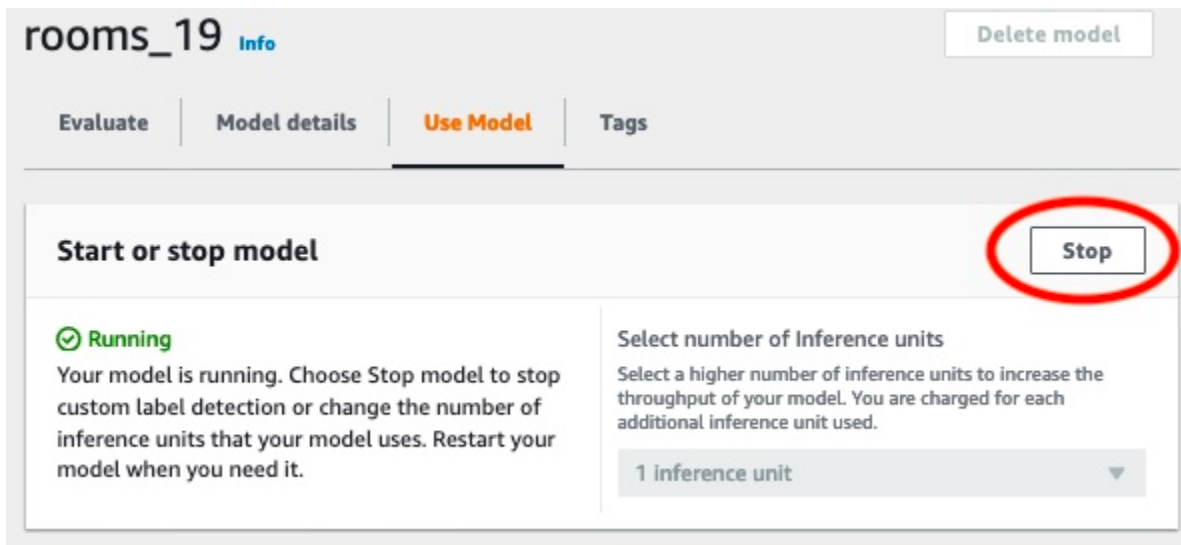
7. Continuez à utiliser le modèle pour analyser d'autres images. Arrêtez le modèle si vous ne l'utilisez plus.

Étape 10 : arrêtez votre modèle

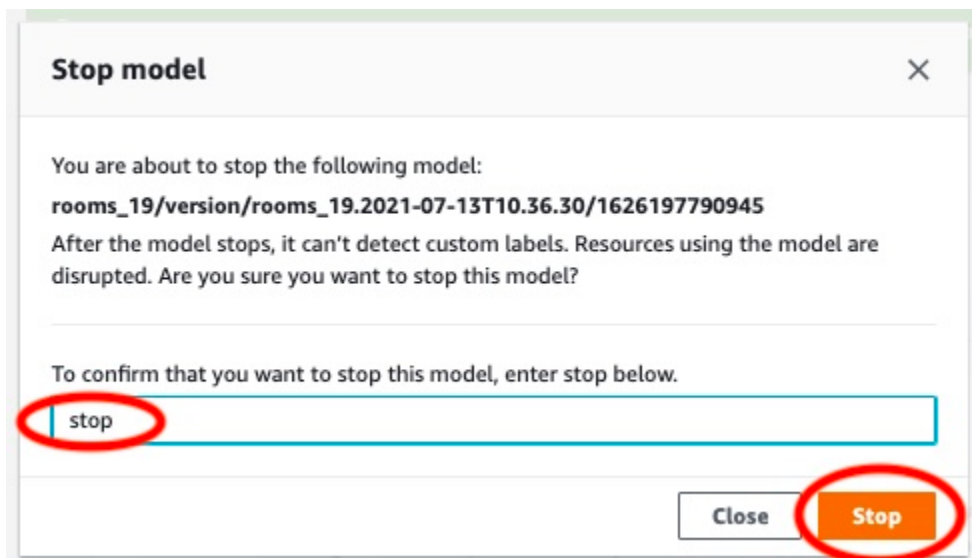
Au cours de cette étape, vous arrêtez d'exécuter votre modèle. Vous êtes facturé en fonction de la durée de fonctionnement de votre modèle. Si vous avez fini d'utiliser le modèle, vous devez l'arrêter.

Pour arrêter votre modèle

1. Dans le `Modèle de démarrage ou d'arrêt` sélection de la section `Arrête`.



2. Dans le Modèle Stop boîte de dialogue, entrez arrête pour confirmer que vous souhaitez arrêter le modèle.



3. Choisissez Arrête pour arrêter votre modèle. Le modèle s'est arrêté lorsque l'état indiqué dans Modèle de démarrage ou d'arrêt la section est Arrêté.

rooms_19 Info
Delete model

Evaluate
Model details
Use Model
Tags

Start or stop model

⊖ Stopped

Your model isn't running. To start running your model, choose Start model or use the example code in Use your model. You can then use your model to find custom labels in images.

Select number of Inference units

Select a higher number of inference units to increase the throughput of your model. You are charged for each additional inference unit used.

1 inference unit
▼

Start

Création d'un modèle d'étiquettes personnalisées Amazon Rekognition

Un modèle est le logiciel que vous entraînez pour trouver les concepts, les scènes et les objets propres à votre entreprise. Vous pouvez créer un modèle à l'aide de la console Amazon Rekognition Custom Labels ou du AWS SDK. Avant de créer un modèle d'étiquettes personnalisées Amazon Rekognition, nous vous recommandons de le lire [Comprendre les étiquettes personnalisées Amazon Rekognition](#).

Cette section fournit des informations sur la console et le SDK sur la création d'un projet, la création de jeux de données d'entraînement et de test pour différents types de modèles et l'entraînement d'un modèle. Les sections suivantes montrent comment améliorer et utiliser votre modèle. Pour obtenir un didacticiel qui vous montre comment créer et utiliser un modèle avec la console, consultez [Tutoriel : Classification d'images](#).

Rubriques

- [Crée un projet](#).
- [Création de jeux de données d'entraînement et de test](#)
- [Entraînement d'un modèle Amazon Rekognition Custom Labels Custom Labels](#)
- [Déboguer un modèle d'entraînement ayant échoué](#)

Crée un projet.

Un projet gère les versions du modèle, le jeu de données d'entraînement et le jeu de données de test d'un modèle. Vous pouvez créer un projet avec la console Amazon Rekognition Custom Labels ou avec l'API. Pour les autres tâches du projet, telles que la suppression d'un projet, consultez [Gestion d'un projet Amazon Rekognition Custom Labels](#).

Création d'un projet Amazon Rekognition Custom Labels (console)

Vous pouvez utiliser la console Amazon Rekognition Custom Labels pour créer un projet. La première fois que vous utilisez la console dans une nouvelle AWS région, Amazon Rekognition Custom Labels vous demande de créer un compartiment Amazon S3 (compartiment de console) dans votre AWS compte. Le bucket est utilisé pour stocker les fichiers de projet. Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels si le compartiment de console n'est pas créé.

Vous pouvez utiliser la console Amazon Rekognition Custom Labels pour créer un projet.

Pour créer un projet (console)

1. Connectez-vous à l'AWS Management Console et ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
2. Dans le volet de gauche, choisissez Use Custom Labels. La page d'accueil des étiquettes personnalisées Amazon Rekognition s'affiche.
3. Sur la page d'accueil d'Amazon Rekognition Custom Labels, choisissez Commencer.
4. Dans le volet de gauche, choisissez Projects.
5. Choisissez Create Project (Créer un projet).
6. Dans Project name (Nom de projet), saisissez un nom pour votre projet.
7. Choisissez Créer un projet pour créer votre projet.
8. Suivez les étapes décrites [Création de jeux de données d'entraînement et de test](#) pour créer les ensembles de données de formation et de test pour votre projet.

Création d'un projet Amazon Rekognition Custom Labels (SDK)

Vous créez un projet Amazon Rekognition Custom Labels en appelant [CreateProject](#). La réponse est un Amazon Resource Name (ARN) qui identifie le projet. Après avoir créé un projet, vous créez des ensembles de données pour la formation et le test d'un modèle. Pour plus d'informations, veuillez consulter [Création de jeux de données d'entraînement et de test avec des images](#).

Pour créer un projet (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS CLI et les AWS SDK. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez le code suivant pour créer un projet.

AWS CLI

L'exemple suivant crée un projet et affiche son ARN.

Remplacez la valeur `project-name` de par le nom du projet que vous souhaitez créer.

```
aws rekognition create-project --project-name my_project \  
--profile custom-labels-access
```

Python

L'exemple suivant crée un projet et affiche son ARN. Fournissez les arguments de ligne de commande suivants :

- `project_name`— le nom du projet que vous souhaitez créer.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

import argparse
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_project(rek_client, project_name):
    """
    Creates an Amazon Rekognition Custom Labels project
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: A name for the new project.
    """

    try:
        #Create the project.
        logger.info("Creating project: %s",project_name)

        response=rek_client.create_project(ProjectName=project_name)

        logger.info("project ARN: %s",response['ProjectArn'])

        return response['ProjectArn']

    except ClientError as err:
        logger.exception("Couldn't create project - %s: %s", project_name,
err.response['Error']['Message'])
        raise

def add_arguments(parser):
```

```
"""
Adds command line arguments to the parser.
:param parser: The command line parser.
"""

parser.add_argument(
    "project_name", help="A name for the new project."
)

def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Creating project: {args.project_name}")

        # Create the project.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        project_arn=create_project(rekognition_client,
            args.project_name)

        print(f"Finished creating project: {args.project_name}")
        print(f"ARN: {project_arn}")

    except ClientError as err:
        logger.exception("Problem creating project: %s", err)
        print(f"Problem creating project: {err}")

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

Java V2

L'exemple suivant crée un projet et affiche son ARN.

Fournissez l'argument de ligne de commande suivant :

- `project_name`— le nom du projet que vous souhaitez créer.

```
/*
 Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

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.CreateProjectRequest;
import software.amazon.awssdk.services.rekognition.model.CreateProjectResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateProject {

    public static final Logger logger =
        Logger.getLogger(CreateProject.class.getName());

    public static String createMyProject(RekognitionClient rekClient, String
        projectName) {

        try {

            logger.log(Level.INFO, "Creating project: {0}", projectName);
            CreateProjectRequest createProjectRequest =
                CreateProjectRequest.builder().projectName(projectName).build();

            CreateProjectResponse response =
                rekClient.createProject(createProjectRequest);

            logger.log(Level.INFO, "Project ARN: {0} ", response.projectArn());
```

```
        return response.projectArn();

    } catch (RekognitionException e) {
        logger.log(Level.SEVERE, "Could not create project: {0}",
e.getMessage());
        throw e;
    }

}

public static void main(String[] args) {

    final String USAGE = "\n" + "Usage: " + "<project_name> <bucket> <image>
\n\n" + "Where:\n"
        + "    project_name - A name for the new project\n\n";

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

    String projectName = args[0];
    String projectArn = null;
    ;

    try {

        // Get the Rekognition client.
        RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

        // Create the project
        projectArn = createMyProject(rekClient, projectName);

        System.out.println(String.format("Created project: %s \nProject ARN:
%s", projectName, projectArn));

        rekClient.close();

    } catch (RekognitionException rekError) {
```



```
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        System.exit(1);
    }

}

}
```

3. Notez le nom de l'ARN du projet qui est affiché dans la réponse. Vous en aurez besoin pour créer un modèle.
4. Suivez les étapes décrites [Création de jeux de données d'entraînement et de test \(kit SDK\)](#) pour créer les ensembles de données de formation et de test pour votre projet.

Création de jeux de données d'entraînement et de test

Un jeu de données est un ensemble d'images et d'étiquettes qui décrivent ces images. Un projet nécessite un jeu de données d'entraînement et un jeu de données de test. Étiquettes personnalisées Amazon Rekognition utilise le jeu de données d'entraînement pour entraîner le modèle. Après l'entraînement, Étiquettes personnalisées Amazon Rekognition utilise le jeu de données de test pour vérifier dans quelle mesure le modèle entraîné prédit les étiquettes correctes.

Vous pouvez créer des jeux de données à l'aide de la console Étiquettes personnalisées Amazon Rekognition ou du kit AWS SDK. Avant de créer un jeu de données, nous vous recommandons de lire [Comprendre les étiquettes personnalisées Amazon Rekognition](#). Pour les autres tâches liées aux jeux de données, consultez [Gestion des jeux de données](#).

Voici les étapes de création de jeux de données d'entraînement et de test pour un projet :

Pour créer des jeux de données d'entraînement et de test pour votre projet

1. Déterminez comment vous devez étiqueter vos jeux de données d'entraînement et de test. Pour plus d'informations, consultez [Utilisation des jeux de données](#).
2. Collectez les images correspondant aux jeux de données d'entraînement et de test. Pour plus d'informations, consultez [the section called "Préparation des images"](#).
3. Créez les jeux de données d'entraînement et de test. Pour plus d'informations, consultez [Création de jeux de données d'entraînement et de test avec des images](#). Si vous utilisez le kit AWS SDK, consultez [Création de jeux de données d'entraînement et de test \(kit SDK\)](#).

4. Si nécessaire, ajoutez des étiquettes ou des cadres de délimitation au niveau de l'image aux images de votre jeu de données. Pour plus d'informations, consultez [Étiquetage des images](#).

Après avoir créé les jeux de données, vous pouvez [entraîner](#) le modèle.

Rubriques

- [Utilisation des jeux de données](#)
- [Préparation des images](#)
- [Création de jeux de données d'entraînement et de test avec des images](#)
- [Étiquetage des images](#)
- [Débogage des jeux de données](#)

Utilisation des jeux de données

La façon dont vous étiquetez les jeux de données d'entraînement et de test de votre projet détermine le type de modèle que vous créez. Avec Étiquettes personnalisées Amazon Rekognition, vous pouvez créer des modèles présentant les caractéristiques suivantes.

- [Recherche d'objets, de scènes et de concepts](#)
- [Recherche de l'emplacement d'objets](#)
- [Recherche de l'emplacement d'une marque](#)

Recherche d'objets, de scènes et de concepts

Le modèle classe les objets, les scènes et les concepts associés à une image complète.

Vous pouvez créer deux types de modèles de classification : la classification d'images et la classification à plusieurs étiquettes. Pour les deux types de modèles de classification, le modèle trouve une ou plusieurs étiquettes correspondantes parmi l'ensemble complet d'étiquettes utilisées pour l'entraînement. Les jeux de données d'entraînement et de test nécessitent tous au moins deux étiquettes.

Image classification

Le modèle classe les images comme appartenant à un ensemble d'étiquettes prédéfinies.

Supposons, par exemple, que vous ayez besoin d'un modèle qui détermine si une image contient un espace de vie. L'image suivante peut avoir une étiquette `living_space` au niveau de l'image.



Pour ce type de modèle, ajoutez une seule étiquette au niveau de l'image à chacune des images des jeux de données d'entraînement et de test. Pour un exemple de projet, consultez [Image classification](#).

Classification à plusieurs étiquettes

Le modèle classe les images dans plusieurs catégories, telles que le type de fleur et le fait qu'elle possède ou non des feuilles. Par exemple, l'image suivante peut comporter les étiquettes au niveau de l'image `euphorbe_méditerranéenne` et `sans_feuilles`.



Pour ce type de modèle, attribuez des étiquettes au niveau de l'image pour chaque catégorie aux images des jeux de données d'entraînement et de test. Pour un exemple de projet, consultez [Classification d'images à étiquettes multiples](#).

Attribution d'étiquettes au niveau de l'image

Si les images sont stockées dans un compartiment Amazon S3, vous pouvez utiliser les [noms de dossier](#) pour ajouter automatiquement des étiquettes au niveau de l'image. Pour plus d'informations, consultez [Compartiment Amazon S3](#). Vous pouvez également ajouter des étiquettes au niveau de l'image aux images après avoir créé un jeu de données. Pour plus d'informations, consultez [the section called "Attribution d'étiquettes au niveau de l'image à une image"](#). Vous pouvez ajouter de nouvelles étiquettes à votre convenance. Pour plus d'informations, consultez [Gestion des étiquettes](#).

Recherche de l'emplacement d'objets

Pour créer un modèle qui prédit l'emplacement des objets dans vos images, vous devez définir des cadres de délimitation et des étiquettes pour les images dans les jeux de données d'entraînement et de test. Un cadre de délimitation entoure étroitement un objet. Par exemple, l'image suivante affiche un cadre de délimitation autour d'un appareil Amazon Echo et d'un appareil Amazon Echo Dot. Une étiquette est attribuée à chaque cadre de délimitation (Amazon Echo ou Amazon Echo Dot).



Pour trouver l'emplacement d'un objet, les jeux de données doivent avoir au moins une étiquette. Pendant l'entraînement du modèle, une autre étiquette est automatiquement créée pour représenter la zone située en dehors des cadres de délimitation sur une image.

Attribution de cadres de délimitation

Lorsque vous créez le jeu de données, vous pouvez inclure des informations relatives aux cadres de délimitation pour vos images. Par exemple, vous pouvez importer un [fichier manifeste](#) au format SageMaker Ground Truth contenant des cadres de délimitation. Vous pouvez également ajouter des cadres de délimitation après avoir créé un jeu de données. Pour plus d'informations, consultez [Étiquetage des objets à l'aide de cadres de délimitation](#). Vous pouvez ajouter de nouvelles étiquettes à votre convenance. Pour plus d'informations, consultez [Gestion des étiquettes](#).

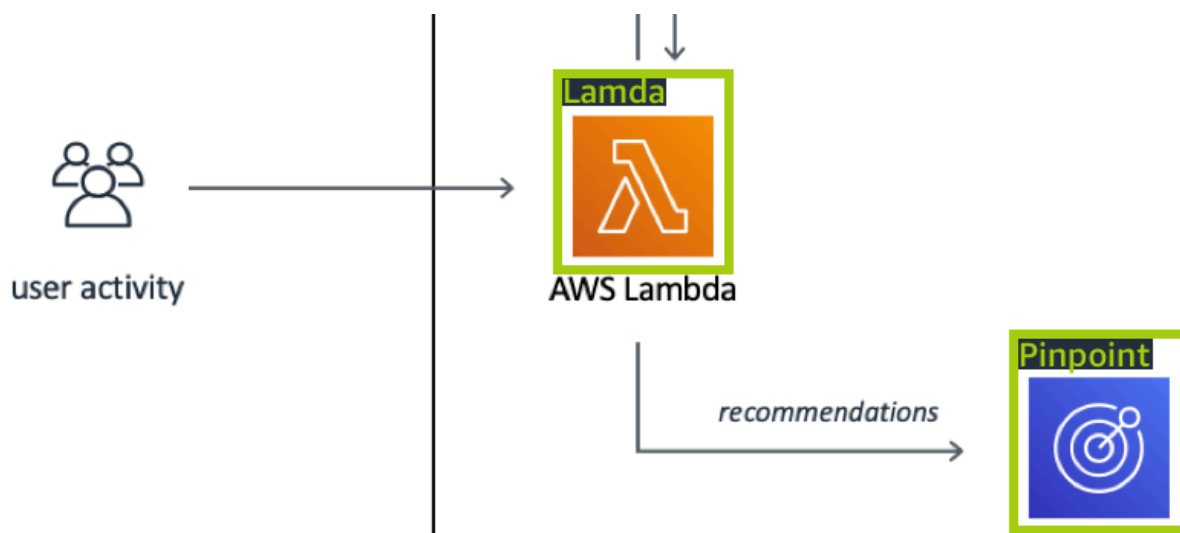
Recherche de l'emplacement d'une marque

Si vous souhaitez trouver l'emplacement d'une marque, telle qu'un logo et un personnage animé, vous pouvez utiliser deux types d'images différents pour les images du jeu de données d'entraînement.

- Images représentant uniquement le logo. Chaque image a besoin d'une seule étiquette au niveau de l'image, qui représente le nom du logo. Par exemple, l'étiquette au niveau de l'image pour l'image suivante pourrait être Lambda.



- Images contenant le logo à des emplacements naturels, comme un match de football ou un schéma architectural. Chaque image d'entraînement nécessite des cadres de délimitation qui entourent chaque instance du logo. Par exemple, l'image suivante présente un schéma architectural avec des cadres de délimitation étiquetés entourant les logos AWS Lambda et Amazon Pinpoint.



Nous vous recommandons de ne pas combiner des étiquettes au niveau de l'image et des cadres de délimitation dans les images d'entraînement.

Les images de test nécessitent des cadres de délimitation qui entourent les instances de la marque que vous souhaitez trouver. Vous pouvez fractionner le jeu de données d'entraînement pour créer le jeu de données de test, uniquement si les images d'entraînement incluent des cadres de délimitation étiquetés. Si les images d'entraînement ne comportent que des étiquettes au niveau de l'image, vous devrez créer un jeu de données de test comprenant des images avec des cadres de délimitation étiquetés. Si vous entraînez un modèle pour qu'il trouve l'emplacement d'une marque, effectuez les actions indiquées dans [Étiquetage des objets à l'aide de cadres de délimitation](#) et [Attribution d'étiquettes au niveau de l'image à une image](#) en fonction de la façon dont vous étiquetez vos images.

L'exemple de projet [Détection de marque](#) montre comment Étiquettes personnalisées Amazon Rekognition utilise des cadres de délimitation étiquetés pour entraîner un modèle à rechercher l'emplacement d'objets.

Exigences d'étiquetage pour les types de modèles

Utilisez le tableau suivant pour déterminer comment étiqueter vos images.

Vous pouvez combiner des étiquettes au niveau de l'image et des images étiquetées dans des cadres de délimitation dans un seul jeu de données. Dans ce cas, Étiquettes personnalisées Amazon Rekognition choisit de créer un modèle au niveau de l'image ou un modèle de localisation des objets.

Exemple	Images d'entraînement	Images de test
Image classification	1 étiquette au niveau de l'image par image	1 étiquette au niveau de l'image par image
Classification à plusieurs étiquettes	Plusieurs étiquettes au niveau de l'image par image	Plusieurs étiquettes au niveau de l'image par image
Recherche de l'emplacement d'une marque	étiquettes au niveau de l'image (vous pouvez également utiliser des cadres de délimitation étiquetés)	Cadres de délimitation étiquetés

Exemple	Images d'entraînement	Images de test
Recherche de l'emplacement d'objets	Cadres de délimitation étiquetés	Cadres de délimitation étiquetés

Préparation des images

Les images de vos jeux de données d'entraînement et de test contiennent les objets, les scènes ou les concepts que vous souhaitez que votre modèle trouve.

Leur contenu devrait se composer d'une variété d'arrière-plans et d'éclairages représentant les images que vous souhaitez que le modèle entraîné identifie.

Cette section fournit des informations sur les images de vos jeux de données d'entraînement et de test.

Format d'image

Vous pouvez entraîner les modèles Étiquettes personnalisées Amazon Rekognition avec des images au format PNG et JPEG. De même, pour détecter les étiquettes personnalisées à l'aide de `DetectCustomLabels`, vous avez besoin d'images au format PNG et JPEG.

Recommandations relatives aux images d'entrée

Étiquettes personnalisées Amazon Rekognition nécessite des images pour entraîner et tester le modèle. Pour préparer vos images, tenez compte des points suivants :

- Choisissez un domaine spécifique pour le modèle que vous souhaitez créer. Par exemple, vous pouvez choisir un modèle pour les vues panoramiques et un autre modèle pour les objets tels que les pièces d'une machine. Étiquettes personnalisées Amazon Rekognition fonctionne mieux si vos images se trouvent dans le domaine choisi.
- Utilisez au moins 10 images pour entraîner le modèle.
- Les images doivent être au format PNG ou JPEG.
- Utilisez des images qui montrent l'objet sous différents éclairages, arrière-plans et résolutions.
- Les images d'entraînement et de test doivent être similaires aux images avec lesquelles vous souhaitez utiliser le modèle.

- Décidez quelles étiquettes attribuer aux images.
- Assurez-vous que les images sont suffisamment grandes en termes de résolution. Pour plus d'informations, consultez [Amazon Rekognition](#).
- Assurez-vous que les occlusions ne masquent pas les objets que vous souhaitez détecter.
- Utilisez des images dont le contraste est suffisant avec l'arrière-plan.
- Utilisez des images lumineuses et nettes. Évitez autant que possible d'utiliser des images qui peuvent être floues en raison du mouvement du sujet et de l'appareil photo.
- Utilisez une image sur laquelle l'objet occupe une grande partie de l'espace.
- Les images du jeu de données de test ne doivent pas être les images du jeu de données d'entraînement. Elles doivent comporter les objets, les scènes et les concepts que le modèle sera entraîné à analyser.

Taille des images

Étiquettes personnalisées Amazon Rekognition utilise un ensemble d'images pour entraîner un modèle. Vous devez utiliser au moins 10 images pour l'entraînement. Étiquettes personnalisées Amazon Rekognition stocke les images d'entraînement et de test dans des jeux de données. Pour plus d'informations, consultez [Création de jeux de données d'entraînement et de test avec des images](#).

Création de jeux de données d'entraînement et de test avec des images

Vous pouvez commencer par un projet avec un seul jeu de données ou avec un jeu de données d'entraînement et un jeu de données de test distincts. Si vous commencez avec un seul jeu de données, Étiquettes personnalisées Amazon Rekognition fractionne le jeu de données pendant l'entraînement afin de créer un jeu de données d'entraînement (80 %) et un jeu de données de test (20 %) pour votre projet. Commencez par un seul jeu de données si vous souhaitez qu'Étiquettes personnalisées Amazon Rekognition détermine où les images sont utilisées pour l'entraînement et les tests. Pour un contrôle complet de l'entraînement, du test et du réglage des performances, nous vous recommandons de démarrer votre projet avec des jeux de données d'entraînement et de test distincts.

Pour créer des jeux de données d'entraînement et de test pour un projet, vous pouvez importer des images depuis l'un des emplacements suivants :

- [Compartiment Amazon S3](#)

- [Ordinateur local](#)
- [Fichier manifeste](#)
- [Jeu de données](#)

Si vous démarrez votre projet avec un jeu de données d'entraînement et un jeu de données de test distincts, vous pouvez utiliser des emplacements source différents pour chacun d'eux.

Selon leur provenance, vos images peuvent ne pas être étiquetées. Par exemple, les images importées à partir d'un ordinateur local ne sont pas étiquetées. Les images importées depuis un fichier manifeste Amazon SageMaker Ground Truth sont étiquetées. Vous pouvez utiliser la console Étiquettes personnalisées Amazon Rekognition pour ajouter, modifier et attribuer des étiquettes. Pour plus d'informations, consultez [Étiquetage des images](#).

Si les images sont chargées avec des erreurs, si des images sont manquantes ou si des étiquettes sont absentes des images, lisez [Déboguer un modèle d'entraînement ayant échoué](#).

Pour plus d'informations sur les jeux de données, consultez [Gestion des jeux de données](#).

Création de jeux de données d'entraînement et de test (kit SDK)

Vous pouvez utiliser le kit AWS SDK pour créer des jeux de données d'entraînement et de test.

Jeu de données d'entraînement

Vous pouvez utiliser le kit AWS SDK pour créer un jeu de données d'entraînement de la manière suivante.

- [CreateDataset](#) À utiliser avec un fichier manifeste au format Amazon Sagemaker que vous fournissez. Pour plus d'informations, consultez [the section called “Création d'un fichier manifeste”](#). Pour obtenir un exemple de code, consultez [Création d'un jeu de données à l'aide d'un fichier manifeste \(SDK\) SageMaker Ground Truth](#).
- Utilisez `CreateDataset` pour copier un jeu de données Étiquettes personnalisées Amazon Rekognition existant. Pour obtenir un exemple de code, consultez [Création d'un jeu de données à l'aide d'un jeu de données existant \(SDK\)](#).
- Créez un jeu de données vide avec `CreateDataset`, puis ajoutez les entrées du jeu de données ultérieurement avec [UpdateDatasetEntries](#). Pour créer un jeu de données vide, consultez [Ajouter un jeu de données à un projet](#). Pour ajouter des images à un jeu de données, consultez [Ajouter](#)

[d'autres images \(SDK\)](#). Vous devez ajouter les entrées du jeu de données avant de pouvoir entraîner un modèle.

Jeu de données de test

Vous pouvez utiliser le kit AWS SDK pour créer un jeu de données de test de la manière suivante :

- [CreateDataset](#) À utiliser avec un fichier manifeste au format Amazon Sagemaker que vous fournissez. Pour plus d'informations, consultez [the section called "Création d'un fichier manifeste"](#). Pour obtenir un exemple de code, consultez [Création d'un jeu de données à l'aide d'un fichier manifeste \(SDK\) SageMaker Ground Truth](#).
- Utilisez `CreateDataset` pour copier un jeu de données Étiquettes personnalisées Amazon Rekognition existant. Pour obtenir un exemple de code, consultez [Création d'un jeu de données à l'aide d'un jeu de données existant \(SDK\)](#).
- Créez un jeu de données vide avec `CreateDataset`, puis ajoutez les entrées du jeu de données ultérieurement avec `UpdateDatasetEntries`. Pour créer un jeu de données vide, consultez [Ajouter un jeu de données à un projet](#). Pour ajouter des images à un jeu de données, consultez [Ajouter d'autres images \(SDK\)](#). Vous devez ajouter les entrées du jeu de données avant de pouvoir entraîner un modèle.
- Fractionnez le jeu de données d'entraînement en un jeu de données d'entraînement et un jeu de données de test distincts. Créez d'abord un jeu de données de test vide avec `CreateDataset`. Déplacez ensuite 20 % des entrées du jeu de données d'entraînement dans le jeu de données de test en appelant [DistributeDatasetEntries](#). Pour créer un jeu de données vide, consultez [Ajouter un jeu de données à un projet \(SDK\)](#). Pour fractionner le jeu de données d'entraînement, consultez [Distribution d'un jeu de données d'entraînement \(SDK\)](#).

Compartiment Amazon S3

Les images sont stockées dans un compartiment Amazon S3. Vous pouvez utiliser le compartiment de la console ou un autre compartiment Amazon S3 de votre compte AWS. Si vous utilisez le compartiment de la console, les autorisations requises sont déjà configurées. Si vous n'utilisez pas le compartiment de la console, consultez [Accès à des compartiments Amazon S3 externes](#).

Note

Vous ne pouvez pas utiliser le kit AWS SDK pour créer un jeu de données directement à partir d'images d'un compartiment Amazon S3. Créez plutôt un fichier manifeste faisant

référence aux emplacements sources des images. Pour de plus amples informations, veuillez consulter la page [Fichier manifeste](#).

Lors de la création du jeu de données, vous pouvez choisir d'attribuer des noms d'étiquette aux images en fonction du nom du dossier contenant les images. Le ou les dossiers doivent être un élément enfant du chemin du dossier Amazon S3 que vous spécifiez dans l'emplacement du dossier S3 lors de la création du jeu de données. Pour créer un jeu de données, consultez [Création d'un jeu de données en important des images depuis un compartiment S3](#).

Par exemple, supposons qu'un compartiment Amazon S3 ait la structure de dossiers suivante. Si vous spécifiez l'emplacement du dossier Amazon S3 comme étant S3-bucket/alexa-devices, l'étiquette echo est attribuée aux images du dossier echo. De même, l'étiquette echo-dot est attribuée aux images du dossier echo-dot. Le nom des dossiers enfants situés plus loin dans la structure de dossiers n'est pas utilisé pour étiqueter les images. Au lieu de cela, le dossier enfant approprié correspondant à l'emplacement du dossier Amazon S3 est utilisé. Par exemple, le label echo-dot white-echo-dot est attribué aux images du dossier. Aucune étiquette n'est attribuée aux images situées au niveau de l'emplacement du dossier S3 (alexa-devices).

Les dossiers situés plus loin dans la structure de dossiers peuvent être utilisés pour étiqueter des images en spécifiant un emplacement de dossier S3 plus profond. Par exemple, si vous spécifiez S3-Bucket/Alexa-Devices/Echo-Dot, les images du dossier sont étiquetées. white-echo-dotwhite-echo-dot Les images situées en dehors de l'emplacement du dossier S3 spécifié, comme echo, ne sont pas importées.

```
S3-bucket
### alexa-devices
  ### echo
  #   ### echo-image-1.png
  #   ### echo-image-2.png
  #   ### .
  #   ### .
  ### echo-dot
    ### white-echo-dot
    #   ### white-echo-dot-image-1.png
    #   ### white-echo-dot-image-2.png
    #
    ### echo-dot-image-1.png
    ### echo-dot-image-2.png
    ### .
```

```
### .
```

Nous vous recommandons d'utiliser le compartiment Amazon S3 (compartiment de la console) créé pour vous par Amazon Rekognition lorsque vous avez ouvert la console pour la première fois dans la région AWS actuelle. Si le compartiment Amazon S3 que vous utilisez est différent du compartiment de la console (c'est-à-dire s'il est externe), la console vous invite à configurer les autorisations appropriées lors de la création du jeu de données. Pour plus d'informations, consultez [the section called "Étape 2 : Configurer les autorisations de la console"](#).

Création d'un jeu de données en important des images depuis un compartiment S3

La procédure suivante vous explique comment créer un jeu de données à l'aide d'images stockées dans le compartiment de la console S3. Les images sont automatiquement étiquetées avec le nom du dossier dans lequel elles sont stockées.

Après avoir importé vos images, vous pouvez ajouter d'autres images, attribuer des étiquettes et ajouter des cadres de délimitation à partir de la page de galerie d'un jeu de données. Pour plus d'informations, consultez [Étiquetage des images](#).

Chargement des images dans un compartiment Amazon Simple Storage Service

1. Créez un dossier dans votre système de fichiers local. Utilisez un nom de dossier tel qu'`alexa-devices`.
2. Dans le dossier que vous venez de créer, ajoutez des dossiers portant le nom de chaque étiquette que vous souhaitez utiliser (par exemple, `echo` et `echo-point`). La structure de dossiers devrait ressembler à ce qui suit.

```
alexa-devices
### echo
#   ### echo-image-1.png
#   ### echo-image-2.png
#   ### .
#   ### .
### echo-dot
### echo-dot-image-1.png
### echo-dot-image-2.png
### .
### .
```

3. Placez les images correspondant à une étiquette dans le dossier portant le même nom d'étiquette.
4. Connectez-vous à la AWS Management Console et ouvrez la console Amazon S3 à l'adresse <https://console.aws.amazon.com/s3/>.
5. [Ajoutez le dossier](#) que vous avez créé à l'étape 1 au compartiment Amazon S3 (compartiment de la console) créé pour vous par Étiquettes personnalisées Amazon Rekognition lors de la première configuration. Pour plus d'informations, consultez [Gestion d'un projet Amazon Rekognition Custom Labels](#).
6. Ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
7. Choisissez Utiliser Custom Labels.
8. Choisissez Démarrer.
9. Dans le volet de navigation de gauche, choisissez Projets.
10. Sur la page Projets, choisissez le projet auquel vous souhaitez ajouter un jeu de données. La page de détails de ce projet s'affiche.
11. Choisissez Créer un jeu de données. La page Créer un jeu de données s'affiche.
12. Dans Démarrage de la configuration, choisissez Démarrer avec un seul jeu de données ou Démarrer avec un jeu de données d'entraînement. Pour créer un modèle de meilleure qualité, nous vous recommandons de commencer par un jeu de données d'entraînement et un jeu de données de test distincts.

Single dataset

- a. Dans la section Détails du jeu de données d'entraînement, choisissez Importer des images depuis le compartiment S3.
- b. Dans la section Détails du jeu de données d'entraînement, entrez les informations relatives aux étapes 13 à 15 de la section Configuration de la source d'image.

Separate training and test datasets

- a. Dans la section Détails du jeu de données d'entraînement, choisissez Importer des images depuis le compartiment S3.
- b. Dans la section Détails du jeu de données d'entraînement, entrez les informations relatives aux étapes 13 à 15 de la section Configuration de la source d'image.

- c. Dans la section Détails du jeu de données de test, choisissez Importer des images depuis le compartiment S3.
 - d. Dans la section Détails du jeu de données de test, entrez les informations relatives aux étapes 13 à 15 de la section Configuration de la source d'image.
13. Choisissez Importer des images depuis le compartiment Amazon S3.
 14. Dans URI S3, entrez l'emplacement du compartiment Amazon S3 et le chemin du dossier.
 15. Choisissez Associer automatiquement des étiquettes aux images en fonction du dossier.
 16. Choisissez Créer des jeux de données. La page des jeux de données de votre projet s'ouvre.
 17. Si vous devez ajouter ou modifier des étiquettes, effectuez les actions indiquées dans [Étiquetage des images](#).
 18. Suivez les étapes décrites dans [Entraînement d'un modèle \(console\)](#) pour entraîner le modèle.

Ordinateur local

Les images sont chargées directement depuis votre ordinateur. Vous pouvez charger jusqu'à 30 images à la fois.

Aucune étiquette n'est associée aux images que vous chargez. Pour plus d'informations, consultez [Étiquetage des images](#). Si vous avez de nombreuses images à charger, pensez à utiliser un compartiment Amazon S3. Pour plus d'informations, consultez [Compartiment Amazon S3](#).

Note

Vous ne pouvez pas utiliser le kit AWS SDK pour créer un jeu de données avec des images locales. Créez plutôt un fichier manifeste et chargez les images dans un compartiment Amazon S3. Pour plus d'informations, consultez [Fichier manifeste](#).

Pour créer un jeu de données à l'aide d'images d'un ordinateur local (console)

1. Ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
2. Choisissez Utiliser Custom Labels.
3. Choisissez Démarrer.
4. Dans le volet de navigation de gauche, choisissez Projets.

5. Sur la page Projets, choisissez le projet auquel vous souhaitez ajouter un jeu de données. La page de détails de ce projet s'affiche.
6. Choisissez Créer un jeu de données. La page Créer un jeu de données s'affiche.
7. Dans Démarrage de la configuration, choisissez Démarrer avec un seul jeu de données ou Démarrer avec un jeu de données d'entraînement. Pour créer un modèle de meilleure qualité, nous vous recommandons de commencer par un jeu de données d'entraînement et un jeu de données de test distincts.

Single dataset

- a. Dans la section Détails du jeu de données d'entraînement, choisissez Charger des images depuis votre ordinateur.
- b. Choisissez Créer un jeu de données.
- c. Sur la page du jeu de données du projet, choisissez Ajouter des images.
- d. Choisissez les images que vous souhaitez charger dans le jeu de données à partir des fichiers de votre ordinateur. Vous pouvez faire glisser les images ou choisir celles que vous souhaitez charger à partir de votre ordinateur local.
- e. Choisissez Charger des images.

Separate training and test datasets

- a. Dans la section Détails du jeu de données d'entraînement, choisissez Charger des images depuis votre ordinateur.
- b. Dans la section Détails du jeu de données de test, choisissez Charger des images depuis votre ordinateur.

Note

Le jeu de données d'entraînement et le jeu de données de test peuvent avoir différentes sources d'images.

- c. Choisissez Créer des jeux de données. La page des jeux de données de votre projet apparaît avec un onglet Entraînement et un onglet Test pour les jeux de données respectifs.
- d. Choisissez Actions, puis sélectionnez Ajouter des images au jeu de données d'entraînement.

- e. Choisissez les images que vous souhaitez charger dans le jeu de données. Vous pouvez faire glisser les images ou choisir celles que vous souhaitez charger à partir de votre ordinateur local.
 - f. Choisissez Charger des images.
 - g. Répétez les étapes 5e à 5g. Pour l'étape 5e, choisissez Actions, puis sélectionnez Ajouter des images au jeu de données de test.
8. Suivez les étapes décrites dans [Étiquetage des images](#) pour étiqueter les images.
 9. Suivez les étapes décrites dans [Entraînement d'un modèle \(console\)](#) pour entraîner le modèle.

Fichier manifeste

Vous pouvez créer un ensemble de données à l'aide d'un fichier manifeste au format Amazon SageMaker Ground Truth. Vous pouvez utiliser le fichier manifeste d'une tâche Amazon SageMaker Ground Truth. Si vos images et vos étiquettes ne sont pas au format d'un fichier manifeste de SageMaker Ground Truth, vous pouvez créer un fichier manifeste de SageMaker format et l'utiliser pour importer vos images étiquetées.

Rubriques

- [Création d'un ensemble de données à l'aide d'un fichier manifeste de SageMaker Ground Truth \(Console\)](#)
- [Création d'un jeu de données à l'aide d'un fichier manifeste \(SDK\) SageMaker Ground Truth](#)
- [Offre d'emploi SageMaker chez Amazon Ground Truth](#)
- [Création d'un fichier manifeste](#)
- [Conversion d'autres formats de jeu de données en fichier manifeste](#)

Création d'un ensemble de données à l'aide d'un fichier manifeste de SageMaker Ground Truth (Console)

La procédure suivante explique comment créer un ensemble de données à l'aide d'un fichier manifeste au format SageMaker Ground Truth.

1. Pour créer un fichier manifeste pour le jeu de données d'entraînement, effectuez l'une des actions suivantes :
 - Créez un fichier manifeste avec un SageMaker GroundTruth Job en suivant les instructions de [Offre d'emploi SageMaker chez Amazon Ground Truth](#).

- Créez votre propre fichier manifeste en suivant les instructions sous [Création d'un fichier manifeste](#).

Si vous souhaitez créer un jeu de données de test, répétez l'étape 1.


2. Ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
3. Choisissez Utiliser Custom Labels.
4. Choisissez Démarrer.
5. Dans le volet de navigation de gauche, choisissez Projets.
6. Sur la page Projets, choisissez le projet auquel vous souhaitez ajouter un jeu de données. La page de détails de ce projet s'affiche.
7. Choisissez Créer un jeu de données. La page Créer un jeu de données s'affiche.
8. Dans Démarrage de la configuration, choisissez Démarrer avec un seul jeu de données ou Démarrer avec un jeu de données d'entraînement. Pour créer un modèle de meilleure qualité, nous vous recommandons de commencer par un jeu de données d'entraînement et un jeu de données de test distincts.

Single dataset

- a. Dans la section des détails du jeu de données d'entraînement, choisissez Importer des images étiquetées par SageMaker Ground Truth.
- b. Dans Emplacement du fichier .manifest, entrez l'emplacement du fichier manifeste que vous avez créé à l'étape 1.
- c. Choisissez Créer un jeu de données. La page des jeux de données de votre projet s'ouvre.

Separate training and test datasets

- a. Dans la section des détails du jeu de données d'entraînement, choisissez Importer des images étiquetées par SageMaker Ground Truth.
- b. Dans Emplacement du fichier .manifest, entrez l'emplacement du fichier manifeste que vous avez créé à l'étape 1 pour le jeu de données d'entraînement.
- c. Dans la section Détails du jeu de données de test, choisissez Importer des images étiquetées par SageMaker Ground Truth.

 Note

Le jeu de données d'entraînement et le jeu de données de test peuvent avoir différentes sources d'images.

- d. Dans Emplacement du fichier .manifest, entrez l'emplacement du fichier manifeste que vous avez créé à l'étape 1 pour le jeu de données de test.
 - e. Choisissez Créer des jeux de données. La page des jeux de données de votre projet s'ouvre.
9. Si vous devez ajouter ou modifier des étiquettes, effectuez les actions indiquées dans [Étiquetage des images](#).
 10. Suivez les étapes décrites dans [Entraînement d'un modèle \(console\)](#) pour entraîner le modèle.

Création d'un jeu de données à l'aide d'un fichier manifeste (SDK) SageMaker Ground Truth

La procédure suivante explique comment créer des ensembles de données d'entraînement ou de test à partir d'un fichier manifeste à l'aide de l'[CreateDatasetAPI](#).

Vous pouvez utiliser un fichier manifeste existant, tel que le résultat d'une [tâche de SageMaker Ground Truth](#), ou créer votre propre [fichier manifeste](#).

1. Si vous ne l'avez pas déjà fait, installez et configurez l'AWS CLI et les kits AWS SDK. Pour plus d'informations, consultez [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Pour créer un fichier manifeste pour le jeu de données d'entraînement, effectuez l'une des actions suivantes :
 - Créez un fichier manifeste avec un SageMaker GroundTruth Job en suivant les instructions de [Offre d'emploi SageMaker chez Amazon Ground Truth](#).
 - Créez votre propre fichier manifeste en suivant les instructions sous [Création d'un fichier manifeste](#).

Si vous souhaitez créer un jeu de données de test, répétez l'étape 2.

3. Utilisez l'exemple de code suivant pour créer le jeu de données d'entraînement et de test.

AWS CLI

Utilisez le code suivant pour créer un jeu de données. Remplacez les éléments suivants :

- `project_arn` : ARN du projet que vous souhaitez ajouter au jeu de données de test.
- `type` : type de jeu de données que vous souhaitez créer (TRAIN pour « entraînement » ou TEST).
- `bucket` : compartiment qui contient le fichier manifeste correspondant au jeu de données.
- `manifest_file` : nom de fichier et chemin d'accès du fichier manifeste.

```
aws rekognition create-dataset --project-arn project_arn \  
  --dataset-type type \  
  --dataset-source '{ "GroundTruthManifest": { "S3Object": { "Bucket": "bucket",  
"Name": "manifest_file" } } }' \  
  --profile custom-labels-access
```

Python

Utilisez les valeurs suivantes pour créer un jeu de données. Fournissez les paramètres de ligne de commande suivants :

- `project_arn` : ARN du projet que vous souhaitez ajouter au jeu de données de test.
- `dataset_type` : type de jeu de données que vous souhaitez créer (train ou test).
- `bucket` : compartiment qui contient le fichier manifeste correspondant au jeu de données.
- `manifest_file` : nom de fichier et chemin d'accès du fichier manifeste.

```
#Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/  
amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-  
SAMPLECODE.)
```

```
import argparse  
import logging  
import time  
import json  
import boto3
```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_dataset(rek_client, project_arn, dataset_type, bucket,
                  manifest_file):
    """
    Creates an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
    dataset.
    :param dataset_type: The type of the dataset that you want to create (train
    or test).
    :param bucket: The S3 bucket that contains the manifest file.
    :param manifest_file: The path and filename of the manifest file.
    """

    try:
        #Create the project
        logger.info("Creating %s dataset for project %s",dataset_type,
                    project_arn)

        dataset_type = dataset_type.upper()

        dataset_source = json.loads(
            '{ "GroundTruthManifest": { "S3Object": { "Bucket": "'
            + bucket
            + '", "Name": "'
            + manifest_file
            + '" } } }'
        )

        response = rek_client.create_dataset(
            ProjectArn=project_arn, DatasetType=dataset_type,
            DatasetSource=dataset_source
        )

        dataset_arn=response['DatasetArn']

        logger.info("dataset ARN: %s",dataset_arn)

        finished=False
        while finished is False:
```

```
dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

status=dataset['DatasetDescription']['Status']

if status == "CREATE_IN_PROGRESS":
    logger.info("Creating dataset: %s ",dataset_arn)
    time.sleep(5)
    continue

if status == "CREATE_COMPLETE":
    logger.info("Dataset created: %s", dataset_arn)
    finished=True
    continue

if status == "CREATE_FAILED":
    error_message = f"Dataset creation failed: {status} :
{dataset_arn}"
    logger.exception(error_message)
    raise Exception (error_message)

    error_message = f"Failed. Unexpected state for dataset creation:
{status} : {dataset_arn}"
    logger.exception(error_message)
    raise Exception(error_message)

return dataset_arn

except ClientError as err:
    logger.exception("Couldn't create dataset: %s",err.response['Error']
['Message'])
    raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
the dataset."
    )
```

```
parser.add_argument(
    "dataset_type", help="The type of the dataset that you want to create
(train or test).")
)

parser.add_argument(
    "bucket", help="The S3 bucket that contains the manifest file."
)

parser.add_argument(
    "manifest_file", help="The path and filename of the manifest file."
)

def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        #Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Creating {args.dataset_type} dataset for project
{args.project_arn}")

        #Create the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        dataset_arn=create_dataset(rekognition_client,
            args.project_arn,
            args.dataset_type,
            args.bucket,
            args.manifest_file)

        print(f"Finished creating dataset: {dataset_arn}")

    except ClientError as err:
        logger.exception("Problem creating dataset: %s", err)
        print(f"Problem creating dataset: {err}")
```

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

Java V2

Utilisez les valeurs suivantes pour créer un jeu de données. Fournissez les paramètres de ligne de commande suivants :

- `project_arn` : ARN du projet que vous souhaitez ajouter au jeu de données de test.
- `dataset_type` : type de jeu de données que vous souhaitez créer (`train` ou `test`).
- `bucket` : compartiment qui contient le fichier manifeste correspondant au jeu de données.
- `manifest_file` : nom de fichier et chemin d'accès du fichier manifeste.

```
/*  
 Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 SPDX-License-Identifier: Apache-2.0  
*/  
  
package com.example.rekognition;  
  
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.CreateDatasetRequest;  
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;  
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;  
import software.amazon.awssdk.services.rekognition.model.DatasetSource;  
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;  
import software.amazon.awssdk.services.rekognition.model.DatasetType;  
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;  
import  
    software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;  
import software.amazon.awssdk.services.rekognition.model.GroundTruthManifest;  
import software.amazon.awssdk.services.rekognition.model.RekognitionException;  
import software.amazon.awssdk.services.rekognition.model.S3Object;  
  
import java.util.logging.Level;
```



```
import java.util.logging.Logger;

public class CreateDatasetManifestFiles {

    public static final Logger logger =
        Logger.getLogger(CreateDatasetManifestFiles.class.getName());

    public static String createMyDataset(RekognitionClient rekClient, String
        projectArn, String datasetType,
        String bucket, String name) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Creating {0} dataset for project : {1} from
                s3://{2}/{3} ",
                new Object[] { datasetType, projectArn, bucket, name });

            DatasetType requestDatasetType = null;

            switch (datasetType) {
                case "train":
                    requestDatasetType = DatasetType.TRAIN;
                    break;
                case "test":
                    requestDatasetType = DatasetType.TEST;
                    break;
                default:
                    logger.log(Level.SEVERE, "Could not create dataset. Unrecognized
                        dataset type: {0}", datasetType);
                    throw new Exception("Could not create dataset. Unrecognized
                        dataset type: " + datasetType);
            }

            GroundTruthManifest groundTruthManifest =
                GroundTruthManifest.builder()

                    .s3Object(S3Object.builder().bucket(bucket).name(name).build()).build();

            DatasetSource datasetSource =
                DatasetSource.builder().groundTruthManifest(groundTruthManifest).build();

            CreateDatasetRequest createDatasetRequest =
                CreateDatasetRequest.builder().projectArn(projectArn)
```

```
.datasetType(requestDatasetType).datasetSource(datasetSource).build());

    CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

    boolean created = false;

    do {

        DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
            .datasetArn(response.datasetArn()).build();
        DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

        DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

        DatasetStatus status = datasetDescription.status();

        logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());

        switch (status) {

            case CREATE_COMPLETE:
                logger.log(Level.INFO, "Dataset created");
                created = true;
                break;

            case CREATE_IN_PROGRESS:
                Thread.sleep(5000);
                break;

            case CREATE_FAILED:
                String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                    + datasetDescription.statusMessage() + " " +
response.datasetArn();
                logger.log(Level.SEVERE, error);
                throw new Exception(error);

            default:
```

```
        String unexpectedError = "Unexpected creation state: " +
datasetDescription.statusAsString() + " "
            + datasetDescription.statusMessage() + " " +
response.datasetArn();
        logger.log(Level.SEVERE, unexpectedError);
        throw new Exception(unexpectedError);
    }

    } while (created == false);

    return response.datasetArn();

} catch (RekognitionException e) {
    logger.log(Level.SEVERE, "Could not create dataset: {0}",
e.getMessage());
    throw e;
}

}

public static void main(String[] args) {

    String datasetType = null;
    String bucket = null;
    String name = null;
    String projectArn = null;
    String datasetArn = null;

    final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>
<dataset_arn>\n\n" + "Where:\n"
        + "    project_arn - the ARN of the project that you want to add
copy the datast to.\n\n"
        + "    dataset_type - the type of the dataset that you want to
create (train or test).\n\n"
        + "    bucket - the S3 bucket that contains the manifest file.\n
\n"
        + "    name - the location and name of the manifest file within
the bucket.\n\n";

    if (args.length != 4) {
        System.out.println(USAGE);
        System.exit(1);
    }
}
```

```
projectArn = args[0];
datasetType = args[1];
bucket = args[2];
name = args[3];

try {

    // Get the Rekognition client
    RekognitionClient rekClient = RekognitionClient.builder()
        .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
        .region(Region.US_WEST_2)
        .build();

    // Create the dataset
    datasetArn = createMyDataset(rekClient, projectArn, datasetType,
bucket, name);

    System.out.println(String.format("Created dataset: %s",
datasetArn));

    rekClient.close();

} catch (RekognitionException rekError) {
    logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
    System.exit(1);
} catch (Exception rekError) {
    logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
    System.exit(1);
}

}
```

4. Si vous devez ajouter ou modifier des étiquettes, consultez [Gestion des étiquettes \(kit SDK\)](#).
5. Suivez les étapes décrites dans [Entraînement d'un modèle \(SDK\)](#) pour entraîner le modèle.

Offre d'emploi SageMaker chez Amazon Ground Truth

Avec Amazon SageMaker Ground Truth, vous pouvez faire appel à des employés d'Amazon Mechanical Turk, un fournisseur de votre choix, ou à du personnel interne du secteur privé, ainsi qu'à un apprentissage automatique qui vous permet de créer un ensemble d'images étiqueté. Amazon Rekognition Custom Labels importe les fichiers manifestes de SageMaker Ground Truth à partir d'un compartiment Amazon S3 que vous spécifiez.

Les étiquettes personnalisées Amazon Rekognition prennent en charge les tâches Ground Truth suivantes SageMaker .

- [Classification d'images](#)
- [Cadre de délimitation](#)

Les fichiers que vous importez sont les images et un fichier manifeste. Le fichier manifeste contient les informations relatives aux étiquettes et aux cadres de délimitation des images que vous importez.

Amazon Rekognition a besoin d'autorisations pour accéder au compartiment Amazon S3 dans lequel vos images sont stockées. Si vous utilisez le compartiment de la console configuré pour vous par Étiquettes personnalisées Amazon Rekognition, les autorisations requises sont déjà configurées. Si vous n'utilisez pas le compartiment de la console, consultez [Accès à des compartiments Amazon S3 externes](#).

Création d'un fichier manifeste avec une tâche SageMaker Ground Truth (Console)

La procédure suivante explique comment créer un jeu de données à l'aide d'images étiquetées par une tâche SageMaker Ground Truth. Les fichiers de sortie des tâches sont stockés dans le compartiment de la console Étiquettes personnalisées Amazon Rekognition.

Pour créer un jeu de données à l'aide d'images étiquetées par une tâche SageMaker Ground Truth (console)

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon S3 à l'adresse <https://console.aws.amazon.com/s3/>.
2. Dans le compartiment de la console, [créez un dossier](#) destiné à vos images d'entraînement.

Note

Le compartiment de la console est créé lorsque vous ouvrez pour la première fois la console Étiquettes personnalisées Amazon Rekognition dans une région AWS. Pour plus d'informations, consultez [Gestion d'un projet Amazon Rekognition Custom Labels](#).

3. [Chargez vos images](#) dans le dossier que vous venez de créer.
4. Dans le compartiment de la console, créez un dossier destiné à la sortie de la tâche Ground Truth.
5. Ouvrez la SageMaker console à l'[adresse https://console.aws.amazon.com/sagemaker/](https://console.aws.amazon.com/sagemaker/).
6. Créez une tâche d'étiquetage Ground Truth. Vous avez besoin de l'URL Amazon S3 correspondant aux dossiers que vous avez créés aux étapes 2 et 4. Pour plus d'informations, consultez [Utiliser Amazon SageMaker Ground Truth pour l'étiquetage des données](#).
7. Notez l'emplacement du fichier `output.manifest` dans le dossier que vous avez créé à l'étape 4. Il devrait se trouver dans le sous-dossier `Ground-Truth-Job-Name/manifests/output`.
8. Suivez les instructions sous [Création d'un ensemble de données à l'aide d'un fichier manifeste de SageMaker Ground Truth \(Console\)](#) pour créer un jeu de données avec le fichier manifeste chargé. Pour l'étape 8, dans Emplacement du fichier `.manifest`, entrez l'URL Amazon S3 correspondant à l'emplacement indiqué à l'étape précédente. Si vous utilisez le kit AWS SDK, effectuez les actions indiquées dans [Création d'un jeu de données à l'aide d'un fichier manifeste \(SDK\) SageMaker Ground Truth](#).
9. Répétez les étapes 1 à 6 pour créer une tâche SageMaker Ground Truth pour votre ensemble de données de test.

Création d'un fichier manifeste

Vous pouvez créer un ensemble de données de test ou d'entraînement en important un fichier manifeste au format SageMaker Ground Truth. Si vos images sont étiquetées dans un format autre qu'un fichier manifeste de SageMaker Ground Truth, utilisez les informations suivantes pour créer un fichier manifeste au format SageMaker Ground Truth.

Les fichiers manifestes sont au format de [lignes JSON](#) où chaque ligne est un objet JSON complet représentant les informations d'étiquetage d'une image. Les étiquettes personnalisées Amazon

Rekognition prennent en charge les manifestes SageMaker Ground Truth contenant des lignes JSON dans les formats suivants :

- [Sortie de la tâche de classification](#) : à utiliser pour ajouter des étiquettes au niveau de l'image à une image. Une étiquette au niveau de l'image définit la classe de scène, de concept ou d'objet (si les informations de localisation de l'objet ne sont pas nécessaires) figurant sur une image. Une image peut avoir plusieurs étiquettes au niveau de l'image. Pour plus d'informations, consultez [Étiquettes au niveau de l'image dans les fichiers manifestes](#).
- [Sortie de la tâche de cadre de délimitation](#) : à utiliser pour étiqueter la classe et l'emplacement d'un ou de plusieurs objets sur une image. Pour plus d'informations, consultez [Localisation d'objets dans les fichiers manifestes](#).

Les lignes JSON au niveau de l'image et de localisation (cadre de délimitation) peuvent être enchaînées les unes aux autres dans le même fichier manifeste.

Note

La mise en forme des exemples de lignes JSON présentés dans cette section a été modifiée pour une meilleure lisibilité.

Lorsque vous importez un fichier manifeste, Étiquettes personnalisées Amazon Rekognition applique des règles de validation pour les limites, la syntaxe et la sémantique. Pour plus d'informations, consultez [Règles de validation des fichiers manifestes](#).

Les images référencées par un fichier manifeste doivent se trouver dans le même compartiment Amazon S3. Le fichier manifeste peut se trouver dans un compartiment Amazon S3 différent du compartiment Amazon S3 qui stocke les images. Vous spécifiez l'emplacement d'une image dans le champ `source-ref` d'une ligne JSON.

Amazon Rekognition a besoin d'autorisations pour accéder au compartiment Amazon S3 dans lequel vos images sont stockées. Si vous utilisez le compartiment de la console configuré pour vous par Étiquettes personnalisées Amazon Rekognition, les autorisations requises sont déjà configurées. Si vous n'utilisez pas le compartiment de la console, consultez [Accès à des compartiments Amazon S3 externes](#).

Rubriques

- [Création d'un fichier manifeste](#)

- [Étiquettes au niveau de l'image dans les fichiers manifestes](#)
- [Localisation d'objets dans les fichiers manifestes](#)
- [Règles de validation des fichiers manifestes](#)

Création d'un fichier manifeste

La procédure suivante crée un projet avec un jeu de données d'entraînement et de test. Les jeux de données sont générés à partir des fichiers manifestes d'entraînement et de test que vous créez.

Pour créer un ensemble de données à l'aide d'un fichier manifeste au format SageMaker Ground Truth (console)

1. Dans le compartiment de la console, [créez un dossier](#) destiné à vos fichiers manifestes.
2. Dans le compartiment de la console, créez un dossier destiné à vos images.
3. Chargez vos images dans le dossier que vous venez de créer.
4. Créez un fichier manifeste au format SageMaker Ground Truth pour votre ensemble de données d'entraînement. Pour plus d'informations, consultez [Étiquettes au niveau de l'image dans les fichiers manifestes](#) et [Localisation d'objets dans les fichiers manifestes](#).

Important

La valeur du champ `source-ref` dans chaque ligne JSON doit correspondre à une image que vous avez chargée.

5. Créez un fichier manifeste au format SageMaker Ground Truth pour votre ensemble de données de test.
6. [Chargez vos fichiers manifestes](#) dans le dossier que vous venez de créer.
7. Notez l'emplacement du fichier manifeste.
8. Suivez les instructions sous [Création d'un ensemble de données à l'aide d'un fichier manifeste de SageMaker Ground Truth \(Console\)](#) pour créer un jeu de données avec le fichier manifeste chargé. Pour l'étape 8, dans Emplacement du fichier `.manifest`, entrez l'URL Amazon S3 correspondant à l'emplacement indiqué à l'étape précédente. Si vous utilisez le kit AWS SDK, effectuez les actions indiquées dans [Création d'un jeu de données à l'aide d'un fichier manifeste \(SDK\) SageMaker Ground Truth](#).

Étiquettes au niveau de l'image dans les fichiers manifestes

Pour importer des étiquettes au niveau des images (images étiquetées avec des scènes, des concepts ou des objets ne nécessitant pas d'informations de localisation), vous devez ajouter des lignes JSON au format JSON au format SageMaker Ground Truth [Classification Job Output](#) à un fichier manifeste. Un fichier manifeste est composé d'une ou de plusieurs lignes JSON, une pour chaque image que vous souhaitez importer.

Tip

Pour simplifier la création d'un fichier manifeste, nous fournissons un script Python qui crée un fichier manifeste à partir d'un fichier CSV. Pour plus d'informations, consultez [Création d'un fichier manifeste à partir d'un fichier CSV](#).

Pour créer un fichier manifeste pour des étiquettes au niveau de l'image

1. Créez un fichier texte vide.
2. Ajoutez une ligne JSON pour chaque image que vous souhaitez importer. Chaque ligne JSON doit ressembler à ce qui suit.

```
{"source-ref":"s3://custom-labels-console-us-east-1-xxxxxxxxxx/gt-job/manifest/IMG_1133.png","TestCLConsoleBucket":0,"TestCLConsoleBucket-metadata":{"confidence":0.95,"job-name":"labeling-job/testclconsolebucket","class-name":"Echo Dot","human-annotated":"yes","creation-date":"2020-04-15T20:17:23.433061","type":"groundtruth/image-classification"}}
```

3. Enregistrez le fichier. Vous pouvez utiliser l'extension `.manifest`, mais cela n'est pas obligatoire.
4. Créez un jeu de données à l'aide du fichier manifeste que vous avez créé. Pour plus d'informations, consultez [Pour créer un ensemble de données à l'aide d'un fichier manifeste au format SageMaker Ground Truth \(console\)](#).

Lignes JSON au niveau de l'image

Dans cette section, nous vous expliquons comment créer une ligne JSON pour une image unique. Examinez l'image suivante. Une scène pour l'image suivante pourrait s'appeler Sunrise.



La ligne JSON de l'image précédente, avec la scène Sunrise, pourrait être la suivante.

```
{
  "source-ref": "s3://bucket/images/sunrise.png",
  "testdataset-classification_Sunrise": 1,
  "testdataset-classification_Sunrise-metadata": {
    "confidence": 1,
    "job-name": "labeling-job/testdataset-classification_Sunrise",
    "class-name": "Sunrise",
    "human-annotated": "yes",
    "creation-date": "2020-03-06T17:46:39.176",
    "type": "groundtruth/image-classification"
  }
}
```

Notez les informations suivantes.

source-ref

(Obligatoire) Emplacement Amazon S3 de l'image. Le format est "`s3://BUCKET/OBJECT_PATH`". Les images d'un jeu de données importé doivent être stockées dans le même compartiment Amazon S3.

testdataset-classification_Sunrise

(Obligatoire) Attribut de l'étiquette. Vous choisissez le nom du champ. La valeur du champ (1 dans l'exemple précédent) est un identifiant d'attribut d'étiquette. Elle n'est pas utilisée par Étiquettes personnalisées Amazon Rekognition et peut être n'importe quelle valeur entière. Les métadonnées correspondantes doivent être identifiées par le nom du champ, nom auquel vous devez ajouter `-metadata`. Par exemple, "`testdataset-classification_Sunrise-metadata`".

testdataset-classification_Sunrise-metadata

(Obligatoire) Métadonnées relatives à l'attribut de l'étiquette. Le nom du champ doit être identique à celui de l'attribut de l'étiquette. La mention `-metadata` doit être ajoutée à la fin du nom.

confidence

(Obligatoire) Pas utilisé actuellement par Étiquettes personnalisées Amazon Rekognition, mais une valeur comprise entre 0 et 1 doit être fournie.

job-name

(Facultatif) Nom que vous choisissez pour la tâche qui traitera l'image.

class-name

(Obligatoire) Nom de classe que vous choisissez pour la scène ou le concept qui s'applique à l'image. Par exemple, "`Sunrise`".

human-annotated

(Obligatoire) Spécifiez "`yes`" si l'annotation a été complétée par un humain. Sinon, spécifiez "`no`".

creation-date

(Obligatoire) Date et heure UTC (Coordinated Universal Time) de création de l'étiquette.

type

(Obligatoire) Type de traitement à appliquer à l'image. Pour les étiquettes au niveau de l'image, la valeur est "groundtruth/image-classification".

Ajout de plusieurs étiquettes au niveau de l'image à une image

Vous pouvez ajouter plusieurs étiquettes à une image. Par exemple, le code JSON suivant ajoute deux étiquettes, football et ball, à une seule image.

```
{
  "source-ref": "S3 bucket location",
  "sport0":0, # FIRST label
  "sport0-metadata": {
    "class-name": "football",
    "confidence": 0.8,
    "type":"groundtruth/image-classification",
    "job-name": "identify-sport",
    "human-annotated": "yes",
    "creation-date": "2018-10-18T22:18:13.527256"
  },
  "sport1":1, # SECOND label
  "sport1-metadata": {
    "class-name": "ball",
    "confidence": 0.8,
    "type":"groundtruth/image-classification",
    "job-name": "identify-sport",
    "human-annotated": "yes",
    "creation-date": "2018-10-18T22:18:13.527256"
  }
} # end of annotations for 1 image
```

Localisation d'objets dans les fichiers manifestes

Vous pouvez importer des images étiquetées avec des informations de localisation d'objets en ajoutant des lignes JSON au format JSON SageMaker Ground Truth [Bounding Box Job Output](#) à un fichier manifeste.

Les informations de localisation représentent l'emplacement d'un objet sur une image.

L'emplacement est représenté par un cadre de délimitation qui entoure l'objet en question.

La structure du cadre de délimitation contient les coordonnées en haut à gauche du cadre de

délimitation ainsi que la largeur et la hauteur de ce dernier. Une ligne JSON au format de cadre de délimitation comprend des cadres de délimitation indiquant les emplacements d'un ou de plusieurs objets sur une image et la classe de chaque objet sur l'image.

Un fichier manifeste est composé d'une ou de plusieurs lignes JSON, chaque ligne contenant les informations relatives à une seule image.

Pour créer un fichier manifeste pour la localisation d'objets

1. Créez un fichier texte vide.
2. Ajoutez une ligne JSON pour chaque image que vous souhaitez importer. Chaque ligne JSON doit ressembler à ce qui suit.

```
{"source-ref": "s3://bucket/images/IMG_1186.png", "bounding-box": {"image_size": [{"width": 640, "height": 480, "depth": 3}], "annotations": [{"class_id": 1, "top": 251, "left": 399, "width": 155, "height": 101}, {"class_id": 0, "top": 65, "left": 86, "width": 220, "height": 334}]}, "bounding-box-metadata": {"objects": [{"confidence": 1}, {"confidence": 1}], "class-map": {"0": "Echo", "1": "Echo Dot"}, "type": "groundtruth/object-detection", "human-annotated": "yes", "creation-date": "2013-11-18T02:53:27", "job-name": "my job"}}
```

3. Enregistrez le fichier. Vous pouvez utiliser l'extension `.manifest`, mais cela n'est pas obligatoire.
4. Créez un jeu de données à l'aide du fichier que vous venez de créer. Pour plus d'informations, consultez [Pour créer un ensemble de données à l'aide d'un fichier manifeste au format SageMaker Ground Truth \(console\)](#).

Lignes JSON du cadre de délimitation des objets

Dans cette section, nous vous expliquons comment créer une ligne JSON pour une image unique. L'image suivante affiche des cadres de délimitation autour des appareils Amazon Echo et Amazon Echo Dot.



Voici la ligne JSON des cadres de délimitation correspondant à l'image précédente.

```
{
  "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
  "bounding-box": {
    "image_size": [{
      "width": 640,
      "height": 480,
      "depth": 3
    }],
    "annotations": [{
      "class_id": 1,
      "top": 251,
      "left": 399,
      "width": 155,
```



```
"height": 101
}, {
  "class_id": 0,
  "top": 65,
  "left": 86,
  "width": 220,
  "height": 334
}]
},
"bounding-box-metadata": {
  "objects": [{
    "confidence": 1
  }, {
    "confidence": 1
  }],
  "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2013-11-18T02:53:27",
  "job-name": "my job"
}
}
```

Notez les informations suivantes.

source-ref

(Obligatoire) Emplacement Amazon S3 de l'image. Le format est "`s3://BUCKET/OBJECT_PATH`". Les images d'un jeu de données importé doivent être stockées dans le même compartiment Amazon S3.

bounding-box

(Obligatoire) Attribut de l'étiquette. Vous choisissez le nom du champ. Contient la taille de l'image et des cadres de délimitation pour chaque objet détecté dans l'image. Les métadonnées correspondantes doivent être identifiées par le nom du champ, nom auquel vous devez ajouter - metadata. Par exemple, "bounding-box-metadata".

image_size

(Obligatoire) Tableau à élément unique contenant la taille de l'image en pixels.

- `height` : (obligatoire) hauteur de l'image, en pixels.
- `width` : (obligatoire) profondeur de l'image en pixels.
- `depth` : (obligatoire) nombre de canaux dans l'image. Pour les images RVB, la valeur est 3. Pas utilisé actuellement par Étiquettes personnalisées Amazon Rekognition, mais une valeur doit être fournie.

annotations

(Obligatoire) Tableau d'informations relatives au cadre de délimitation pour chaque objet détecté dans l'image.

- `class_id` : (obligatoire) correspond à l'étiquette dans `class-map`. Dans l'exemple précédent, l'objet avec la valeur 1 comme `class_id` est l'appareil Echo Dot sur l'image.
- `top` : (obligatoire) distance entre le haut de l'image et le haut du cadre de délimitation, en pixels.
- `left` : (obligatoire) distance entre la gauche de l'image et la gauche du cadre de délimitation, en pixels.
- `width` : (obligatoire) largeur du cadre de délimitation, en pixels.
- `height` : (obligatoire) hauteur du cadre de délimitation, en pixels.

bounding-box-metadata

(Obligatoire) Métadonnées relatives à l'attribut de l'étiquette. Le nom du champ doit être identique à celui de l'attribut de l'étiquette. La mention `-metadata` doit être ajoutée à la fin du nom. Tableau d'informations relatives au cadre de délimitation pour chaque objet détecté dans l'image.

Objets

(Obligatoire) Tableau d'objets qui se trouvent dans l'image. Correspond au tableau `annotations` par index. L'attribut de confiance n'est pas utilisé par Étiquettes personnalisées Amazon Rekognition.

class-map

(Obligatoire) Mappage des classes qui s'appliquent aux objets détectés dans l'image.

type

(Obligatoire) Type de tâche de classification. "groundtruth/object-detection" identifie la tâche en tant que détection d'objets.

creation-date

(Obligatoire) Date et heure UTC (Coordinated Universal Time) de création de l'étiquette.

human-annotated

(Obligatoire) Spécifiez "yes" si l'annotation a été complétée par un humain. Sinon, spécifiez "no".

job-name

(Facultatif) Nom de la tâche qui traitera l'image.

Règles de validation des fichiers manifestes

Lorsque vous importez un fichier manifeste, Étiquettes personnalisées Amazon Rekognition applique des règles de validation pour les limites, la syntaxe et la sémantique. Le schéma SageMaker Ground Truth impose la validation de la syntaxe. Pour plus d'informations, consultez [Sorties](#). Vous trouverez ci-dessous les règles de validation des limites et de la sémantique.

Note

- Les règles de non-validité de 20 % s'appliquent cumulativement à toutes les règles de validation. Si l'importation dépasse la limite de 20 % en raison d'une combinaison quelconque, telle que 15 % de code JSON non valide et 15 % d'images non valides, l'importation échoue.
- Chaque objet de jeu de données est une ligne dans le manifeste. Les lignes vierges/non valides sont également considérées comme des objets de jeu de données.
- Les chevauchements correspondent aux (étiquettes communes entre le test et l'entraînement)/(étiquettes d'entraînement).

Rubriques

- [Limites](#)
- [Sémantique](#)

Limites

Validation	Limite	Erreur signalée
Taille du fichier manifeste	1 Go maximum	Erreur
Nombre maximal de lignes pour un fichier manifeste	Maximum de 250 000 objets de jeu de données sous forme de lignes dans un manifeste.	Erreur
Limite inférieure du nombre total d'objets de jeu de données valides par étiquette	≥ 1	Erreur
Limite inférieure au niveau des étiquettes	≥ 2	Erreur
Limite supérieure au niveau des étiquettes	≤ 250	Erreur
Nombre minimal de cadres de délimitation par image	0	Aucun
Nombre maximal de cadres de délimitation par image	50	Aucun

Sémantique

Validation	Limite	Erreur signalée
Manifeste vide		Erreur
Objet source-ref manquant/ inaccessible	Nombre d'objets inférieur à 20 %	Avertissement
Objet source-ref manquant/ inaccessible	Nombre d'objets > 20 %	Erreur

Validation	Limite	Erreur signalée
Étiquettes de test non présentes dans le jeu de données d'entraînement	Au moins 50 % de chevauchement dans les étiquettes	Erreur
Combinaison d'exemples d'étiquettes et d'objets pour une même étiquette dans un jeu de données. Classification et détection pour la même classe dans un objet de jeu de données.		Aucune erreur ni aucun avertissement
Chevauchement des ressources entre le test et l'entraînement	Il ne doit pas y avoir de chevauchement entre les jeux de données de test et d'entraînement.	
Les images d'un jeu de données doivent provenir du même compartiment	Erreur si les objets se trouvent dans un autre compartiment	Erreur

Conversion d'autres formats de jeu de données en fichier manifeste

Vous pouvez utiliser les informations suivantes pour créer des fichiers manifestes SageMaker au format Amazon à partir de différents formats de jeux de données sources. Après avoir créé le fichier manifeste, utilisez-le pour créer un jeu de données. Pour plus d'informations, consultez [Fichier manifeste](#).

Rubriques

- [Conversion des jeux de données COCO](#)
- [Transformation des fichiers manifestes multi-labels de SageMaker Ground Truth](#)
- [Création d'un fichier manifeste à partir d'un fichier CSV](#)

Conversion des jeux de données COCO

[COCO](#) est un format permettant de spécifier des jeux de données de détection, de segmentation et de sous-titrage des objets à grande échelle. Cet [exemple](#) Python vous montre comment transformer un jeu de données au format de détection d'objets COCO en un [fichier manifeste Étiquettes personnalisées Amazon Rekognition au format cadre de délimitation](#). Cette section inclut également des informations que vous pouvez utiliser pour écrire votre propre code.

Un fichier JSON au format COCO se compose de cinq sections fournissant des informations pour un jeu de données complet. Pour plus d'informations, consultez [Format COCO](#).

- `info` : informations générales sur le jeu de données.
- `licenses` : informations de licence pour les images du jeu de données.
- `images` : liste des images du jeu de données.
- `annotations` : liste d'annotations (y compris les cadres de délimitation) présentes dans toutes les images du jeu de données.
- `categories` : liste des catégories d'étiquettes.

Vous aurez besoin d'informations provenant des listes `images`, `annotations` et `categories` pour créer un fichier manifeste Étiquettes personnalisées Amazon Rekognition.

Un fichier manifeste Étiquettes personnalisées Amazon Rekognition est au format de lignes JSON. Chaque ligne contient les informations relatives aux cadres de délimitation et aux étiquettes d'un ou de plusieurs objets dans une image. Pour plus d'informations, consultez [Localisation d'objets dans les fichiers manifestes](#).

Mappage d'objets COCO avec une ligne JSON Étiquettes personnalisées

Pour transformer un jeu de données au format COCO, mappez le jeu de données COCO avec un fichier manifeste Étiquettes personnalisées Amazon Rekognition pour la localisation d'objets. Pour plus d'informations, consultez [Localisation d'objets dans les fichiers manifestes](#). Pour créer une ligne JSON pour chaque image, le fichier manifeste doit mapper les ID des champs `image`, `annotation` et `category` du jeu de données COCO.

Voici un exemple de fichier manifeste COCO. Pour plus d'informations, consultez [Format COCO](#).

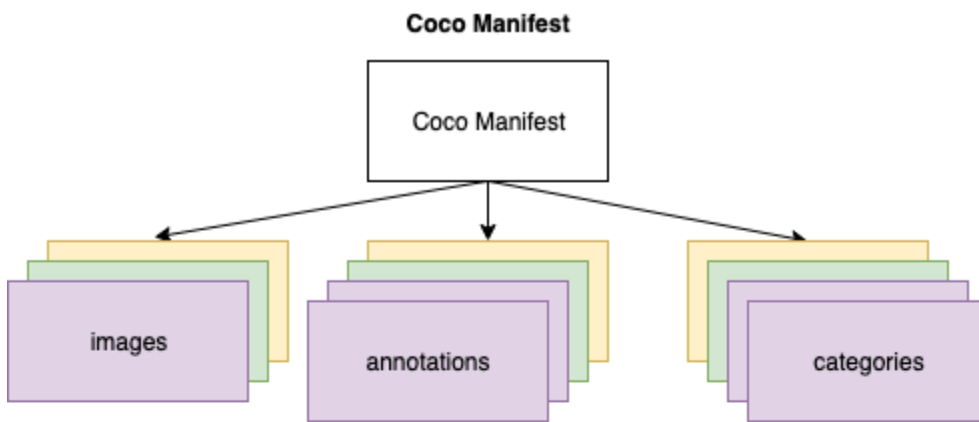
```
{
  "info": {
```

```

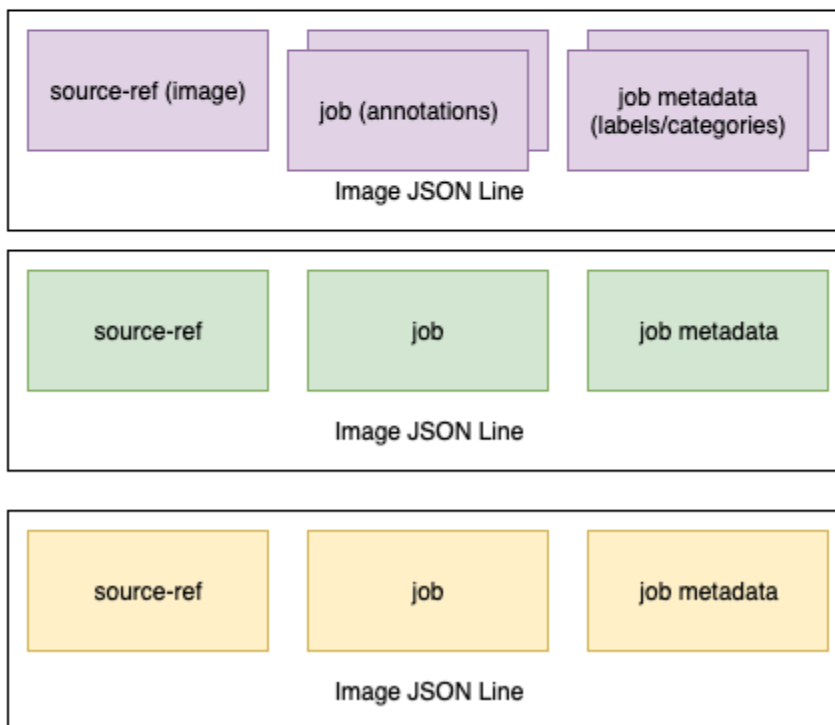
    "description": "COCO 2017 Dataset","url": "http://cocodataset.org","version":
"1.0","year": 2017,"contributor": "COCO Consortium","date_created": "2017/09/01"
  },
  "licenses": [
    {"url": "http://creativecommons.org/licenses/by/2.0/","id": 4,"name":
"Attribution License"}
  ],
  "images": [
    {"id": 242287, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/xxxxxxxxxxxxx.jpg", "flickr_url": "http://farm3.staticflickr.com/2626/
xxxxxxxxxxxxx.jpg", "width": 426, "height": 640, "file_name": "xxxxxxxxx.jpg",
"date_captured": "2013-11-15 02:41:42"},
    {"id": 245915, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/nnnnnnnnnnn.jpg", "flickr_url": "http://farm1.staticflickr.com/88/
xxxxxxxxxxxxx.jpg", "width": 640, "height": 480, "file_name": "nnnnnnnnnnn.jpg",
"date_captured": "2013-11-18 02:53:27"}
  ],
  "annotations": [
    {"id": 125686, "category_id": 0, "iscrowd": 0, "segmentation": [[164.81,
417.51,.....167.55, 410.64]], "image_id": 242287, "area": 42061.80340000001, "bbox":
[19.23, 383.18, 314.5, 244.46]},
    {"id": 1409619, "category_id": 0, "iscrowd": 0, "segmentation": [[376.81,
238.8,.....382.74, 241.17]], "image_id": 245915, "area": 3556.2197000000015,
"bbox": [399, 251, 155, 101]},
    {"id": 1410165, "category_id": 1, "iscrowd": 0, "segmentation": [[486.34,
239.01,.....495.95, 244.39]], "image_id": 245915, "area": 1775.8932499999994,
"bbox": [86, 65, 220, 334]}
  ],
  "categories": [
    {"supercategory": "speaker","id": 0,"name": "echo"},
    {"supercategory": "speaker","id": 1,"name": "echo dot"}
  ]
}

```

Le schéma suivant montre comment les listes de jeux de données COCO pour un jeu de données sont mappées avec les lignes JSON d'Étiquettes personnalisées Amazon Rekognition pour une image. Les couleurs qui sont les mêmes indiquent des informations relatives à une seule image.



Custom Labels JSON Lines



Pour obtenir les objets COCO pour une seule ligne JSON

1. Pour chaque image de la liste d'images, récupérez l'annotation dans la liste d'annotations où la valeur du champ d'annotation `image_id` correspond à celle du champ `id` de l'image.
2. Pour chaque annotation ayant une correspondance dans l'étape 1, parcourez la liste `categories` et récupérez chaque `category` où la valeur du champ `category_id` correspond au champ `category_id` de l'objet annotation.

3. Créez une ligne JSON pour l'image à l'aide des objets `image`, `annotation` et `category` associés. Pour mapper les champs, consultez [Mappage de champs d'objet COCO avec des champs d'objet de ligne JSON Étiquettes personnalisées](#).
4. Répétez les étapes 1 à 3 jusqu'à ce que vous ayez créé des lignes JSON pour chaque objet image de la liste `images`.

Pour obtenir un exemple de code, consultez [Conversion d'un jeu de données COCO](#).

Mappage de champs d'objet COCO avec des champs d'objet de ligne JSON Étiquettes personnalisées

Après avoir identifié les objets COCO pour une ligne Étiquettes personnalisées Amazon Rekognition, vous devez mapper les champs d'objets COCO avec les champs d'objet de ligne JSON Étiquettes personnalisées Amazon Rekognition respectifs. L'exemple suivant de ligne JSON Étiquettes personnalisées Amazon Rekognition mappe une image (`id=000000245915`) avec l'exemple JSON COCO précédent. Notez les informations suivantes.

- `source-ref` correspond à l'emplacement de l'image dans un compartiment Amazon S3. Si vos images COCO ne sont pas stockées dans un compartiment Amazon S3, vous devez les y transférer.
- La liste `annotations` contient un objet `annotation` pour chaque objet de l'image. Un objet `annotation` comprend des informations relatives à un cadre de délimitation (`top`, `left`, `width`, `height`) et un identifiant d'étiquette (`class_id`).
- L'identifiant d'étiquette (`class_id`) correspond à la liste `class-map` figurant dans les métadonnées. Il répertorie les étiquettes utilisées sur l'image.

```
{
  "source-ref": "s3://custom-labels-bucket/images/000000245915.jpg",
  "bounding-box": {
    "image_size": {
      "width": 640,
      "height": 480,
      "depth": 3
    },
    "annotations": [{
      "class_id": 0,
      "top": 251,
      "left": 399,
```

```
    "width": 155,
    "height": 101
  }, {
    "class_id": 1,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
  }
],
"bounding-box-metadata": {
  "objects": [{
    "confidence": 1
  }, {
    "confidence": 1
  }],
  "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
}
}
```

Utilisez les informations suivantes pour mapper les champs du fichier manifeste Étiquettes personnalisées Amazon Rekognition avec les champs JSON du jeu de données COCO.

source-ref

URL au format S3 pour l'emplacement de l'image. L'image doit être stockée dans un compartiment S3. Pour plus d'informations, consultez [source-ref](#). Si le champ COCO `coco_url` pointe vers l'emplacement d'un compartiment S3, vous pouvez utiliser la valeur de `coco_url` pour la valeur de `source-ref`. Vous pouvez également mapper `source-ref` avec le champ (COCO) `file_name` et ajouter dans le code de transformation le chemin S3 requis vers l'endroit où l'image est stockée.

bounding-box

Nom d'attribut d'étiquette de votre choix. Pour plus d'informations, consultez [bounding-box](#).

image_size

Taille de l'image, en pixels. Correspond à un objet `image` dans la liste des [images](#).

- `height`-> [image](#).height
- `width`-> [image](#).width
- `depth`-> Pas utilisé par Étiquettes personnalisées Amazon Rekognition, mais une valeur doit être fournie.

annotations

Liste d'objets annotation. Il y a une annotation pour chaque objet de l'image.

annotation

Contient les informations relatives au cadre de délimitation pour une instance d'un objet sur l'image.

- `class_id` -> mappage d'ID numérique avec la liste `class-map` d'Étiquettes personnalisées.
- `top` -> [bbox](#)[1]
- `left` -> [bbox](#)[0]
- `width` -> [bbox](#)[2]
- `height` -> [bbox](#)[3]

bounding-box-metadata

Métadonnées pour l'attribut d'étiquette. Inclut les étiquettes et les identifiants d'étiquettes. Pour plus d'informations, consultez [bounding-box-metadata](#).

Objets

Tableau des objets de l'image. Correspond à la liste `annotations` par index.

Objet

- `confidence`-> Pas utilisé par Étiquettes personnalisées Amazon Rekognition, mais une valeur (1) doit être fournie.

class-map

Mappage des étiquettes (classes) qui s'appliquent aux objets détectés dans l'image. Correspond aux objets de catégorie dans la liste des [catégories](#).

- id -> [category](#).id
- id value -> [category](#).name

type

Doit être groundtruth/object-detection

human-annotated

Spécifiez yes ou no. Pour plus d'informations, consultez [bounding-box-metadata](#).

creation-date -> [image](#).date_captured

Date et heure de création de l'image. Correspond au champ [image](#).date_captured d'une image dans la liste des images COCO. Étiquettes personnalisées Amazon Rekognition attend du format creation-date qu'il correspond à Y-M-DTH:M:S.

job-name

Nom de tâche de votre choix.

Format COCO

Un jeu de données COCO se compose de cinq sections d'informations qui fournissent des informations pour le jeu de données complet. Le format d'un jeu de données de détection d'objets COCO est documenté dans la section [Format de données COCO](#).

- info : informations générales sur le jeu de données.
- licenses : informations de licence pour les images du jeu de données.
- [images](#) : liste des images du jeu de données.
- [annotations](#) : liste d'annotations (y compris les cadres de délimitation) présentes dans toutes les images du jeu de données.
- [categories](#) : liste des catégories d'étiquettes.

Pour créer un manifeste Étiquettes personnalisées, utilisez les listes images, annotations et catégories du fichier manifeste COCO. Les autres sections (info, licences) ne sont pas obligatoires. Voici un exemple de fichier manifeste COCO.

```
{
  "info": {
    "description": "COCO 2017 Dataset","url": "http://cocodataset.org","version":
"1.0","year": 2017,"contributor": "COCO Consortium","date_created": "2017/09/01"
  },
  "licenses": [
    {"url": "http://creativecommons.org/licenses/by/2.0/","id": 4,"name":
"Attribution License"}
  ],
  "images": [
    {"id": 242287, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/xxxxxxxxxxxxx.jpg", "flickr_url": "http://farm3.staticflickr.com/2626/
xxxxxxxxxxxxx.jpg", "width": 426, "height": 640, "file_name": "xxxxxxxx.jpg",
"date_captured": "2013-11-15 02:41:42"},
    {"id": 245915, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/nnnnnnnnnnn.jpg", "flickr_url": "http://farm1.staticflickr.com/88/
xxxxxxxxxxxxx.jpg", "width": 640, "height": 480, "file_name": "nnnnnnnnnn.jpg",
"date_captured": "2013-11-18 02:53:27"}
  ],
  "annotations": [
    {"id": 125686, "category_id": 0, "iscrowd": 0, "segmentation": [[164.81,
417.51,.....167.55, 410.64]], "image_id": 242287, "area": 42061.80340000001, "bbox":
[19.23, 383.18, 314.5, 244.46]},
    {"id": 1409619, "category_id": 0, "iscrowd": 0, "segmentation": [[376.81,
238.8,.....382.74, 241.17]], "image_id": 245915, "area": 3556.2197000000015,
"bbox": [399, 251, 155, 101]},
    {"id": 1410165, "category_id": 1, "iscrowd": 0, "segmentation": [[486.34,
239.01,.....495.95, 244.39]], "image_id": 245915, "area": 1775.8932499999994,
"bbox": [86, 65, 220, 334]}
  ],
  "categories": [
    {"supercategory": "speaker","id": 0,"name": "echo"},
    {"supercategory": "speaker","id": 1,"name": "echo dot"}
  ]
}
```

liste d'images

Les images référencées par un jeu de données COCO sont répertoriées dans le tableau d'images. Chaque objet image contient des informations sur l'image, telles que le nom du fichier image. Dans l'exemple d'objet image ci-dessous, notez les informations suivantes et les champs obligatoires pour créer un fichier manifeste Étiquettes personnalisées Amazon Rekognition.

- `id` : (obligatoire) identifiant unique de l'image. Le champ `id` correspond au champ `id` du tableau d'annotations (où les informations relatives aux cadres de délimitation sont stockées).
- `license` : (facultatif) correspond au tableau de licences.
- `coco_url` : (facultatif) emplacement de l'image.
- `flickr_url` : (facultatif) emplacement de l'image sur Flickr.
- `width` : (obligatoire) largeur de l'image.
- `height` : (obligatoire) hauteur de l'image.
- `file_name` : (obligatoire) nom du fichier image. Dans cet exemple, `file_name` et `id` correspondent, mais cela n'est pas obligatoire pour les jeux de données COCO.
- `date_captured` : (obligatoire) date et heure de capture de l'image.

```
{
  "id": 245915,
  "license": 4,
  "coco_url": "http://images.cocodataset.org/val2017/nnnnnnnnnnnnn.jpg",
  "flickr_url": "http://farm1.staticflickr.com/88/nnnnnnnnnnnnnnnnnnn.jpg",
  "width": 640,
  "height": 480,
  "file_name": "000000245915.jpg",
  "date_captured": "2013-11-18 02:53:27"
}
```

liste des annotations (cadres de délimitation)

Les informations relatives aux cadres de délimitation pour tous les objets de toutes les images sont stockées dans la liste des annotations. Un seul objet d'annotation contient les informations relatives au cadre de délimitation correspondant à un seul objet et l'étiquette de l'objet sur une image. Il existe un objet d'annotation pour chaque instance d'un objet sur une image.

Dans l'exemple ci-dessous, notez les informations suivantes et les champs obligatoires pour créer un fichier manifeste Étiquettes personnalisées Amazon Rekognition.

- `id` : (facultatif) identifiant de l'annotation.
- `image_id` : (obligatoire) correspond à l'image `id` dans le tableau d'images.
- `category_id` : (obligatoire) identifiant de l'étiquette qui identifie l'objet dans un cadre de délimitation. Il correspond au champ `id` du tableau des catégories.
- `iscrowd` : (facultatif) spécifie si l'image contient une foule d'objets.
- `segmentation` : (facultatif) informations de segmentation pour les objets d'une image. Étiquettes personnalisées Amazon Rekognition ne prend pas en charge la segmentation.
- `area` : (facultatif) zone de l'annotation.
- `bbox` : (obligatoire) contient les coordonnées, en pixels, d'un cadre de délimitation autour d'un objet sur l'image.

```
{
  "id": 1409619,
  "category_id": 1,
  "iscrowd": 0,
  "segmentation": [
    [86.0, 238.8, .....382.74, 241.17]
  ],
  "image_id": 245915,
  "area": 3556.2197000000015,
  "bbox": [86, 65, 220, 334]
}
```

liste des catégories

Les informations relatives aux étiquettes sont stockées dans le tableau des catégories. Dans l'exemple d'objet de catégorie ci-dessous, notez les informations suivantes et les champs obligatoires pour créer un fichier manifeste Étiquettes personnalisées Amazon Rekognition.

- `supercategory` : (facultatif) catégorie parent d'une étiquette.
- `id` : (obligatoire) identifiant de l'étiquette. Le champ `id` correspond au champ `category_id` d'un objet annotation. Dans l'exemple suivant, l'identifiant d'un appareil echo dot est 2.
- `name` : (obligatoire) nom de l'étiquette.

```
{"supercategory": "speaker", "id": 2, "name": "echo dot"}
```

Conversion d'un jeu de données COCO

Utilisez l'exemple Python suivant pour transformer les informations du cadre de délimitation d'un jeu de données au format COCO en un fichier manifeste Étiquettes personnalisées Amazon Rekognition. Le code charge le fichier manifeste créé dans votre compartiment Amazon S3. Le code fournit également une commande AWS CLI que vous pouvez utiliser pour charger vos images.

Pour convertir un jeu de données COCO (kit SDK)

1. Si vous ne l'avez pas déjà fait :
 - a. Vérifiez que vous disposez des autorisations AmazonS3FullAccess. Pour plus d'informations, consultez [Configuration des autorisations du kit SDK](#).
 - b. Installez et configurez l'AWS CLI et les kits SDK AWS. Pour plus d'informations, consultez [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez le code Python suivant pour convertir un jeu de données COCO. Définissez les valeurs suivantes.
 - `s3_bucket` : nom du compartiment S3 dans lequel vous souhaitez stocker les images et le fichier manifeste Étiquettes personnalisées Amazon Rekognition.
 - `s3_key_path_images` : chemin d'accès vers l'endroit où vous souhaitez placer les images dans le compartiment S3 (`s3_bucket`).
 - `s3_key_path_manifest_file` : chemin d'accès vers l'endroit où vous souhaitez placer le fichier manifeste Étiquettes personnalisées dans le compartiment S3 (`s3_bucket`).
 - `local_path` : chemin local vers lequel l'exemple ouvre le jeu de données COCO en entrée et enregistre également le nouveau fichier manifeste Étiquettes personnalisées.
 - `local_images_path` : chemin local vers les images que vous souhaitez utiliser pour l'entraînement.
 - `coco_manifest` : nom de fichier du jeu de données COCO en entrée.
 - `cl_manifest_file` : nom du fichier manifeste créé par l'exemple. Ce fichier est enregistré à l'emplacement indiqué par `local_path`. Par convention, il possède l'extension `.manifest`, mais cela n'est pas obligatoire.
 - `job_name` : nom de la tâche Étiquettes personnalisées.

```
import json
import os
```

```
import random
import shutil
import datetime
import boto3
import boto3
import PIL.Image as Image
import io

#S3 location for images
s3_bucket = 'bucket'
s3_key_path_manifest_file = 'path to custom labels manifest file/'
s3_key_path_images = 'path to images/'
s3_path='s3://' + s3_bucket + '/' + s3_key_path_images
s3 = boto3.resource('s3')

#Local file information
local_path='path to input COCO dataset and output Custom Labels manifest/'
local_images_path='path to COCO images/'
coco_manifest = 'COCO dataset JSON file name'
coco_json_file = local_path + coco_manifest
job_name='Custom Labels job name'
cl_manifest_file = 'custom_labels.manifest'

label_attribute = 'bounding-box'

open(local_path + cl_manifest_file, 'w').close()

# class representing a Custom Label JSON line for an image
class cl_json_line:
    def __init__(self, job, img):

        #Get image info. Annotations are dealt with separately
        sizes=[]
        image_size={}
        image_size["width"] = img["width"]
        image_size["depth"] = 3
        image_size["height"] = img["height"]
        sizes.append(image_size)

        bounding_box={}
        bounding_box["annotations"] = []
        bounding_box["image_size"] = sizes

        self.__dict__["source-ref"] = s3_path + img['file_name']
```

```
self.__dict__[job] = bounding_box

#get metadata
metadata = {}
metadata['job-name'] = job_name
metadata['class-map'] = {}
metadata['human-annotated']='yes'
metadata['objects'] = []
date_time_obj = datetime.datetime.strptime(img['date_captured'], '%Y-%m-%d
%H:%M:%S')
metadata['creation-date']= date_time_obj.strftime('%Y-%m-%dT%H:%M:%S')
metadata['type']='groundtruth/object-detection'

self.__dict__[job + '-metadata'] = metadata

print("Getting image, annotations, and categories from COCO file...")

with open(coco_json_file) as f:

    #Get custom label compatible info
    js = json.load(f)
    images = js['images']
    categories = js['categories']
    annotations = js['annotations']

    print('Images: ' + str(len(images)))
    print('annotations: ' + str(len(annotations)))
    print('categories: ' + str(len(categories)))

print("Creating CL JSON lines...")

images_dict = {image['id']: cl_json_line(label_attribute, image) for image in
images}

print('Parsing annotations...')
for annotation in annotations:

    image=images_dict[annotation['image_id']]

    cl_annotation = {}
    cl_class_map={}
```



```
# get bounding box information
cl_bounding_box={}
cl_bounding_box['left'] = annotation['bbox'][0]
cl_bounding_box['top'] = annotation['bbox'][1]

cl_bounding_box['width'] = annotation['bbox'][2]
cl_bounding_box['height'] = annotation['bbox'][3]
cl_bounding_box['class_id'] = annotation['category_id']

getattr(image, label_attribute)['annotations'].append(cl_bounding_box)

for category in categories:
    if annotation['category_id'] == category['id']:
        getattr(image, label_attribute + '-metadata')['class-map']
[category['id']] = category['name']

cl_object={}
cl_object['confidence'] = int(1) #not currently used by Custom Labels
getattr(image, label_attribute + '-metadata')['objects'].append(cl_object)

print('Done parsing annotations')

# Create manifest file.
print('Writing Custom Labels manifest...')

for im in images_dict.values():

    with open(local_path+cl_manifest_file, 'a+') as outfile:
        json.dump(im.__dict__, outfile)
        outfile.write('\n')
        outfile.close()

# Upload manifest file to S3 bucket.
print ('Uploading Custom Labels manifest file to S3 bucket')
print('Uploading' + local_path + cl_manifest_file + ' to ' +
s3_key_path_manifest_file)
print(s3_bucket)
s3 = boto3.resource('s3')
s3.Bucket(s3_bucket).upload_file(local_path + cl_manifest_file,
s3_key_path_manifest_file + cl_manifest_file)

# Print S3 URL to manifest file,
```

```
print ('S3 URL Path to manifest file. ')
print('\033[1m s3://' + s3_bucket + '/' + s3_key_path_manifest_file +
      cl_manifest_file + '\033[0m')

# Display aws s3 sync command.
print ('\nAWS CLI s3 sync command to upload your images to S3 bucket. ')
print ('\033[1m aws s3 sync ' + local_images_path + ' ' + s3_path + '\033[0m')
```

3. Exécutez le code.
4. Dans la sortie du programme, notez la commande `s3 sync`. Vous en aurez besoin à l'étape suivante.
5. À partir d'une invite de commande, exécutez la commande `s3 sync`. Vos images sont importées dans le compartiment S3. Si la commande échoue pendant le chargement, exécutez-la à nouveau jusqu'à ce que les images locales soient synchronisées avec le compartiment S3.
6. Dans la sortie du programme, notez le chemin de l'URL S3 vers le fichier manifeste. Vous en aurez besoin à l'étape suivante.
7. Suivez les instructions sous [Création d'un ensemble de données à l'aide d'un fichier manifeste de SageMaker Ground Truth \(Console\)](#) pour créer un jeu de données avec le fichier manifeste chargé. Pour l'étape 8, dans Emplacement du fichier `.manifest`, entrez l'URL Amazon S3 que vous avez notée à l'étape précédente. Si vous utilisez le kit AWS SDK, effectuez les actions indiquées dans [Création d'un jeu de données à l'aide d'un fichier manifeste \(SDK\) SageMaker Ground Truth](#).

Transformation des fichiers manifestes multi-labels de SageMaker Ground Truth

Cette rubrique explique comment transformer un fichier manifeste Amazon SageMaker Ground Truth multi-étiquettes en un fichier manifeste au format Amazon Rekognition Custom Labels.

SageMaker Les fichiers manifestes de Ground Truth pour les tâches à étiquettes multiples sont formatés différemment des fichiers manifestes au format Amazon Rekognition Custom Labels. La classification à plusieurs étiquettes a lieu lorsqu'une image est classée dans un ensemble de classes, mais qu'elle peut appartenir à plusieurs classes à la fois. Dans ce cas, l'image peut potentiellement comporter plusieurs étiquettes, telles que football et ballon.

Pour plus d'informations sur les tâches SageMaker Ground Truth à étiquettes multiples, consultez la section [Classification d'images \(étiquettes multiples\)](#). Pour plus d'informations sur les fichiers

manifestes Étiquettes personnalisées Amazon Rekognition à plusieurs étiquettes, consultez [the section called “Ajout de plusieurs étiquettes au niveau de l'image à une image”](#).

Obtenir le fichier manifeste pour une tâche de SageMaker Ground Truth

La procédure suivante explique comment obtenir le fichier manifeste de sortie (`output.manifest`) pour une tâche Amazon SageMaker Ground Truth. Utilisez `output.manifest` comme entrée pour la procédure suivante.

Pour télécharger un fichier manifeste de travail de SageMaker Ground Truth

1. Ouvrez la console à l'adresse <https://console.aws.amazon.com/sagemaker/>.
2. Dans le volet de navigation, choisissez Ground Truth, puis Étiquetage des tâches.
3. Choisissez la tâche d'étiquetage qui contient le fichier manifeste que vous souhaitez utiliser.
4. Sur la page de détails, cliquez sur le lien sous Emplacement de l'ensemble de données de sortie. La console Amazon S3 s'ouvre à l'emplacement du jeu de données.
5. Choisissez Manifests, `output` puis `output.manifest`.
6. Pour télécharger le fichier manifeste, choisissez Actions d'objet, puis Télécharger.

Transformation d'un fichier SageMaker manifeste à étiquettes multiples

La procédure suivante crée un fichier manifeste Amazon Rekognition Custom Labels au format multi-étiquettes à partir d'un fichier manifeste au format multi-étiquettes existant. SageMaker GroundTruth

Note

Pour exécuter le code, vous avez besoin de la version 3 de Python ou d'une version supérieure.

Pour transformer un fichier SageMaker manifeste à étiquettes multiples

1. Exécutez le code Python suivant. Indiquez le nom du fichier manifeste que vous avez créé dans [Obtenir le fichier manifeste pour une tâche de SageMaker Ground Truth](#) en tant qu'argument de ligne de commande.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
"""
```

Purpose

Shows how to create an Amazon Rekognition Custom Labels format manifest file from an Amazon SageMaker Ground Truth Image Classification (Multi-label) format manifest file.

```
"""
import json
import logging
import argparse
import os.path

logger = logging.getLogger(__name__)

def create_manifest_file(ground_truth_manifest_file):
    """
    Creates an Amazon Rekognition Custom Labels format manifest file from
    an Amazon SageMaker Ground Truth Image Classification (Multi-label) format
    manifest file.
    :param: ground_truth_manifest_file: The name of the Ground Truth manifest file,
    including the relative path.
    :return: The name of the new Custom Labels manifest file.
    """

    logger.info('Creating manifest file from %s', ground_truth_manifest_file)
    new_manifest_file =
    f'custom_labels_{os.path.basename(ground_truth_manifest_file)}'

    # Read the SageMaker Ground Truth manifest file into memory.
    with open(ground_truth_manifest_file) as gt_file:
        lines = gt_file.readlines()

    # Iterate through the lines one at a time to generate the
    # new lines for the Custom Labels manifest file.
    with open(new_manifest_file, 'w') as the_new_file:
        for line in lines:
            # job_name - The of the Amazon Sagemaker Ground Truth job.
            job_name = ''
            # Load in the old json item from the Ground Truth manifest file
            old_json = json.loads(line)

            # Get the job name
            keys = old_json.keys()
            for key in keys:
                if 'source-ref' not in key and '-metadata' not in key:
                    job_name = key
```

```
new_json = {}
# Set the location of the image
new_json['source-ref'] = old_json['source-ref']

# Temporarily store the list of labels
labels = old_json[job_name]

# Iterate through the labels and reformat to Custom Labels format
for index, label in enumerate(labels):
    new_json[f'{job_name}{index}'] = index
    metadata = {}
    metadata['class-name'] = old_json[f'{job_name}-metadata']['class-
map'][str(label)]
    metadata['confidence'] = old_json[f'{job_name}-metadata']
['confidence-map'][str(label)]
    metadata['type'] = 'groundtruth/image-classification'
    metadata['job-name'] = old_json[f'{job_name}-metadata']['job-name']
    metadata['human-annotated'] = old_json[f'{job_name}-metadata']
['human-annotated']
    metadata['creation-date'] = old_json[f'{job_name}-metadata']
['creation-date']
    # Add the metadata to new json line
    new_json[f'{job_name}{index}-metadata'] = metadata
    # Write the current line to the json file
    the_new_file.write(json.dumps(new_json))
    the_new_file.write('\n')

logger.info('Created %s', new_manifest_file)
return new_manifest_file

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "manifest_file", help="The Amazon SageMaker Ground Truth manifest file"
        "that you want to use."
    )

def main():
```

```
logging.basicConfig(level=logging.INFO,
                    format="%(levelname)s: %(message)s")

try:
    # get command line arguments
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
    add_arguments(parser)
    args = parser.parse_args()
    # Create the manifest file
    manifest_file = create_manifest_file(args.manifest_file)
    print(f'Manifest file created: {manifest_file}')
except FileNotFoundError as err:
    logger.exception('File not found: %s', err)
    print(f'File not found: {err}. Check your manifest file.')

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

2. Notez le nom du nouveau fichier manifeste affiché par le script. Vous l'utiliserez à l'étape suivante.
3. [Chargez les fichiers manifestes](#) dans le compartiment Amazon S3 que vous souhaitez utiliser pour stocker ce type de fichier.

Note

Assurez-vous qu'Étiquettes personnalisées Amazon Rekognition a accès au compartiment Amazon S3 référencé dans le champ `source-ref` des lignes JSON du fichier manifeste. Pour plus d'informations, consultez [Accès à des compartiments Amazon S3 externes](#). Si la tâche Ground Truth stocke des images dans le compartiment de la console Étiquettes personnalisées Amazon Rekognition, vous n'avez pas besoin d'ajouter d'autorisations.

4. Suivez les instructions sous [Création d'un ensemble de données à l'aide d'un fichier manifeste de SageMaker Ground Truth \(Console\)](#) pour créer un jeu de données avec le fichier manifeste chargé. Pour l'étape 8, dans Emplacement du fichier `.manifest`, entrez l'URL Amazon S3 correspondant à l'emplacement du fichier manifeste. Si vous utilisez le kit AWS SDK, effectuez les actions indiquées dans [Création d'un jeu de données à l'aide d'un fichier manifeste \(SDK SageMaker Ground Truth\)](#).

Création d'un fichier manifeste à partir d'un fichier CSV

Cet exemple de script Python simplifie la création d'un fichier manifeste en utilisant un fichier CSV pour étiqueter les images. C'est vous qui créez le fichier CSV. Le fichier manifeste convient à la [classification d'images à plusieurs étiquettes](#) ou [Classification d'images à étiquettes multiples](#). Pour plus d'informations, consultez [Trouvez des objets, des scènes et des concepts](#).

Note

Ce script ne crée pas de fichier manifeste adapté à la [localisation des emplacements d'objets](#) ni à la [localisation des marques](#).

Un fichier manifeste décrit les images utilisées pour entraîner un modèle (par exemple, les emplacements des images et les étiquettes attribuées aux images). Un fichier manifeste est composé d'une ou plusieurs lignes JSON. Chaque ligne JSON décrit une seule image. Pour plus d'informations, consultez [the section called “Étiquettes au niveau de l'image dans les fichiers manifestes”](#).

Un fichier CSV représente des données tabulaires réparties sur plusieurs lignes d'un fichier texte. Les champs sur une ligne sont séparés par une virgule. Pour plus d'informations, consultez la section [Valeurs séparées par des virgules](#). Pour ce script, chaque ligne du fichier CSV représente une image unique et correspond à une ligne JSON dans le fichier manifeste. Pour créer un fichier CSV pour un fichier manifeste prenant en charge la [classification d'images à plusieurs étiquettes](#), vous devez ajouter une ou plusieurs étiquettes au niveau de l'image à chaque ligne. Pour créer un fichier manifeste adapté à la [Image classification](#), vous devez ajouter une seule étiquette au niveau de l'image à chaque ligne.

Par exemple, le fichier CSV suivant décrit les images du projet de mise en route [Classification d'images à étiquettes multiples](#) (Flowers).

```
camellia1.jpg,camellia,with_leaves
camellia2.jpg,camellia,with_leaves
camellia3.jpg,camellia,without_leaves
helleborus1.jpg,helleborus,without_leaves,not_fully_grown
helleborus2.jpg,helleborus,with_leaves,fully_grown
helleborus3.jpg,helleborus,with_leaves,fully_grown
jonquil1.jpg,jonquil,with_leaves
jonquil2.jpg,jonquil,with_leaves
```

```
jonquil3.jpg,jonquil,with_leaves
jonquil4.jpg,jonquil,without_leaves
mauve_honey_myrtle1.jpg,mauve_honey_myrtle,without_leaves
mauve_honey_myrtle2.jpg,mauve_honey_myrtle,with_leaves
mauve_honey_myrtle3.jpg,mauve_honey_myrtle,with_leaves
mediterranean_spurge1.jpg,mediterranean_spurge,with_leaves
mediterranean_spurge2.jpg,mediterranean_spurge,without_leaves
```

Le script génère des lignes JSON pour chaque ligne. Par exemple, voici la ligne JSON pour la première ligne (`camellia1.jpg,camellia,with_leaves`).

```
{"source-ref": "s3://bucket/flowers/train/camellia1.jpg","camellia": 1,"camellia-metadata":{"confidence": 1,"job-name": "labeling-job/camellia","class-name": "camellia","human-annotated": "yes","creation-date": "2022-01-21T14:21:05","type": "groundtruth/image-classification"},"with_leaves": 1,"with_leaves-metadata":{"confidence": 1,"job-name": "labeling-job/with_leaves","class-name": "with_leaves","human-annotated": "yes","creation-date": "2022-01-21T14:21:05","type": "groundtruth/image-classification"}}
```

Dans l'exemple de fichier CSV, le chemin Amazon S3 vers l'image n'est pas présent. Si le fichier CSV n'inclut pas le chemin Amazon S3 des images, utilisez l'argument de ligne de commande `--s3_path` pour spécifier cette information pour l'image.

Le script enregistre la première entrée pour chaque image dans un fichier image CSV dédoublé. Le fichier image CSV dédoublé contient une seule instance de chaque image trouvée dans le fichier CSV d'entrée. Les autres occurrences d'une image dans le fichier CSV d'entrée sont enregistrées dans un fichier image CSV dupliqué. Si le script détecte des images dupliquées, examinez le fichier image CSV dupliqué et mettez à jour le fichier image CSV dédoublé si nécessaire. Réexécutez le script avec le fichier dédoublé. Si aucun doublon n'est trouvé dans le fichier CSV d'entrée, le script supprime le fichier image CSV dédoublé et le fichier image CSV dupliqué, car ils sont vides.

Dans cette procédure, c'est vous qui créez le fichier CSV et qui exécutez le script Python pour créer le fichier manifeste.

Pour créer un fichier manifeste à partir d'un fichier CSV

1. Créez un fichier CSV avec les champs suivants dans chaque ligne (une ligne par image). N'ajoutez pas de ligne d'en-tête au fichier CSV.

Champ 1	Champ 2	Champ n
Nom de l'image ou chemin Amazon S3 de l'image. Par exemple, <code>s3://my-bucket/flowers/train/camellia1.jpg</code> . Vous ne pouvez pas avoir à la fois des images avec le chemin Amazon S3 et des images sans ce chemin.	Première étiquette au niveau de l'image.	Une ou plusieurs étiquettes supplémentaires au niveau de l'image, séparées par des virgules. Ajoutez- en uniquement si vous souhaitez créer un fichier manifeste prenant en charge la classification d'images à plusieurs étiquettes .

Par exemple : `camellia1.jpg,camellia,with_leaves` ou `s3://my-bucket/flowers/train/camellia1.jpg,camellia,with_leaves`

- Enregistrez le fichier CSV.
- Exécutez le script Python suivant. Fournissez les arguments suivants :
 - `csv_file` : fichier CSV que vous avez créé à l'étape 1.
 - `manifest_file` : nom du fichier manifeste que vous souhaitez créer.
 - (Facultatif) `--s3_path s3://path_to_folder/` : chemin Amazon S3 à ajouter aux noms de fichiers image (champ 1). Utilisez `--s3_path` si les images du champ 1 ne contiennent pas déjà de chemin S3.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

from datetime import datetime, timezone
import argparse
import logging
import csv
import os
import json

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service documentation.
```

Shows how to create an image-level (classification) manifest file from a CSV file. You can specify multiple image level labels per image.

CSV file format is

```
image,label,label,..
```

If necessary, use the bucket argument to specify the S3 bucket folder for the images.

<https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-gt-cl-transform.html>

```
"""
```

```
logger = logging.getLogger(__name__)
```

```
def check_duplicates(csv_file, deduplicated_file, duplicates_file):
```

```
    """
```

```
    Checks for duplicate images in a CSV file. If duplicate images
    are found, deduplicated_file is the deduplicated CSV file - only the first
    occurrence of a duplicate is recorded. Other duplicates are recorded in
    duplicates_file.
```

```
    :param csv_file: The source CSV file.
```

```
    :param deduplicated_file: The deduplicated CSV file to create. If no duplicates
    are found
```

```
    this file is removed.
```

```
    :param duplicates_file: The duplicate images CSV file to create. If no
    duplicates are found
```

```
    this file is removed.
```

```
    :return: True if duplicates are found, otherwise false.
```

```
    """
```

```
    logger.info("Deduplicating %s", csv_file)
```

```
    duplicates_found = False
```

```
    # Find duplicates.
```

```
    with open(csv_file, 'r', newline='', encoding="UTF-8") as f,\
        open(deduplicated_file, 'w', encoding="UTF-8") as dedup,\
        open(duplicates_file, 'w', encoding="UTF-8") as duplicates:
```

```
        reader = csv.reader(f, delimiter=',')
```

```
        dedup_writer = csv.writer(dedup)
```

```
        duplicates_writer = csv.writer(duplicates)
```

```
        entries = set()
```

```
        for row in reader:
```

```
        # Skip empty lines.
        if not ''.join(row).strip():
            continue

        key = row[0]
        if key not in entries:
            dedup_writer.writerow(row)
            entries.add(key)
        else:
            duplicates_writer.writerow(row)
            duplicates_found = True

    if duplicates_found:
        logger.info("Duplicates found check %s", duplicates_file)

    else:
        os.remove(duplicates_file)
        os.remove(deduplicated_file)

    return duplicates_found

def create_manifest_file(csv_file, manifest_file, s3_path):
    """
    Reads a CSV file and creates a Custom Labels classification manifest file.
    :param csv_file: The source CSV file.
    :param manifest_file: The name of the manifest file to create.
    :param s3_path: The S3 path to the folder that contains the images.
    """
    logger.info("Processing CSV file %s", csv_file)

    image_count = 0
    label_count = 0

    with open(csv_file, newline='', encoding="UTF-8") as csvfile,\
        open(manifest_file, "w", encoding="UTF-8") as output_file:

        image_classifications = csv.reader(
            csvfile, delimiter=',', quotechar='|')

        # Process each row (image) in CSV file.
        for row in image_classifications:
            source_ref = str(s3_path)+row[0]
```

```
        image_count += 1

        # Create JSON for image source ref.
        json_line = {}
        json_line['source-ref'] = source_ref

        # Process each image level label.
        for index in range(1, len(row)):
            image_level_label = row[index]

            # Skip empty columns.
            if image_level_label == '':
                continue
            label_count += 1

            # Create the JSON line metadata.
            json_line[image_level_label] = 1
            metadata = {}
            metadata['confidence'] = 1
            metadata['job-name'] = 'labeling-job/' + image_level_label
            metadata['class-name'] = image_level_label
            metadata['human-annotated'] = "yes"
            metadata['creation-date'] = \
                datetime.now(timezone.utc).strftime('%Y-%m-%dT%H:%M:%S.%f')
            metadata['type'] = "groundtruth/image-classification"

            json_line[f'{image_level_label}-metadata'] = metadata

            # Write the image JSON Line.
            output_file.write(json.dumps(json_line))
            output_file.write('\n')

    output_file.close()
    logger.info("Finished creating manifest file %s\nImages: %s\nLabels: %s",
                manifest_file, image_count, label_count)

    return image_count, label_count

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """
```

```
parser.add_argument(
    "csv_file", help="The CSV file that you want to process."
)

parser.add_argument(
    "--s3_path", help="The S3 bucket and folder path for the images."
    " If not supplied, column 1 is assumed to include the S3 path.",
    required=False
)

def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        s3_path = args.s3_path
        if s3_path is None:
            s3_path = ''

        # Create file names.
        csv_file = args.csv_file
        file_name = os.path.splitext(csv_file)[0]
        manifest_file = f'{file_name}.manifest'
        duplicates_file = f'{file_name}-duplicates.csv'
        deduplicated_file = f'{file_name}-deduplicated.csv'

        # Create manifest file, if there are no duplicate images.
        if check_duplicates(csv_file, deduplicated_file, duplicates_file):
            print(f"Duplicates found. Use {duplicates_file} to view duplicates "
                  f"and then update {deduplicated_file}. ")
            print(f"{deduplicated_file} contains the first occurrence of a
duplicate. "
                  "Update as necessary with the correct label information.")
            print(f"Re-run the script with {deduplicated_file}")
        else:
```

```
print("No duplicates found. Creating manifest file.")

image_count, label_count = create_manifest_file(csv_file,
                                                manifest_file,
                                                s3_path)

print(f"Finished creating manifest file: {manifest_file} \n"
      f"Images: {image_count}\nLabels: {label_count}")

except FileNotFoundError as err:
    logger.exception("File not found: %s", err)
    print(f"File not found: {err}. Check your input CSV file.")

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

4. Si vous prévoyez d'utiliser un jeu de données de test, répétez les étapes 1 à 3 afin de créer un fichier manifeste pour le jeu de données de test.
5. Si nécessaire, copiez les images dans le chemin du compartiment Amazon S3 que vous avez spécifié dans la colonne 1 du fichier CSV (ou dans la ligne de commande `--s3_path`). Vous pouvez utiliser la commande S3 AWS suivante :

```
aws s3 cp --recursive your-local-folder s3://your-target-S3-location
```

6. [Chargez les fichiers manifestes](#) dans le compartiment Amazon S3 que vous souhaitez utiliser pour stocker ce type de fichier.

Note

Assurez-vous qu'Étiquettes personnalisées Amazon Rekognition a accès au compartiment Amazon S3 référencé dans le champ `source-ref` des lignes JSON du fichier manifeste. Pour plus d'informations, consultez [Accès à des compartiments Amazon S3 externes](#). Si la tâche Ground Truth stocke des images dans le compartiment de la console Étiquettes personnalisées Amazon Rekognition, vous n'avez pas besoin d'ajouter d'autorisations.

7. Suivez les instructions sous [Création d'un ensemble de données à l'aide d'un fichier manifeste de SageMaker Ground Truth \(Console\)](#) pour créer un jeu de données avec le fichier manifeste

chargé. Pour l'étape 8, dans Emplacement du fichier .manifest, entrez l'URL Amazon S3 correspondant à l'emplacement du fichier manifeste. Si vous utilisez le kit AWS SDK, effectuez les actions indiquées dans [Création d'un jeu de données à l'aide d'un fichier manifeste \(SDK SageMaker Ground Truth\)](#).

Jeu de données

Si vous avez déjà créé un jeu de données, vous pouvez copier son contenu dans un nouveau jeu de données. Pour créer un jeu de données à partir d'un jeu de données existant à l'aide du kit AWS SDK, consultez [Création d'un jeu de données à l'aide d'un jeu de données existant \(SDK\)](#).

Pour créer un jeu de données à partir d'un jeu de données Étiquettes personnalisées Amazon Rekognition existant (console)

1. Ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
2. Choisissez Utiliser Custom Labels.
3. Choisissez Démarrer.
4. Dans le volet de navigation de gauche, choisissez Projets.
5. Sur la page Projets, choisissez le projet auquel vous souhaitez ajouter un jeu de données. La page de détails de ce projet s'affiche.
6. Choisissez Créer un jeu de données. La page Créer un jeu de données s'affiche.
7. Dans Démarrage de la configuration, choisissez Démarrer avec un seul jeu de données ou Démarrer avec un jeu de données d'entraînement. Pour créer un modèle de meilleure qualité, nous vous recommandons de commencer par un jeu de données d'entraînement et un jeu de données de test distincts.

Single dataset

- a. Dans la section Détails du jeu de données d'entraînement, choisissez Copier un jeu de données Étiquettes personnalisées Amazon Rekognition existant.
- b. Dans la section Détails du jeu de données d'entraînement, dans la zone d'édition Jeu de données, tapez ou sélectionnez le nom du jeu de données que vous souhaitez copier.
- c. Choisissez Créer un jeu de données. La page des jeux de données de votre projet s'ouvre.

Separate training and test datasets

- a. Dans la section Détails du jeu de données d'entraînement, choisissez Copier un jeu de données Étiquettes personnalisées Amazon Rekognition existant.
- b. Dans la section Détails du jeu de données d'entraînement, dans la zone d'édition Jeu de données, tapez ou sélectionnez le nom du jeu de données que vous souhaitez copier.
- c. Dans la section Détails du jeu de données de test, choisissez Copier un jeu de données Étiquettes personnalisées Amazon Rekognition existant.
- d. Dans la section Détails du jeu de données de test, dans la zone d'édition Jeu de données, tapez ou sélectionnez le nom du jeu de données que vous souhaitez copier.

Note

Le jeu de données d'entraînement et le jeu de données de test peuvent avoir différentes sources d'images.

- e. Choisissez Créer des jeux de données. La page des jeux de données de votre projet s'ouvre.
8. Si vous devez ajouter ou modifier des étiquettes, effectuez les actions indiquées dans [Étiquetage des images](#).
 9. Suivez les étapes décrites dans [Entraînement d'un modèle \(console\)](#) pour entraîner le modèle.

Étiquetage des images

Une étiquette identifie un objet, une scène, un concept ou un cadre de délimitation autour d'un objet dans une image. Par exemple, si votre jeu de données contient des images de chiens, vous pouvez ajouter des étiquettes pour les races de chiens.

Après avoir importé vos images dans un jeu de données, vous devrez peut-être ajouter des étiquettes à des images ou corriger des images mal étiquetées. Par exemple, les images ne sont pas étiquetées si elles sont importées depuis un ordinateur local. La galerie du jeu de données vous permet d'ajouter de nouvelles étiquettes au jeu de données et d'attribuer des étiquettes et des cadres de délimitation à ses images.

La façon dont vous étiquetez les images des jeux de données détermine le type de modèle entraîné par Étiquettes personnalisées Amazon Rekognition. Pour plus d'informations, consultez [Utilisation des jeux de données](#).

Rubriques

- [Gestion des étiquettes](#)
- [Attribution d'étiquettes au niveau de l'image à une image](#)
- [Étiquetage des objets à l'aide de cadres de délimitation](#)

Gestion des étiquettes

Vous pouvez gérer les étiquettes à l'aide de la console Étiquettes personnalisées Amazon Rekognition. Il n'existe pas d'API spécifique pour gérer les étiquettes : les étiquettes sont ajoutées au jeu de données lorsque vous le créez avec `CreateDataset` ou lorsque vous ajoutez d'autres images au jeu de données avec `UpdateDatasetEntries`.

Rubriques

- [Gestion des étiquettes \(console\)](#)
- [Gestion des étiquettes \(kit SDK\)](#)

Gestion des étiquettes (console)

Vous pouvez utiliser la console Étiquettes personnalisées Amazon Rekognition pour ajouter, modifier ou supprimer des étiquettes dans un jeu de données. Pour ajouter une étiquette à un jeu de données, vous pouvez ajouter une étiquette que vous créez ou importez des étiquettes à partir d'un jeu de données existant dans Rekognition.

Rubriques

- [Ajout de nouvelles étiquettes \(console\)](#)
- [Modification et suppression d'étiquettes \(console\)](#)

Ajout de nouvelles étiquettes (console)

Vous pouvez spécifier les nouvelles étiquettes que vous souhaitez ajouter au jeu de données.

Ajout d'étiquettes à l'aide de la fenêtre d'édition

Pour ajouter une nouvelle étiquette (console)

1. Ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
2. Choisissez Utiliser Custom Labels.
3. Choisissez Démarrer.
4. Dans le volet de navigation de gauche, choisissez Projets.
5. Sur la page Projets, choisissez le projet que vous voulez utiliser. La page de détails de ce projet s'affiche.
6. Si vous souhaitez ajouter des étiquettes au jeu de données d'entraînement, choisissez l'onglet Entraînement. Sinon, choisissez l'onglet Test pour ajouter des étiquettes au jeu de données de test.
7. Choisissez Commencer l'étiquetage pour passer en mode d'étiquetage.
8. Dans la section Étiquettes de la galerie de jeux de données, choisissez Gérer les étiquettes pour ouvrir la boîte de dialogue correspondante.
9. Dans la zone d'édition, saisissez un nouveau nom d'étiquette.
10. Choisissez Ajouter une étiquette.
11. Répétez les étapes 9 et 10 jusqu'à ce que vous ayez créé toutes les étiquettes dont vous avez besoin.
12. Choisissez Enregistrer pour enregistrer les étiquettes que vous avez ajoutées.

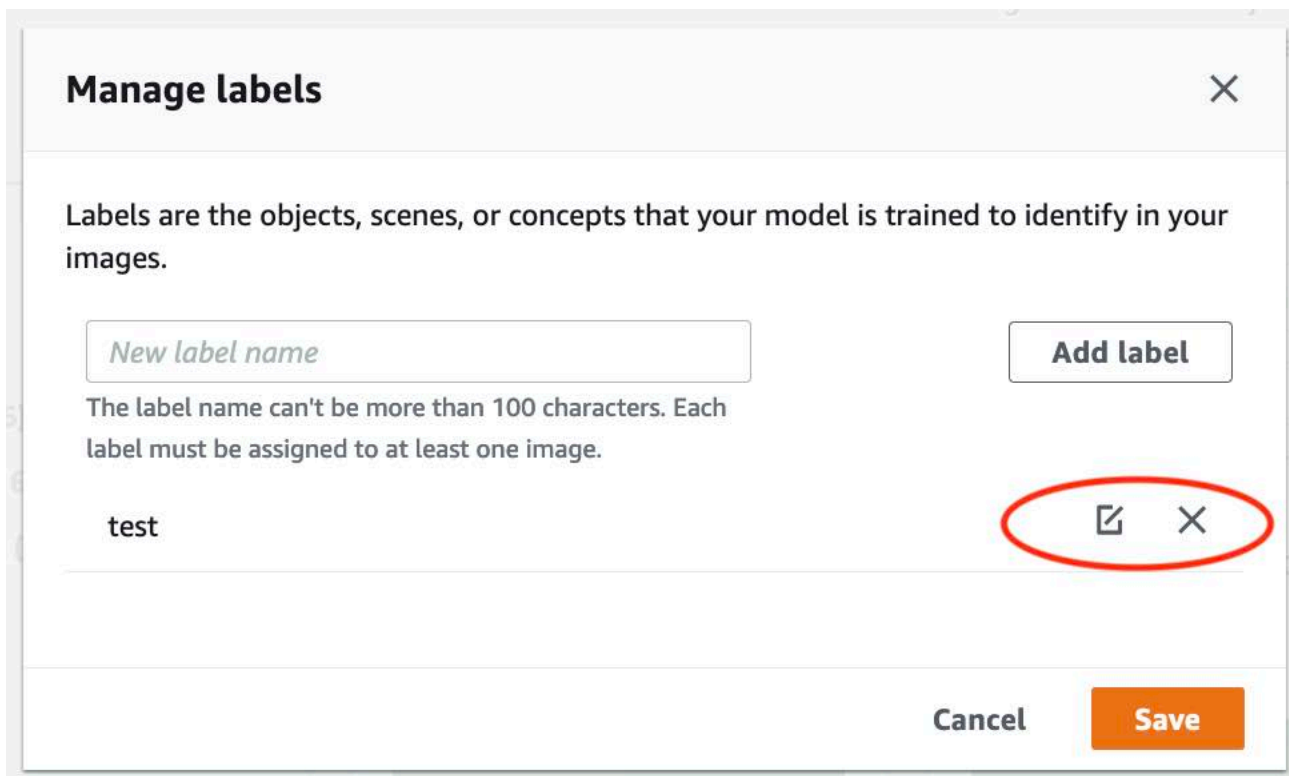
Modification et suppression d'étiquettes (console)

Vous pouvez renommer ou supprimer des étiquettes après les avoir ajoutées à un jeu de données. Vous ne pouvez supprimer que les étiquettes qui ne sont attribuées à aucune image.

Pour renommer ou supprimer une étiquette existante (console)

1. Ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
2. Choisissez Utiliser Custom Labels.
3. Choisissez Démarrer.
4. Dans le volet de navigation de gauche, choisissez Projets.

5. Sur la page Projets, choisissez le projet que vous voulez utiliser. La page de détails de ce projet s'affiche.
6. Si vous souhaitez modifier ou supprimer des étiquettes dans le jeu de données d'entraînement, cliquez sur l'onglet Entraînement. Sinon, choisissez l'onglet Test pour modifier ou supprimer des étiquettes dans le jeu de données de test.
7. Choisissez Commencer l'étiquetage pour passer en mode d'étiquetage.
8. Dans la section Étiquettes de la galerie de jeux de données, choisissez Gérer les étiquettes pour ouvrir la boîte de dialogue correspondante.
9. Choisissez l'étiquette que vous souhaitez modifier ou supprimer.



- a. Si vous choisissez l'icône de suppression (X), l'étiquette est supprimée de la liste.
 - b. Si vous souhaitez modifier l'étiquette, choisissez l'icône d'édition (crayon et bloc-notes) et entrez un nouveau nom d'étiquette dans la zone d'édition.
10. Choisissez Save pour enregistrer les changements.

Gestion des étiquettes (kit SDK)

Il n'existe pas d'API unique qui gère les étiquettes des jeux de données. Si vous créez un jeu de données avec `CreateDataset`, les étiquettes présentes dans le fichier manifeste ou dans le jeu

de données copié, créez l'ensemble initial d'étiquettes. Si vous ajoutez d'autres images avec l'API `UpdateDatasetEntries`, les nouvelles étiquettes présentes dans les entrées sont ajoutées au jeu de données. Pour plus d'informations, consultez [Ajouter d'autres images \(SDK\)](#). Pour supprimer des étiquettes d'un jeu de données, vous devez supprimer toutes les annotations d'étiquettes du jeu de données.

Pour supprimer des étiquettes d'un jeu de données

1. Appelez `ListDatasetEntries` pour obtenir les entrées du jeu de données. Pour obtenir un exemple de code, consultez [Répertorier les entrées du jeu de données \(SDK\)](#).
2. Dans le fichier, supprimez toutes les annotations d'étiquette. Pour plus d'informations, consultez [Étiquettes au niveau de l'image dans les fichiers manifestes](#) et [the section called "Localisation d'objets dans les fichiers manifestes"](#).
3. Utilisez ce fichier pour mettre à jour le jeu de données avec l'API `UpdateDatasetEntries`. Pour plus d'informations, consultez [Ajouter d'autres images \(SDK\)](#).

Attribution d'étiquettes au niveau de l'image à une image

Les étiquettes au niveau de l'image vous permettent d'entraîner des modèles qui classent les images dans des catégories. Une étiquette au niveau de l'image indique qu'une image contient un objet, une scène ou un concept. Par exemple, l'image suivante présente une rivière. Si votre modèle classe les images comme contenant des rivières, vous devez ajouter une étiquette rivière au niveau de l'image. Pour plus d'informations, consultez [Utilisation des jeux de données](#).



Un jeu de données contenant des étiquettes au niveau de l'image doit comporter au moins deux étiquettes. Chaque image doit comporter au moins une étiquette qui identifie son objet, sa scène ou son concept.

Pour attribuer des étiquettes au niveau de l'image à une image (console)

1. Ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
2. Choisissez Utiliser Custom Labels.
3. Choisissez Démarrer.
4. Dans le volet de navigation de gauche, choisissez Projets.
5. Sur la page Projets, choisissez le projet que vous voulez utiliser. La page de détails de ce projet s'affiche.
6. Dans le volet de navigation de gauche, choisissez Dataset.

7. Si vous souhaitez ajouter des étiquettes au jeu de données d'entraînement, choisissez l'onglet Entraînement. Sinon, choisissez l'onglet Test pour ajouter des étiquettes au jeu de données de test.
8. Choisissez Commencer l'étiquetage pour passer en mode d'étiquetage.
9. Dans la galerie d'images, sélectionnez une ou plusieurs images auxquelles vous souhaitez ajouter des étiquettes. Vous ne pouvez sélectionner des images que sur une seule page à la fois. Pour sélectionner une plage contiguë d'images sur une page :
 - a. Sélectionnez la première image de la plage.
 - b. Appuyez sur la touche Maj et maintenez-la enfoncée.
 - c. Sélectionnez la dernière plage d'images. Les images situées entre la première et la deuxième image sont également sélectionnées.
 - d. Relâchez la touche Maj.
10. Choisissez Attribuer des étiquettes au niveau de l'image.
11. Dans la boîte de dialogue Attribuer une étiquette au niveau de l'image aux images sélectionnées, sélectionnez une étiquette que vous souhaitez attribuer à l'image ou aux images.
12. Choisissez Attribuer pour attribuer l'étiquette à l'image.
13. Répétez l'étiquetage jusqu'à ce que chaque image soit annotée avec les étiquettes requises.
14. Choisissez Enregistrer les Modifications pour enregistrer vos Modifications.

Attribution d'étiquettes au niveau de l'image (kit SDK)

Vous pouvez utiliser l'API `UpdateDatasetEntries` pour ajouter ou mettre à jour les étiquettes au niveau de l'image qui sont attribuées à une image. `UpdateDatasetEntries` utilise une ou plusieurs lignes JSON. Chaque ligne JSON représente une seule image. Pour une image avec une étiquette au niveau de l'image, la ligne JSON ressemble à ce qui suit.

```
{"source-ref":"s3://custom-labels-console-us-east-1-nnnnnnnnnn/gt-job/manifest/IMG_1133.png","TestCLConsoleBucket":0,"TestCLConsoleBucket-metadata":{"confidence":0.95,"job-name":"labeling-job/testclconsolebucket","class-name":"Echo Dot","human-annotated":"yes","creation-date":"2020-04-15T20:17:23.433061","type":"groundtruth/image-classification"}}
```

Le champ `source-ref` indique l'emplacement de l'image. La ligne JSON inclut également les étiquettes au niveau de l'image attribuées à l'image. Pour plus d'informations, consultez [the section called "Étiquettes au niveau de l'image dans les fichiers manifestes"](#).

Pour attribuer des étiquettes au niveau de l'image à une image

1. Pour obtenir la ligne get JSON correspondant à l'image existante, utilisez `ListDatasetEntries`. Pour le champ `source-ref`, spécifiez l'emplacement de l'image à laquelle vous souhaitez attribuer l'étiquette. Pour plus d'informations, consultez [Répertorier les entrées du jeu de données \(SDK\)](#).
2. Mettez à jour la ligne JSON renvoyée à l'étape précédente à l'aide des informations indiquées dans [Étiquettes au niveau de l'image dans les fichiers manifestes](#).
3. Appelez `UpdateDatasetEntries` pour mettre à jour l'image. Pour plus d'informations, consultez [Ajouter d'autres images à un jeu de données](#).

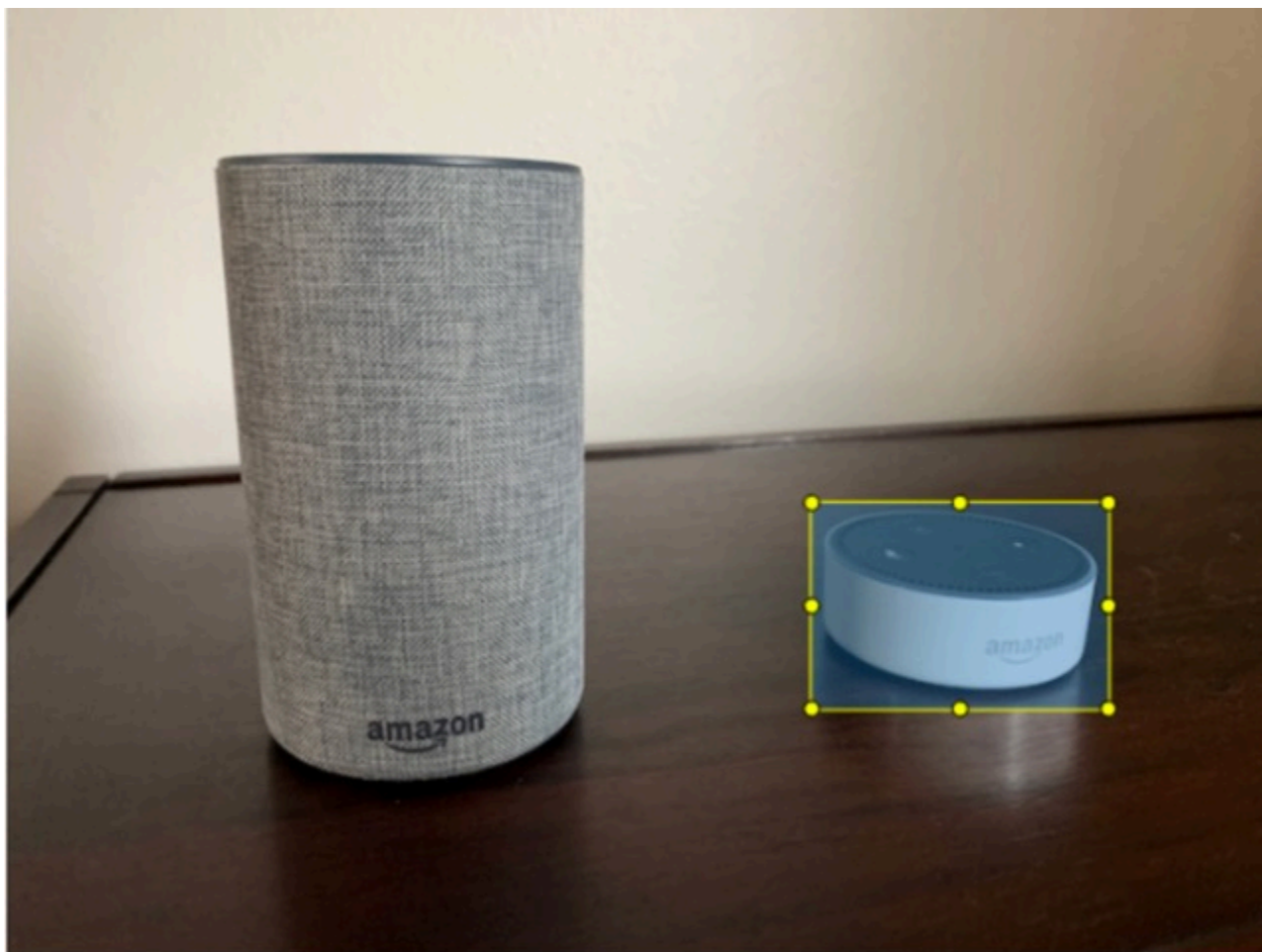
Étiquetage des objets à l'aide de cadres de délimitation

Si vous souhaitez que votre modèle détecte l'emplacement des objets dans une image, vous devez identifier la nature de l'objet et son emplacement dans l'image. Un cadre de délimitation isole un objet dans une image. Les cadres de délimitation vous permettent d'entraîner un modèle à détecter différents objets dans la même image. Vous identifiez l'objet en attribuant une étiquette au cadre de délimitation.

Note

Si vous entraînez un modèle à rechercher des objets, des scènes et des concepts avec des étiquettes au niveau de l'image, vous n'avez pas besoin de suivre cette étape.

Par exemple, si vous souhaitez entraîner un modèle qui détecte les appareils Amazon Echo Dot, vous devez dessiner un cadre de délimitation autour de chaque appareil Echo Dot sur une image et attribuer une étiquette nommée Echo Dot au cadre de délimitation. L'image suivante affiche un cadre de délimitation autour d'un appareil Echo Dot. L'image contient également un appareil Amazon Echo sans cadre de délimitation.



Localisation d'objets à l'aide de cadres de délimitation (console)

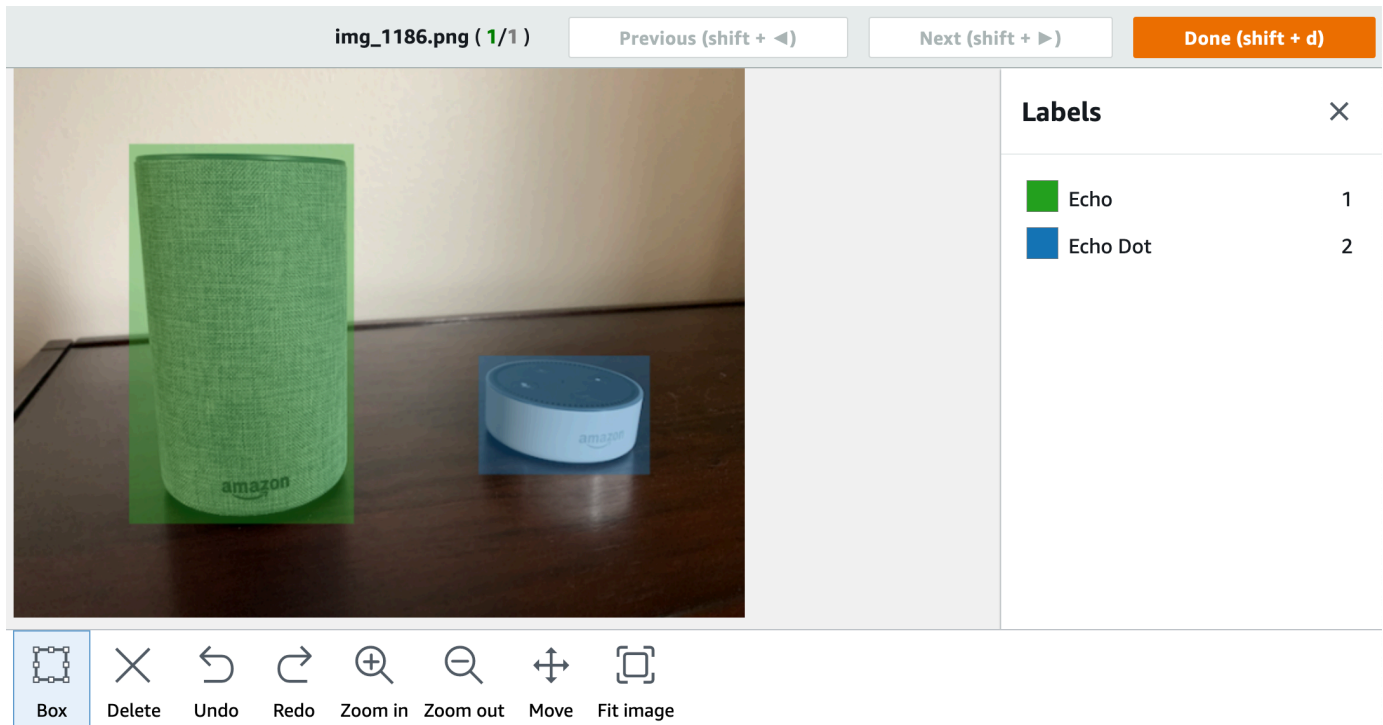
Dans cette procédure, vous utilisez la console pour dessiner des cadres de délimitation autour des objets dans vos images. Vous pouvez également identifier des objets dans l'image en attribuant des étiquettes au cadre de délimitation.

Note

Vous ne pouvez pas utiliser le navigateur Safari pour ajouter des cadres de délimitation aux images. Pour déterminer quels navigateurs sont pris en charge, consultez [Configuration d'Étiquettes personnalisées Amazon Rekognition](#).

Avant de pouvoir ajouter des cadres de délimitation, vous devez ajouter au moins une étiquette au jeu de données. Pour plus d'informations, consultez [Ajout de nouvelles étiquettes \(console\)](#).

1. Ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
2. Choisissez Utiliser Custom Labels.
3. Choisissez Démarrer.
4. Dans le volet de navigation de gauche, choisissez Projets.
5. Sur la page Projets, choisissez le projet que vous voulez utiliser. La page de détails de ce projet s'affiche.
6. Sur la page des détails du projet, choisissez Étiqueter des images.
7. Si vous souhaitez ajouter des cadres de délimitation aux images du jeu de données d'entraînement, choisissez l'onglet Entraînement. Sinon, choisissez l'onglet Test pour ajouter des cadres de délimitation aux images du jeu de données de test.
8. Choisissez Commencer l'étiquetage pour passer en mode d'étiquetage.
9. Dans la galerie d'images, choisissez les images auxquelles vous souhaitez ajouter des cadres de délimitation.
10. Choisissez Dessiner un cadre de délimitation. Une série de conseils apparaissent avant l'affichage de l'éditeur de cadres de délimitation.
11. Dans le volet Étiquettes de droite, sélectionnez l'étiquette que vous souhaitez attribuer à un cadre de délimitation.
12. Dans l'outil de dessin, placez le pointeur en haut à gauche de l'objet souhaité.
13. Appuyez sur le bouton gauche de la souris et tracez un cadre autour de l'objet. Essayez de dessiner le cadre de délimitation le plus près possible de l'objet.
14. Relâchez le bouton de la souris. Le cadre de délimitation est mis en évidence.
15. Choisissez Suivant si vous avez d'autres images à étiqueter. Sinon, choisissez OK pour terminer l'étiquetage.



16. Répétez les étapes 1 à 7 jusqu'à ce que vous ayez créé un cadre de délimitation dans chaque image contenant des objets.
17. Choisissez Enregistrer les Modifications pour enregistrer vos Modifications.
18. Choisissez Quitter pour quitter le mode d'étiquetage.

Localisation d'objets à l'aide de cadres de délimitation (kit SDK)

Vous pouvez utiliser l'API `UpdateDatasetEntries` pour ajouter ou mettre à jour les informations de localisation d'un objet pour une image. `UpdateDatasetEntries` utilise une ou plusieurs lignes JSON. Chaque ligne JSON représente une seule image. Pour la localisation d'objets, une ligne JSON ressemble à ce qui suit.

```
{
  "source-ref": "s3://bucket/images/IMG_1186.png",
  "bounding-box": {
    "image_size": [
      {"width": 640, "height": 480, "depth": 3}
    ],
    "annotations": [
      {
        "class_id": 1,
        "top": 251,
        "left": 399,
        "width": 155,
        "height": 101
      },
      {
        "class_id": 0,
        "top": 65,
        "left": 86,
        "width": 220,
        "height": 334
      }
    ],
    "bounding-box-metadata": {
      "objects": [
        {"confidence": 1},
        {"confidence": 1}
      ],
      "class-map": {
        "0": "Echo",
        "1": "Echo Dot"
      },
      "type": "groundtruth/object-detection",
      "human-annotated": "yes",
      "creation-date": "2013-11-18T02:53:27",
      "job-name": "my job"
    }
  }
}
```

Le champ `source-ref` indique l'emplacement de l'image. La ligne JSON inclut également des cadres de délimitation étiquetés pour chaque objet de l'image. Pour plus d'informations, consultez [the section called "Localisation d'objets dans les fichiers manifestes"](#).

Pour attribuer des cadres de délimitation à une image

1. Pour obtenir la ligne get JSON correspondant à l'image existante, utilisez `ListDatasetEntries`. Pour le champ `source-ref`, spécifiez l'emplacement de l'image à laquelle vous souhaitez attribuer l'étiquette au niveau de l'image. Pour plus d'informations, consultez [Répertoire des entrées du jeu de données \(SDK\)](#).
2. Mettez à jour la ligne JSON renvoyée à l'étape précédente à l'aide des informations indiquées dans [Localisation d'objets dans les fichiers manifestes](#).
3. Appelez `UpdateDatasetEntries` pour mettre à jour l'image. Pour plus d'informations, consultez [Ajouter d'autres images à un jeu de données](#).

Débogage des jeux de données

Lors de la création d'un jeu de données, deux types d'erreurs peuvent se produire : les erreurs définitives et les erreurs non définitives. Les erreurs définitives peuvent arrêter la création ou la mise à jour du jeu de données. Les erreurs non définitives n'interrompent pas la création ni la mise à jour du jeu de données.

Rubriques

- [Erreurs définitives](#)
- [Erreurs non définitives](#)

Erreurs définitives

Il existe deux types d'erreurs définitives : les erreurs de fichier qui entraînent l'échec de la création du jeu de données et les erreurs de contenu qu'Étiquettes personnalisées Amazon Rekognition supprime du jeu de données. La création du jeu de données échoue s'il y a trop d'erreurs de contenu.

Rubriques

- [Erreurs de fichier définitives](#)
- [Erreurs définitives de contenu](#)

Erreurs de fichier définitives

Voici des exemples d'erreurs de fichier. Vous pouvez obtenir des informations sur les erreurs de fichier en appelant `DescribeDataset` et en vérifiant les champs `Status` et `StatusMessage`. Pour obtenir un exemple de code, consultez [Décrire un ensemble de données \(SDK\)](#).

- [ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT](#)
- [ERROR_MANIFEST_SIZE_TOO_LARGE](#).
- [ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM](#)
- [ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET](#)
- [ERROR_TOO_MANY_RECORDS_IN_ERROR](#)
- [ERROR_MANIFEST_TOO_MANY_LABELS](#)
- [ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_DISTRIBUTE](#)

ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT

Message d'erreur

Le contenu ou l'extension du fichier manifeste ne sont pas valides.

Le fichier manifeste d'entraînement ou de test n'a pas d'extension de fichier ou son contenu n'est pas valide.

Pour corriger l'erreur `ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT`

- Vérifiez les causes possibles suivantes dans les fichiers manifestes d'entraînement et de test.
 - Le fichier manifeste ne comporte aucune extension de fichier. Par convention, l'extension de fichier est `.manifest`.
 - Impossible de trouver la clé ou le compartiment Amazon S3 du fichier manifeste.

ERROR_MANIFEST_SIZE_TOO_LARGE

Error message (Message d'erreur)

La taille du fichier manifeste dépasse la taille maximale prise en charge.

La taille (en octets) du fichier manifeste d'entraînement ou de test est trop importante. Pour plus d'informations, consultez [Amazon Rekognition](#). Un fichier manifeste peut comporter un nombre

de lignes JSON inférieur au nombre maximal autorisé, mais pourtant dépasser la taille de fichier maximale.

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger l'erreur La taille du fichier manifeste dépasse la taille maximale prise en charge.

Pour corriger l'erreur **ERROR_MANIFEST_SIZE_TOO_LARGE**

1. Vérifiez quels manifestes d'entraînement et de test dépassent la taille de fichier maximale.
2. Réduisez le nombre de lignes JSON dans les fichiers manifestes dont la taille est trop importante. Pour plus d'informations, consultez [Création d'un fichier manifeste](#).

ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM

Error message (Message d'erreur)

Le fichier manifeste contient trop de lignes.

En savoir plus

Le nombre de lignes JSON (nombre d'images) dans le fichier manifeste est supérieur à la limite autorisée. Cette limite est différente pour les modèles au niveau de l'image et les modèles de localisation des objets. Pour plus d'informations, consultez [Amazon Rekognition](#).

Les erreurs de ligne JSON sont validées jusqu'à ce que le nombre de lignes JSON atteigne la limite **ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM**.

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger une erreur **ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM**.

Pour corriger l'erreur **ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM**

- Réduisez le nombre de lignes JSON dans le manifeste. Pour plus d'informations, consultez [Création d'un fichier manifeste](#).

ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET

Error message (Message d'erreur)

Les autorisations du compartiment S3 sont incorrectes.

La fonctionnalité Étiquettes personnalisées Amazon Rekognition n'est pas autorisée à accéder à un ou plusieurs compartiments contenant les fichiers manifestes d'entraînement et de test.

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

Pour corriger l'erreur **ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET**

- Vérifiez les autorisations du ou des compartiments contenant les fichiers manifestes d'entraînement et de test. Pour plus d'informations, consultez [Étape 2 : Configurer les autorisations d'accès à la console Étiquettes personnalisées Amazon Rekognition](#).

ERROR_TOO_MANY_RECORDS_IN_ERROR

Error message (Message d'erreur)

Le fichier manifeste contient trop d'erreurs définitives.

Pour corriger l'erreur **ERROR_TOO_MANY_RECORDS_IN_ERROR**

- Réduisez le nombre de lignes JSON (images) présentant des erreurs définitives de contenu. Pour plus d'informations, consultez [Erreurs de contenu du manifeste du terminal](#).

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

ERROR_MANIFEST_TOO_MANY_LABELS

Error message (Message d'erreur)

Le fichier manifeste contient trop d'étiquettes.

En savoir plus

Le nombre d'étiquettes uniques dans le manifeste (jeu de données) est supérieur à la limite autorisée. Si le jeu de données d'entraînement est fractionné pour créer un jeu de données de test, le nombre d'étiquettes est déterminé après le fractionnement.

Pour corriger l'erreur ERROR_MANIFEST_TOO_MANY_LABELS (Console)

- Supprimez des étiquettes du jeu de données. Pour plus d'informations, consultez [Gestion des étiquettes](#). Les étiquettes sont automatiquement supprimées des images et des cadres de délimitation du jeu de données.

Pour corriger l'erreur ERROR_MANIFEST_TOO_MANY_LABELS (Ligne JSON)

- Manifestes avec lignes JSON au niveau de l'image : si l'image possède une seule étiquette, supprimez les lignes JSON des images utilisant l'étiquette souhaitée. Si la ligne JSON contient plusieurs étiquettes, supprimez uniquement l'objet JSON correspondant à l'étiquette souhaitée. Pour plus d'informations, consultez [Ajout de plusieurs étiquettes au niveau de l'image à une image](#).

Manifestes avec lignes JSON d'emplacement d'objet : supprimez le cadre de délimitation et les informations associées à l'étiquette que vous souhaitez supprimer. Procédez ainsi pour chaque ligne JSON contenant l'étiquette souhaitée. Vous devez supprimer l'étiquette du tableau `class-map` et les objets correspondants dans le tableau `objects` et `annotations`. Pour plus d'informations, consultez [Localisation d'objets dans les fichiers manifestes](#).

ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_DISTRIBUTE

Error message (Message d'erreur)

Le fichier manifeste ne contient pas suffisamment d'images étiquetées pour distribuer le jeu de données.

La distribution des jeux de données se produit lorsqu'Étiquettes personnalisées Amazon Rekognition fractionne un jeu de données d'entraînement pour créer un jeu de données de test. Vous pouvez également fractionner un jeu de données en appelant l'API `DistributeDatasetEntries`.

Pour corriger l'erreur ERROR_MANIFEST_TOO_MANY_LABELS

- Ajout d'autres images étiquetées au jeu de données d'entraînement

Erreurs définitives de contenu

Voici quelques exemples d'erreurs définitives de contenu. Lors de la création d'un jeu de données, les images présentant des erreurs définitives de contenu sont supprimées du jeu de données. Le jeu de données peut continuer à être utilisé pour l'entraînement. S'il y a trop d'erreurs de contenu, le jeu de données/la mise à jour échoue. Les erreurs définitives de contenu liées aux opérations du jeu de données ne sont pas affichées dans la console ni renvoyées par `DescribeDataset` ou une autre API. Si vous remarquez que des images ou des annotations sont absentes de vos jeux de données, vérifiez les fichiers manifestes de ces derniers pour détecter les problèmes suivants :

- La longueur d'une ligne JSON est trop longue. La longueur maximale est de 100 000 caractères.
- La valeur `source-ref` est absente d'une ligne JSON.
- Le format d'une valeur `source-ref` dans une ligne JSON n'est pas valide.
- Le contenu d'une ligne JSON n'est pas valide.
- La valeur d'un champ `source-ref` apparaît plusieurs fois. Une image ne peut être référencée qu'une seule fois dans un jeu de données.

Pour plus d'informations sur le champ `source-ref`, consultez [Création d'un fichier manifeste](#).

Erreurs non définitives

Les erreurs non définitives qui peuvent se produire lors de la création ou de la mise à jour du jeu de données sont les suivantes. Ces erreurs peuvent invalider une ligne JSON entière ou des annotations au sein d'une ligne JSON. Si une ligne JSON contient une erreur, elle n'est pas utilisée pour l'entraînement. Si une annotation dans une ligne JSON contient une erreur, la ligne JSON continue à être utilisée pour l'entraînement, mais sans l'annotation corrompue. Pour plus d'informations sur les lignes JSON, consultez [Création d'un fichier manifeste](#).

Vous pouvez accéder aux erreurs non définitives depuis la console et en appelant l'API `ListDatasetEntries`. Pour plus d'informations, consultez [Répertoire des entrées du jeu de données \(SDK\)](#).

Les erreurs suivantes sont également renvoyées pendant l'entraînement. Nous vous recommandons de corriger ces erreurs avant d'entraîner votre modèle. Pour plus d'informations, consultez [Erreurs de validation de lignes JSON non terminales](#).

- [ERROR_NO_LABEL_ATTRIBUTES](#)
- [ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT](#)

- [ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT](#)
- [ERROR_NO_VALID_LABEL_ATTRIBUTES](#)
- [BOÎTE DE DÉLIMITATION D'ERREUR NON VALIDE](#)
- [DIMENSION DE L'IMAGE ERROR_NON VALIDE](#)
- [ERROR_BOUNDING_BOX_TOO_SMALL](#)
- [ERREUR_NO_VALIDATION_ANNOTATIONS](#)
- [ERROR_MISSING_BOUNDING_BOX_CONFIDENCE](#)
- [ERROR_MISSING_CLASS_MAP_ID](#)
- [ERREUR_TOO_MANY_BOUNDING_BOXES](#)
- [TYPE DE CAS D'UTILISATION D'ERREUR NON PRIS EN CHARGE](#)
- [ERROR_INVALID_LABEL_NAME_LENGTH](#)

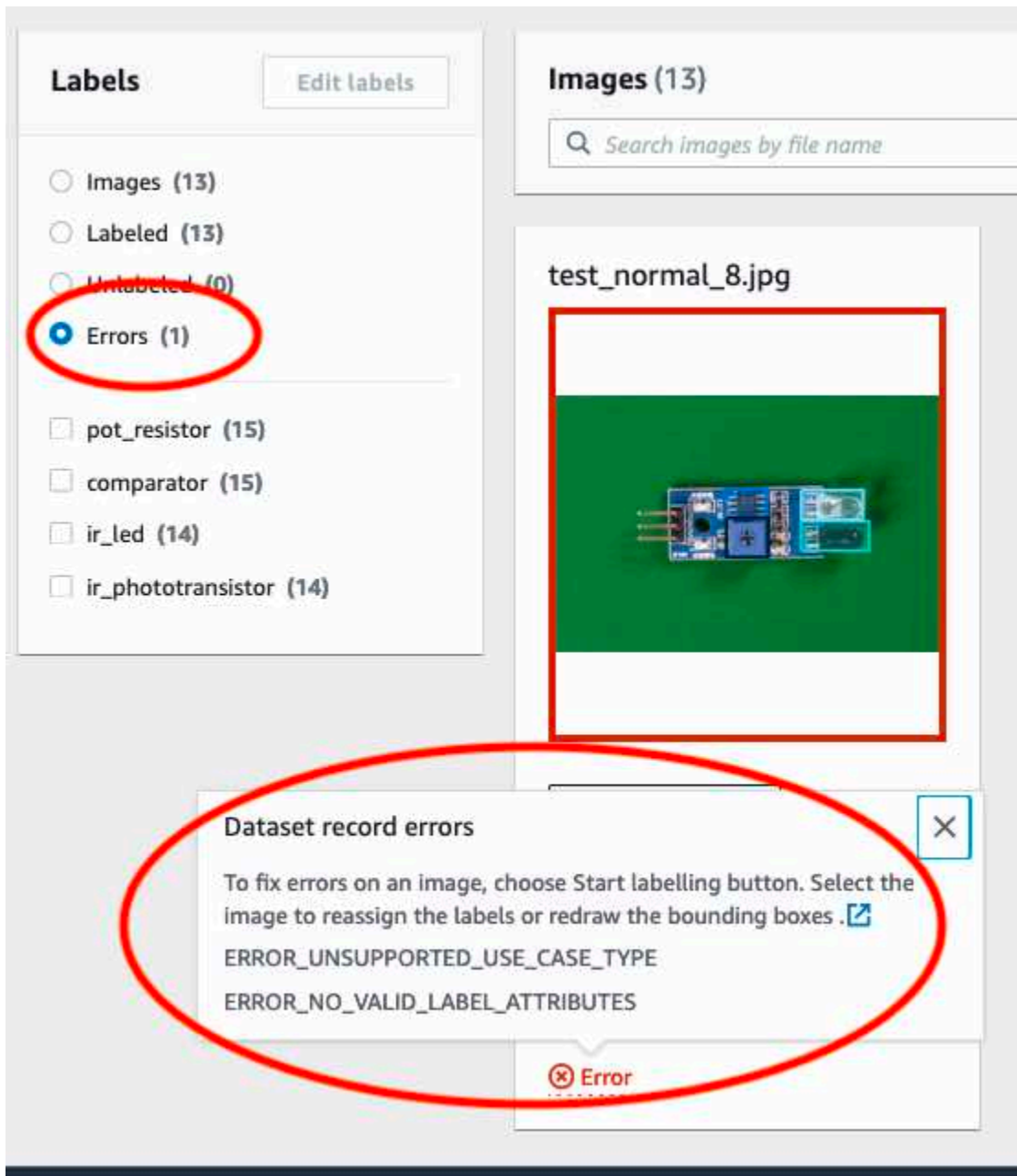
Accès aux erreurs non définitives

Vous pouvez utiliser la console pour déterminer quelles images d'un jeu de données présentent des erreurs non définitives. Vous pouvez également appeler l'API `ListDatasetEntries` pour obtenir les messages d'erreur. Pour plus d'informations, consultez [Répertoire des entrées du jeu de données \(SDK\)](#).

Pour accéder aux erreurs non définitives (console)

1. Ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
2. Choisissez Utiliser Custom Labels.
3. Choisissez Démarrer.
4. Dans le volet de navigation de gauche, choisissez Projets.
5. Sur la page Projets, choisissez le projet que vous voulez utiliser. La page de détails de ce projet s'affiche.
6. Si vous souhaitez afficher les erreurs non définitives dans le jeu de données d'entraînement, choisissez l'onglet Entraînement. Sinon, choisissez l'onglet Test pour afficher les erreurs non définitives dans le jeu de données de test.
7. Dans la section Étiquettes de la galerie du jeu de données, sélectionnez Erreurs. La galerie du jeu de données est filtrée pour n'afficher que les images comportant des erreurs.

8. Choisissez Erreur sous une image pour voir le code d'erreur. Utilisez les informations fournies sous [Erreurs de validation de lignes JSON non terminales](#) pour corriger l'erreur.



Entraînement d'un modèle Amazon Rekognition Custom Labels

Vous pouvez entraîner un modèle à l'aide de la console Amazon Rekognition Custom Labels ou de l'API Amazon Rekognition Custom Labels. Si l'apprentissage du modèle échoue, utilisez les

informations fournies [Déboguer un modèle d'entraînement ayant échoué](#) pour trouver la cause de l'échec.

 Note

Vous êtes facturé en fonction du temps nécessaire à l'entraînement d'un modèle. La formation dure généralement de 30 minutes à 24 heures. Pour plus d'informations, consultez la section [Horaires de formation](#).

Une nouvelle version d'un modèle est créée chaque fois que le modèle est entraîné. Amazon Rekognition Custom Labels crée un nom pour le modèle qui est une combinaison du nom du projet et de l'horodatage de la création du modèle.

Pour entraîner votre modèle, Amazon Rekognition Custom Labels crée une copie de votre formation source et des images de test. Par défaut, les images copiées sont chiffrées au repos à l'aide d'une clé détenue et gérée par AWS. Vous pouvez également choisir d'utiliser votre propre AWS KMS key. Si vous utilisez votre propre clé KMS, vous devrez disposer des autorisations suivantes sur la clé KMS.

- km :CreateGrant
- km :DescribeKey

Pour plus d'informations, consultez les [concepts AWS Key Management Service](#). Vos images sources ne sont pas affectées.

Vous pouvez utiliser le chiffrement côté serveur KMS (SSE-KMS) pour chiffrer les images d'entraînement et de test dans votre compartiment Amazon S3, avant qu'elles ne soient copiées par Amazon Rekognition Custom Labels. Pour autoriser Amazon Rekognition Custom Labels à accéder à vos images, votre AWS compte doit disposer des autorisations suivantes sur la clé KMS.

- km :GenerateDataKey
- kms:Decrypt

Pour plus d'informations, consultez [la section Protection des données à l'aide du chiffrement côté serveur avec les clés KMS stockées dans AWS Key Management Service \(SSE-KMS\)](#).

Une fois que vous avez entraîné un modèle, vous pouvez évaluer sa performance et apporter des améliorations. Pour plus d'informations, veuillez consulter [Amélioration d'un modèle d'étiquettes personnalisées Amazon Rekognition certifié](#).

Pour d'autres tâches relatives aux modèles, telles que le balisage d'un modèle, consultez [Gestion d'un modèle Amazon Rekognition Custom Labels](#).

Rubriques

- [Entraînement d'un modèle \(console\)](#)
- [Entraînement d'un modèle \(SDK\)](#)

Entraînement d'un modèle (console)

Vous pouvez utiliser la console Amazon Rekognition Custom Labels pour entraîner un modèle.

La formation nécessite un projet avec un jeu de données d'entraînement et un jeu de données de test. Si votre projet ne comporte pas de jeu de données de test, la console Amazon Rekognition Custom Labels divise le jeu de données d'entraînement pendant l'entraînement pour en créer un pour votre projet. Les images choisies constituent un échantillon représentatif et ne sont pas utilisées dans l'ensemble de données d'apprentissage. Nous vous recommandons de diviser votre jeu de données d'entraînement uniquement si vous ne disposez pas d'un autre jeu de données de test que vous puissiez utiliser. La division d'un jeu de données d'entraînement réduit le nombre d'images disponibles pour l'entraînement.

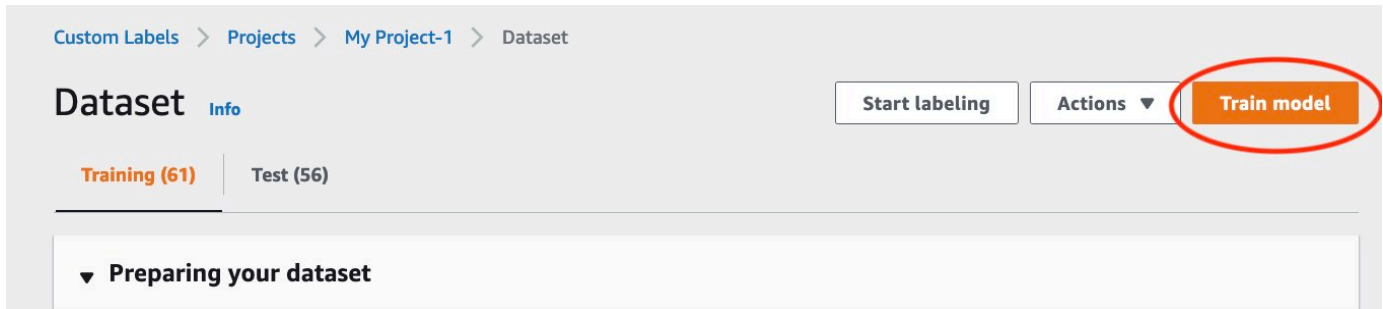
Note

Vous êtes facturé en fonction du temps nécessaire à l'entraînement d'un modèle. Pour plus d'informations, consultez la section [Horaires de formation](#).

Pour entraîner votre modèle (console)

1. Ouvrez la console Amazon Rekognition à l'[adresse https://console.aws.amazon.com/rekognition/](https://console.aws.amazon.com/rekognition/).
2. Choisissez Utiliser des étiquettes personnalisées.
3. Dans le panneau de navigation de gauche, choisissez Projects.
4. Sur la page Projets, choisissez le projet qui contient le modèle que vous souhaitez entraîner.

5. Sur la page Projet, sélectionnez Modèle de train.



6. (Facultatif) Si vous souhaitez utiliser votre propre clé de chiffrement AWS KMS, procédez comme suit :
 - a. Dans Chiffrement des données d'image, choisissez Personnaliser les paramètres de chiffrement (avancés).
 - b. Dans `encryption.aws_kms_key`, entrez le nom de ressource Amazon (ARN) de votre clé ou choisissez une clé AWS KMS existante. Pour créer une nouvelle clé, choisissez Créer une clé AWS IMS.
7. (Facultatif) Si vous souhaitez ajouter des balises à votre modèle, procédez comme suit :
 - a. Dans la section Tags, choisissez Ajouter un nouveau tag.
 - b. Entrez ce qui suit :
 - i. Le nom de la clé dans Key.
 - ii. La valeur de la clé dans Value.
 - c. Pour ajouter d'autres balises, répétez les étapes 6a et 6b.
 - d. (Facultatif) Si vous souhaitez supprimer une balise, choisissez Remove (Supprimer) en regard de la balise à supprimer. Si vous supprimez une étiquette précédemment enregistrée, elle est supprimée lorsque vous enregistrez vos modifications.
8. Sur la page Modèle de train, choisissez Modèle de train. Amazon Resource Name (Amazon Resource ARN) de votre projet doit se trouver dans la zone d'édition Choose project (Choisir un projet). Dans le cas contraire, saisissez ARN Amazon de votre projet.

Custom Labels > Train model

Train model

Training details [Info](#)

Choose project
Amazon Rekognition Custom Labels trains a new version of the model within the project you choose.

arn:aws:rekognition:us-east-

Tags [Info](#)

A tag is a label that you can assign to your model. Each tag consists of a key and an optional value.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

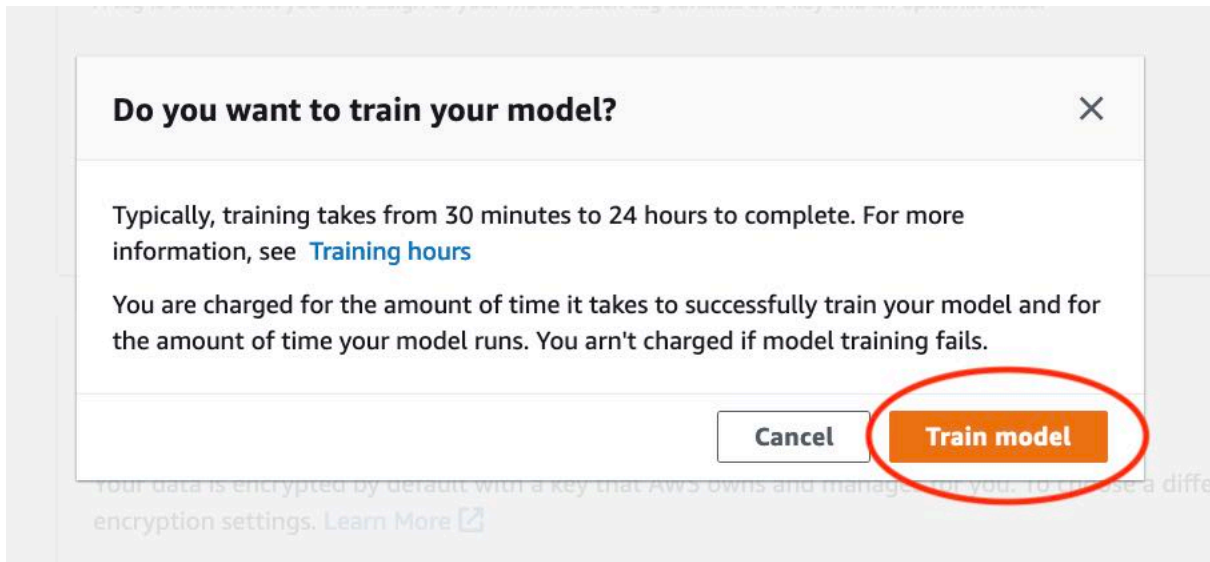
Image Data Encryption

Your data is encrypted by default with a key that AWS owns and manages for you. To choose a different key, customize your encryption settings. [Learn More](#)

Customize encryption settings (advanced)

Cancel **Train Model**

9. Dans la section Voulez-vous entraîner votre modèle ? dans la boîte de dialogue, choisissez Train model (Modèle de train).



10. Dans la section Modèles de la page du projet, vous pouvez vérifier l'état actuel dans la Model Status colonne où se déroule la formation. La formation d'un modèle prend un certain temps.

Custom Labels > Projects > My-Project-1

My-Project-1 Info

How it works

Creating your dataset

1. Create dataset
A dataset is a collection of images, and image labels, that you use to train or test a model.

Created

2. Label images
Labels identify objects, scenes, or concepts on an entire image, or they identify object locations on an image.

Label images

Training your model

3. Train model
Depending on the training dataset, the training model finds image-level scenes and concepts, or it finds object locations.

Train model

Evaluating your model

4. Check performance metrics
Performance metrics tell you if your model needs additional training before you can use it.

Check metrics

Project details

Project name	Created	Dataset	Models
My-Project-1	October 04, 2021 at 13:05:06 (UTC-07:00)	🔄	1

Models (1) Delete model Download validation results ▾

🔍 Find resources < 1 ... >

<input type="checkbox"/>	Name	Date created	Training dataset	Test dataset	Model performance (F1 score)	Model status	Status message
<input type="checkbox"/>	My-Project-1.2021-10-04T13.52.53	October 04, 2021			N/A	TRAINING_IN_PROGRESS	The model is being trained.

11. Une fois la formation terminée, choisissez le nom du modèle. L'entraînement est terminé lorsque le statut du modèle est `TRAINING_COMPLETED`. Si la formation échoue, lisez [Déboguer un modèle d'entraînement ayant échoué](#).

The screenshot shows the Amazon Rekognition console interface for a project named 'rooms_19'. At the top, there is a 'Delete project' button. Below it is a 'Create datasets' information box. The main section is titled 'Models (1)' and contains a table with the following data:

Name	Date created	Training dataset	Testing dataset	Model performance	Model status	Status message
rooms_19.2021-07-13T10.36.30	July 13, 2021	rooms_19_training_dataset	rooms_19_test_dataset	0.902	TRAINING_COMPLETED	The model is ready to run.

12. Étape suivante : Évaluer votre modèle. Pour plus d'informations, veuillez consulter [Amélioration d'un modèle d'étiquettes personnalisées Amazon Rekognition certifié](#).

Entraînement d'un modèle (SDK)

Vous entraînez un mannequin en l'appelant [CreateProjectVersion](#). Pour entraîner un modèle, les informations suivantes sont nécessaires :

- Name (Nom) — Nom unique de la version du modèle.
- Amazon Resource Name (Amazon Resource Name) du projet qui gère le modèle.
- Emplacement des résultats de formation : l'emplacement Amazon S3 dans lequel les résultats sont placés. Vous pouvez utiliser le même emplacement que le compartiment Amazon S3 de la console, ou vous pouvez en choisir un autre. Nous vous recommandons de choisir un autre emplacement, car cela vous permet de définir des autorisations et d'éviter d'éventuels conflits de noms avec les résultats de formation liés à l'utilisation de la console Amazon Rekognition Custom Labels.

La formation utilise les ensembles de données de formation et de test associés au projet. Pour plus d'informations, veuillez consulter [Gestion des jeux de données](#).

Note

Vous pouvez éventuellement spécifier des fichiers manifestes de jeux de données d'entraînement et de test externes à un projet. Si vous ouvrez la console après avoir entraîné un modèle à l'aide de fichiers manifestes externes, Amazon Rekognition Custom Labels crée les ensembles de données pour vous en utilisant le dernier ensemble de fichiers manifestes utilisé pour la formation. Vous ne pouvez plus entraîner une version de modèle pour le

projet en spécifiant des fichiers manifestes externes. Pour plus d'informations, consultez [CreateProjectVersion](#).

Le formulaire de réponse `CreateProjectVersion` est un ARN que vous utilisez pour identifier la version du modèle dans les demandes suivantes. Vous pouvez également utiliser l'ARN pour sécuriser la version du modèle. Pour plus d'informations, veuillez consulter [Sécurisation des projets d'étiquettes personnalisées Amazon Rekognition](#).

La formation d'une version modèle prend un certain temps. Les exemples Python et Java présentés dans cette rubrique utilisent des serveurs qui attendent la fin de la formation. Un serveur est une méthode utilitaire qui interroge la survenue d'un état particulier. Vous pouvez également obtenir l'état actuel de la formation en appelant `DescribeProjectVersions`. La formation est terminée lorsque la valeur du `Status` champ est `TRAINING_COMPLETED`. Une fois la formation terminée, vous pouvez évaluer la qualité du modèle en consultant les résultats de l'évaluation.

Entraînement d'un modèle (SDK)

L'exemple suivant montre comment entraîner un modèle à l'aide des ensembles de données d'entraînement et de test associés à un projet.

Pour entraîner un modèle (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS CLI clés AWS SDK. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez l'exemple de code suivant pour entraîner un projet.

AWS CLI

L'exemple suivant crée un modèle. L'ensemble de données d'entraînement est divisé pour créer l'ensemble de données de test. Remplacez les éléments suivants :

- `my_project_arn` avec l'Amazon Resource Name (Amazon Resource Name) du projet.
- `version_name` avec le nom de version unique de votre choix.
- `output_bucket` avec le nom du compartiment Amazon S3 dans lequel Amazon Rekognition Custom Labels enregistre les résultats de formation.
- `output_folder` avec le nom du dossier dans lequel les résultats de la formation sont enregistrés.

- (paramètre facultatif) `--kms-key-id` avec l'identifiant de votre clé principale client AWS Key Management Service.

```
aws rekognition create-project-version \  
  --project-arn project_arn \  
  --version-name version_name \  
  --output-config '{"S3Bucket": "output_bucket", "S3KeyPrefix": "output_folder"}' \  
  \  
  --profile custom-labels-access
```

Python

L'exemple suivant crée un modèle. Fournissez les arguments de ligne de commande suivants :

- `project_arn`— Amazon Resource Name (Amazon Resource Name) du projet.
- `version_name`— Un nom de version unique pour le modèle de votre choix.
- `output_bucket`— le nom du compartiment Amazon S3 dans lequel Amazon Rekognition Custom Labels enregistre les résultats de formation.
- `output_folder`— le nom du dossier dans lequel les résultats de la formation sont enregistrés.

Vous pouvez éventuellement fournir les paramètres de ligne de commande suivants pour associer une balise à votre modèle :

- `tag`— un nom de balise de votre choix que vous souhaitez joindre au modèle.
- `tag_value` la valeur de la balise.

```
#Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/  
amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-  
SAMPLECODE.)  
  
import argparse  
import logging  
import json
```

```
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def train_model(rek_client, project_arn, version_name, output_bucket,
               output_folder, tag_key, tag_key_value):
    """
    Trains an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to train a
    model.
    :param version_name: A version for the model.
    :param output_bucket: The S3 bucket that hosts training output.
    :param output_folder: The path for the training output within output_bucket
    :param tag_key: The name of a tag to attach to the model. Pass None to
    exclude
    :param tag_key_value: The value of the tag. Pass None to exclude

    """

    try:
        #Train the model

        status=""
        logger.info("training model version %s for project %s",
                    version_name, project_arn)

        output_config = json.loads(
            '{"S3Bucket": "'
            + output_bucket
            + '", "S3KeyPrefix": "'
            + output_folder
            + '" } '
        )

        tags={}

        if tag_key is not None and tag_key_value is not None:
            tags = json.loads(
                '{"' + tag_key + '":"' + tag_key_value + '"}'
            )
```

```
response=rek_client.create_project_version(
    ProjectArn=project_arn,
    VersionName=version_name,
    OutputConfig=output_config,
    Tags=tags
)

logger.info("Started training: %s", response['ProjectVersionArn'])

# Wait for the project version training to complete.

project_version_training_completed_waiter =
rek_client.get_waiter('project_version_training_completed')
project_version_training_completed_waiter.wait(ProjectArn=project_arn,
VersionNames=[version_name])

# Get the completion status.

describe_response=rek_client.describe_project_versions(ProjectArn=project_arn,
VersionNames=[version_name])
for model in describe_response['ProjectVersionDescriptions']:
    logger.info("Status: %s", model['Status'])
    logger.info("Message: %s", model['StatusMessage'])
    status=model['Status']

logger.info("finished training")

return response['ProjectVersionArn'], status

except ClientError as err:
    logger.exception("Couldn't create model: %s", err.response['Error']
['Message'] )
    raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """
```

```
parser.add_argument(
    "project_arn", help="The ARN of the project in which you want to train a
model"
)

parser.add_argument(
    "version_name", help="A version name of your choosing."
)

parser.add_argument(
    "output_bucket", help="The S3 bucket that receives the training
results."
)

parser.add_argument(
    "output_folder", help="The folder in the S3 bucket where training
results are stored."
)

parser.add_argument(
    "--tag_name", help="The name of a tag to attach to the model",
required=False
)

parser.add_argument(
    "--tag_value", help="The value for the tag.", required=False
)

def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Training model version {args.version_name} for project
{args.project_arn}")
```

```
# Train the model.
session = boto3.Session(profile_name='custom-labels-access')
rekognition_client = session.client("rekognition")

model_arn, status=train_model(rekognition_client,
                              args.project_arn,
                              args.version_name,
                              args.output_bucket,
                              args.output_folder,
                              args.tag_name,
                              args.tag_value)

print(f"Finished training model: {model_arn}")
print(f"Status: {status}")

except ClientError as err:
    logger.exception("Problem training model: %s", err)
    print(f"Problem training model: {err}")
except Exception as err:
    logger.exception("Problem training model: %s", err)
    print(f"Problem training model: {err}")

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

Java V2

L'exemple suivant entraîne un modèle. Fournissez les arguments de ligne de commande suivants :

- `project_arn`— Amazon Resource Name (Amazon Resource Name) du projet.
- `version_name`— Un nom de version unique pour le modèle de votre choix.
- `output_bucket`— le nom du compartiment Amazon S3 dans lequel Amazon Rekognition Custom Labels enregistre les résultats de formation.
- `output_folder`— le nom du dossier dans lequel les résultats de la formation sont enregistrés.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.CreateProjectVersionRequest;
import
    software.amazon.awssdk.services.rekognition.model.CreateProjectVersionResponse;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
    software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.waiters.RekognitionWaiter;

import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;

public class TrainModel {

    public static final Logger logger =
        Logger.getLogger(TrainModel.class.getName());

    public static String trainMyModel(RekognitionClient rekClient, String
        projectArn, String versionName,
        String outputBucket, String outputFolder) {

        try {

            OutputConfig outputConfig =
                OutputConfig.builder().s3Bucket(outputBucket).s3KeyPrefix(outputFolder).build();
```

```
        logger.log(Level.INFO, "Training Model for project {0}",
projectArn);
        CreateProjectVersionRequest createProjectVersionRequest =
CreateProjectVersionRequest.builder()

.projectArn(projectArn).versionName(versionName).outputConfig(outputConfig).build();

        CreateProjectVersionResponse response =
rekClient.createProjectVersion(createProjectVersionRequest);

        logger.log(Level.INFO, "Model ARN: {0}",
response.projectVersionArn());
        logger.log(Level.INFO, "Training model...");

        // wait until training completes

        DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                .versionNames(versionName)
                .projectArn(projectArn)
                .build();

        RekognitionWaiter waiter = rekClient.waiter();

        WaiterResponse<DescribeProjectVersionsResponse> waiterResponse =
waiter

.waitUntilProjectVersionTrainingCompleted(describeProjectVersionsRequest);

        Optional<DescribeProjectVersionsResponse> optionalResponse =
waiterResponse.matched().response();

        DescribeProjectVersionsResponse describeProjectVersionsResponse =
optionalResponse.get();

        for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                .projectVersionDescriptions()) {
            System.out.println("ARN: " +
projectVersionDescription.projectVersionArn());
            System.out.println("Status: " +
projectVersionDescription.statusAsString());
            System.out.println("Message: " +
projectVersionDescription.statusMessage());
```



```
    }

    return response.projectVersionArn();

} catch (RekognitionException e) {
    logger.log(Level.SEVERE, "Could not train model: {0}",
e.getMessage());
    throw e;
}

}

public static void main(String args[]) {

    String versionName = null;
    String projectArn = null;
    String projectVersionArn = null;
    String bucket = null;
    String location = null;

    final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>
<output_bucket> <output_folder>\n\n" + "Where:\n"
        + "    project_arn - The ARN of the project that you want to use.
\n\n"
        + "    version_name - A version name for the model.\n\n"
        + "    output_bucket - The S3 bucket in which to place the
training output. \n\n"
        + "    output_folder - The folder within the bucket that the
training output is stored in. \n\n";

    if (args.length != 4) {
        System.out.println(USAGE);
        System.exit(1);
    }

    projectArn = args[0];
    versionName = args[1];
    bucket = args[2];
    location = args[3];

    try {

        // Get the Rekognition client.
        RekognitionClient rekClient = RekognitionClient.builder()
```

```
        .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
        .region(Region.US_WEST_2)
        .build();

        // Train model
        projectVersionArn = trainMyModel(rekClient, projectArn, versionName,
bucket, location);

        System.out.println(String.format("Created model: %s for Project ARN:
%s", projectVersionArn, projectArn));

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        System.exit(1);
    }
}
}
```

3. Si la formation échoue, lisez [Déboguer un modèle d'entraînement ayant échoué](#).

Débuguer un modèle d'entraînement ayant échoué

Vous pouvez rencontrer des erreurs lors de l'apprentissage du modèle. Amazon Rekognition Custom Labels signale des erreurs d'entraînement dans la console et dans la réponse de

[DescribeProjectVersions](#)

Les erreurs sont soit terminales (l'entraînement ne peut pas continuer), soit non terminales (l'entraînement peut continuer). Pour les erreurs liées au contenu des ensembles de données d'entraînement et de test, vous pouvez télécharger les résultats de validation (un [résumé du manifeste et des manifestes de validation de l'entraînement et des tests](#)). Utilisez les codes d'erreur figurant dans les résultats de validation pour obtenir plus d'informations dans cette section. Cette section fournit également des informations sur les erreurs du fichier manifeste (erreurs du terminal qui se produisent avant que le contenu du fichier manifeste ne soit validé).

Note

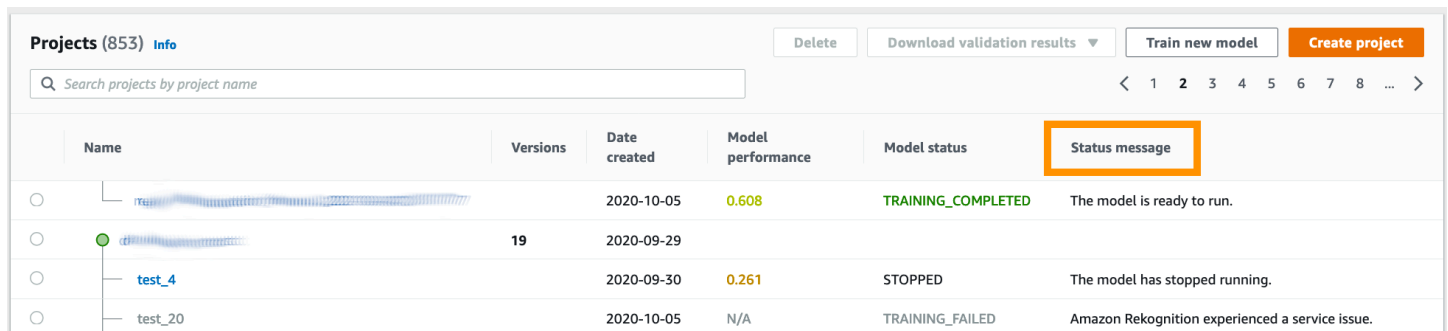
Un manifeste est le fichier utilisé pour stocker le contenu d'un ensemble de données.

Vous pouvez corriger certaines erreurs à l'aide de la console Amazon Rekognition Custom Labels. D'autres erreurs peuvent vous obliger à mettre à jour les fichiers manifestes de formation ou de test. Vous aurez peut-être besoin d'apporter d'autres modifications, telles que les autorisations IAM. Pour plus d'informations, consultez la documentation sur les erreurs individuelles.

Erreurs de terminal

Les erreurs du terminal interrompent l'apprentissage d'un modèle. Il existe trois catégories d'erreurs d'apprentissage du terminal : les erreurs de service, les erreurs de fichier manifeste et les erreurs de contenu du manifeste.

Dans la console, Amazon Rekognition Custom Labels affiche les erreurs de terminal associées à un modèle dans la colonne des messages d'état de la page des projets.



Name	Versions	Date created	Model performance	Model status	Status message
test_1		2020-10-05	0.608	TRAINING_COMPLETED	The model is ready to run.
test_2	19	2020-09-29			
test_4		2020-09-30	0.261	STOPPED	The model has stopped running.
test_20		2020-10-05	N/A	TRAINING_FAILED	Amazon Rekognition experienced a service issue.

Si vous utilisez le AWS SDK, vous pouvez savoir si une erreur du fichier manifeste du terminal ou une erreur de contenu du manifeste du terminal s'est produite en consultant la réponse de [DescribeProjectVersions](#). Dans ce cas, la `Status` valeur est `TRAINING_FAILED` et `StatusMessage` le champ contient l'erreur.

Erreurs de service

Les erreurs de service du terminal se produisent lorsqu'Amazon Rekognition rencontre un problème de service et ne peut pas poursuivre la formation. Par exemple, l'échec d'un autre service dont dépend Amazon Rekognition Custom Labels. Amazon Rekognition Custom Labels signale des erreurs de service dans la console car Amazon Rekognition a rencontré un problème de service. Si vous utilisez le AWS SDK, les erreurs de service qui se produisent pendant la formation

sont signalées comme une `InternalServerError` exception par [CreateProjectVersion](#) et [DescribeProjectVersions](#).

Si une erreur de service se produit, réessayez d'entraîner le modèle. Si la formation continue d'échouer, contactez [AWS Support](#) et incluez toutes les informations d'erreur signalées avec l'erreur de service.

Erreurs du fichier manifeste du terminal

Les erreurs de fichier manifeste sont des erreurs de terminal, présentes dans les ensembles de données d'entraînement et de test, qui se produisent au niveau du fichier ou dans plusieurs fichiers. Les erreurs du fichier manifeste sont détectées avant que le contenu des ensembles de données d'entraînement et de test ne soit validé. Les erreurs du fichier manifeste empêchent le signalement d'erreurs de [validation non liées au terminal](#). Par exemple, un fichier manifeste d'entraînement vide génère l'erreur « Le fichier manifeste est vide ». Le fichier étant vide, aucune erreur de validation de ligne JSON non terminale ne peut être signalée. Le résumé du manifeste n'est pas non plus créé.

Vous devez corriger les erreurs du fichier manifeste avant de pouvoir entraîner votre modèle.

La liste suivante répertorie les erreurs du fichier manifeste.

- [L'extension ou le contenu du fichier manifeste ne sont pas valides.](#)
- [Le fichier manifeste est vide.](#)
- [La taille du fichier manifeste dépasse la taille maximale prise en charge.](#)
- [Impossible d'écrire dans le compartiment S3 de sortie.](#)
- [Les autorisations du compartiment S3 sont incorrectes.](#)

Erreurs de contenu du manifeste du terminal

Les erreurs de contenu du manifeste sont des erreurs terminales liées au contenu d'un manifeste. Par exemple, si le message d'erreur « Le [fichier manifeste ne contient pas suffisamment d'images étiquetées par étiquette pour effectuer un découpage automatique](#) », l'entraînement ne peut pas se terminer car le jeu de données d'entraînement ne contient pas suffisamment d'images étiquetées pour créer un jeu de données de test.

En plus d'être signalée dans la console et dans la réponse de `DescribeProjectVersions`, l'erreur est signalée dans le résumé du manifeste avec toute autre erreur de contenu du manifeste du terminal. Pour plus d'informations, veuillez consulter [Présentation du récapitulatif du manifeste](#).

Les erreurs de ligne JSON non terminales sont également signalées dans des manifestes de résultats d'entraînement et de validation des tests distincts. Les erreurs de ligne JSON non terminales détectées par les étiquettes personnalisées Amazon Rekognition ne sont pas nécessairement liées aux erreurs de contenu du manifeste qui interrompent l'apprentissage. Pour plus d'informations, veuillez consulter [Comprendre les manifestes de résultats de validation des formations et des tests](#).

Vous devez corriger les erreurs de contenu du manifeste avant de pouvoir entraîner votre modèle.

Les messages d'erreur relatifs aux erreurs de contenu du manifeste sont les suivants.

- [Le fichier manifeste contient trop de lignes non valides.](#)
- [Le fichier manifeste contient des images provenant de plusieurs compartiments S3.](#)
- [Identifiant de propriétaire non valide pour le compartiment Images S3.](#)
- [Le fichier manifeste ne contient pas suffisamment d'images étiquetées par étiquette pour effectuer un découpage automatique.](#)
- [Le fichier manifeste comporte trop peu d'étiquettes.](#)
- [Le fichier manifeste comporte trop d'étiquettes.](#)
- [Moins de {} % de chevauchement d'étiquettes entre les fichiers manifestes d'entraînement et de test.](#)
- [Le fichier manifeste comporte trop peu d'étiquettes utilisables.](#)
- [Moins de {} % de chevauchement d'étiquettes utilisables entre les fichiers manifestes d'entraînement et de test.](#)
- [Impossible de copier les images depuis le compartiment S3.](#)


Erreurs de validation de lignes JSON non terminales

Les erreurs de validation des lignes JSON ne sont pas des erreurs terminales qui ne nécessitent pas l'ajout d'étiquettes personnalisées Amazon Rekognition pour arrêter la formation d'un modèle.

Les erreurs de validation de la ligne JSON ne sont pas affichées dans la console.

Dans les ensembles de données d'apprentissage et de test, une ligne JSON représente les informations d'apprentissage ou de test pour une seule image. Les erreurs de validation dans une ligne JSON, telles qu'une image non valide, sont signalées dans les manifestes de validation d'entraînement et de test. Amazon Rekognition Custom Labels termine la formation à l'aide des autres lignes JSON valides figurant dans le manifeste. Pour plus d'informations, veuillez consulter

[Comprendre les manifestes de résultats de validation des formations et des tests](#). Pour plus d'informations sur les règles de validation, consultez [Règles de validation des fichiers manifestes](#).

 Note

L'entraînement échoue s'il y a trop d'erreurs de ligne JSON.

Nous vous recommandons également de corriger les erreurs de ligne JSON non terminales, car elles peuvent entraîner de futures erreurs ou avoir un impact sur l'apprentissage de votre modèle.

Les étiquettes personnalisées Amazon Rekognition peuvent générer les erreurs de validation de ligne JSON non terminales suivantes.

- [La clé source-ref est manquante.](#)
- [Le format de la valeur de référence source n'est pas valide.](#)
- [Aucun attribut d'étiquette n'a été trouvé.](#)
- [Le format de l'attribut d'étiquette {} n'est pas valide.](#)
- [Le format des métadonnées de l'attribut d'étiquette n'est pas valide.](#)
- [Aucun attribut d'étiquette valide n'a été trouvé.](#)
- [Une ou plusieurs zones de délimitation présentent une valeur de confiance manquante.](#)
- [Un ou plusieurs identifiants de classe sont absents de la carte des classes.](#)
- [Le format de la ligne JSON n'est pas valide.](#)
- [L'image n'est pas valide. Vérifiez le chemin S3 et/ou les propriétés de l'image.](#)
- [Le cadre de sélection contient des valeurs hors cadre.](#)
- [La hauteur et la largeur du cadre de délimitation sont trop petites.](#)
- [Le nombre de cases de délimitation est supérieur au maximum autorisé.](#)
- [Aucune annotation valide n'a été trouvée.](#)

Présentation du récapitulatif du manifeste

Le récapitulatif du manifeste contient les informations suivantes.

- Informations sur les erreurs [Erreurs de contenu du manifeste du terminal](#) rencontrées lors de la validation.

- Informations sur l'emplacement des erreurs [Erreurs de validation de lignes JSON non terminales](#) dans les ensembles de données d'entraînement et de test.
- Statistiques d'erreur, telles que le nombre total de lignes JSON non valides détectées dans les ensembles de données d'entraînement et de test.

Le résumé du manifeste est créé pendant l'entraînement s'il n'y en a pas [Erreurs du fichier manifeste du terminal](#). Pour obtenir l'emplacement du fichier récapitulatif du manifeste (manifest_summary.json), consultez. [Obtenir les résultats de validation](#)

Note

[Les erreurs de service](#) et [les erreurs du fichier manifeste](#) ne sont pas signalées dans le résumé du manifeste. Pour plus d'informations, veuillez consulter [Erreurs de terminal](#).

Pour plus d'informations sur les erreurs spécifiques liées au contenu du manifeste, reportez-vous à la section [Erreurs de contenu du manifeste du terminal](#).

Format du fichier récapitulatif du manifeste

Un fichier manifeste comporte 2 sections, `statistics` et `errors`.

`statistics`

`statistics` contient des informations sur les erreurs dans les ensembles de données d'entraînement et de test.

- `training`— statistiques et erreurs détectées dans le jeu de données d'entraînement.
- `testing`— statistiques et erreurs détectées dans l'ensemble de données de test.

Les objets du `errors` tableau contiennent le code d'erreur et le message concernant les erreurs de contenu du manifeste.

Le `error_line_indices` tableau contient les numéros de ligne pour chaque ligne JSON du manifeste d'apprentissage ou de test qui comporte une erreur. Pour plus d'informations, veuillez consulter [Corriger les erreurs d'entraînement](#).

erreurs

Erreurs affectant à la fois l'ensemble de données d'entraînement et de test. Par exemple, cela [ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP](#) se produit lorsqu'il n'y a pas suffisamment d'étiquettes utilisables qui chevauchent les ensembles de données d'entraînement et de test.

```
{
  "statistics": {
    "training": {
      "use_case": String, # Possible values are IMAGE_LEVEL_LABELS,
OBJECT_LOCALIZATION and NOT_DETERMINED
      "total_json_lines": Number, # Total number json lines (images) in the
training manifest.
      "valid_json_lines": Number, # Total number of JSON Lines (images)
that can be used for training.
      "invalid_json_lines": Number, # Total number of invalid JSON Lines.
They are not used for training.
      "ignored_json_lines": Number, # JSON Lines that have a valid schema but
have no annotations. The aren't used for training and aren't counted as invalid.
      "error_json_line_indices": List[int], # Contains a list of line numbers
for JSON line errors in the training dataset.
      "errors": [
        {
          "code": String, # Error code for a training manifest content
error.
          "message": String # Description for a training manifest content
error.
        }
      ]
    },
    "testing": {
      "use_case": String, # Possible values are IMAGE_LEVEL_LABELS,
OBJECT_LOCALIZATION and NOT_DETERMINED
      "total_json_lines": Number, # Total number json lines (images) in the
manifest.
      "valid_json_lines": Number, # Total number of JSON Lines (images) that
can be used for testing.
      "invalid_json_lines": Number, # Total number of invalid JSON Lines.
They are not used for testing.
      "ignored_json_lines": Number, # JSON Lines that have a valid schema but
have no annotations. They aren't used for testing and aren't counted as invalid.
    }
  }
}
```



```

        "error_json_line_indices": List[int], # contains a list of error record
line numbers in testing dataset.
        "errors": [
            {
                "code": String,  # # Error code for a testing manifest content
error.
                "message": String # Description for a testing manifest content
error.
            }
        ]
    },
    "errors": [
        {
            "code": String, # # Error code for errors that span the training and
testing datasets.
            "message": String # Description of the error.
        }
    ]
}

```

Exemple de récapitulatif de manifeste

L'exemple suivant est un résumé partiel du manifeste qui montre une erreur de contenu du manifeste du terminal ([ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST](#)). Le `error_json_line_indices` tableau contient les numéros de ligne des erreurs de ligne JSON non terminales dans le manifeste de validation d'apprentissage ou de test correspondant.

```

{
  "errors": [],
  "statistics": {
    "training": {
      "use_case": "NOT_DETERMINED",
      "total_json_lines": 301,
      "valid_json_lines": 146,
      "invalid_json_lines": 155,
      "ignored_json_lines": 0,
      "errors": [
        {
          "code": "ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST",
          "message": "The manifest file contains too many invalid rows."
        }
      ]
    },


```

```
        "error_json_line_indices": [
            15,
            16,
            17,
            22,
            23,
            24,
            .
            .
            .
            .
            300
        ]
    },
    "testing": {
        "use_case": "NOT_DETERMINED",
        "total_json_lines": 15,
        "valid_json_lines": 13,
        "invalid_json_lines": 2,
        "ignored_json_lines": 0,
        "errors": [],
        "error_json_line_indices": [
            13,
            15
        ]
    }
}
```

Comprendre les manifestes de résultats de validation des formations et des tests

Au cours de la formation, Amazon Rekognition Custom Labels crée des manifestes de résultats de validation pour contenir les erreurs de ligne JSON non terminales. Les manifestes des résultats de validation sont des copies des ensembles de données d'entraînement et de test auxquels des informations d'erreur ont été ajoutées. Vous pouvez accéder aux manifestes de validation une fois la formation terminée. Pour plus d'informations, veuillez consulter [Obtenir les résultats de validation](#). Amazon Rekognition Custom Labels crée également un résumé du manifeste qui inclut des informations générales sur les erreurs de ligne JSON, telles que l'emplacement des erreurs

et le nombre d'erreurs de ligne JSON. Pour plus d'informations, veuillez consulter [Présentation du récapitulatif du manifeste](#).

 Note

Les résultats de validation (manifestes des résultats de validation de la formation et des tests et résumé du manifeste) ne sont créés que s'il n'y en a pas [Erreurs du fichier manifeste du terminal](#).

Un manifeste contient des lignes JSON pour chaque image du jeu de données. Dans les manifestes des résultats de validation, les informations d'erreur des lignes JSON sont ajoutées aux lignes JSON où des erreurs se produisent.

Une erreur de ligne JSON est une erreur non terminale liée à une seule image. Une erreur de validation non terminale peut invalider la totalité ou une partie de la ligne JSON. Par exemple, si l'image référencée dans une ligne JSON n'est pas au format PNG ou JPG, une [ERREUR_INVALID_IMAGE](#) erreur se produit et l'intégralité de la ligne JSON est exclue de l'apprentissage. L'entraînement se poursuit avec d'autres lignes JSON valides.

Dans une ligne JSON, une erreur peut signifier que la ligne JSON peut toujours être utilisée pour l'entraînement. Par exemple, si la valeur gauche de l'un des quatre cadres de délimitation associés à une étiquette est négative, le modèle est toujours entraîné à l'aide des autres cadres de sélection valides. Des informations d'erreur de ligne JSON sont renvoyées pour le cadre de délimitation non valide ([BOÎTE DE DÉLIMITATION D'ERREUR NON VALIDE](#)). Dans cet exemple, les informations d'erreur sont ajoutées à l'annotation objet dans lequel l'erreur se produit.

Les erreurs d'avertissement, par exemple [AVERTISSEMENT_NO_ANNOTATIONS](#), ne sont pas utilisées pour l'entraînement et sont considérées comme des lignes JSON ignorées (`ignored_json_lines`) dans le résumé du manifeste. Pour plus d'informations, veuillez consulter [Présentation du récapitulatif du manifeste](#). De plus, les lignes JSON ignorées ne sont pas prises en compte dans le calcul du seuil d'erreur de 20 % pour l'entraînement et les tests.

Pour plus d'informations sur des erreurs de validation de données non terminales spécifiques, consultez [Erreurs de validation de lignes JSON non terminales](#).

Note

S'il y a trop d'erreurs de validation des données, l'entraînement est arrêté et une erreur de [ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST](#) terminal est signalée dans le résumé du manifeste.

Pour plus d'informations sur la correction des erreurs de ligne JSON, consultez [Corriger les erreurs d'entraînement](#).

Format d'erreur de ligne JSON

Les étiquettes personnalisées Amazon Rekognition ajoutent des informations d'erreur de validation non terminales aux lignes JSON au niveau de l'image et au format de localisation des objets. Pour plus d'informations, veuillez consulter [the section called "Création d'un fichier manifeste"](#).

Erreurs au niveau de l'image

L'exemple suivant montre les `Error` tableaux d'une ligne JSON au niveau de l'image. Il existe deux séries d'erreurs. Erreurs liées aux métadonnées des attributs d'étiquette (dans cet exemple, les métadonnées sportives) et erreurs liées à l'image. Une erreur inclut un code d'erreur (code), un message d'erreur (message). Pour plus d'informations, veuillez consulter [Étiquettes au niveau de l'image dans les fichiers manifestes](#).

```
{
  "source-ref": String,
  "sport": Number,
  "sport-metadata": {
    "class-name": String,
    "confidence": Float,
    "type": String,
    "job-name": String,
    "human-annotated": String,
    "creation-date": String,
    "errors": [
      {
        "code": String, # error codes for label
        "message": String # Description and additional contextual details of
the error
      }
    ]
  }
}
```

```
    },
    "errors": [
      {
        "code": String, # error codes for image
        "message": String # Description and additional contextual details of the
error
      }
    ]
  }
}
```

Erreurs de localisation d'objets

L'exemple suivant montre les tableaux d'erreurs dans une ligne JSON de localisation d'objets. La ligne JSON contient un `Errors` tableau d'informations pour les champs des sections suivantes de la ligne JSON. Chaque `Error` objet inclut le code d'erreur et le message d'erreur.

- `attribut d'étiquette` : erreurs concernant les champs d'attribut d'étiquette. Voir `bounding-box` l'exemple.
- `annotations` : les erreurs d'annotation (cadres de délimitation) sont stockées dans le `annotations` tableau à l'intérieur de l'attribut `label`.
- `label attribute-metadata` — Erreurs concernant les métadonnées de l'attribut d'étiquette. Voir `bounding-box-metadata` l'exemple.
- `image` : erreurs non liées aux champs de métadonnées de l'attribut d'étiquette, de l'annotation et de l'attribut d'étiquette.

Pour plus d'informations, veuillez consulter [Localisation d'objets dans les fichiers manifestes](#).

```
{
  "source-ref": String,
  "bounding-box": {
    "image_size": [
      {
        "width": Int,
        "height": Int,
        "depth": Int,
      }
    ],
    "annotations": [
      {
        "class_id": Int,
```

```

        "left": Int,
        "top": Int,
        "width": Int,
        "height": Int,
        "errors": [ # annotation field errors
            {
                "code": String, # annotation field error code
                "message": String # Description and additional contextual
details of the error
            }
        ]
    },
    "errors": [ #label attribute field errors
        {
            "code": String, # error code
            "message": String # Description and additional contextual details of
the error
        }
    ],
    "bounding-box-metadata": {
        "objects": [
            {
                "confidence": Float
            }
        ],
        "class-map": {
            String: String
        },
        "type": String,
        "human-annotated": String,
        "creation-date": String,
        "job-name": String,
        "errors": [ #metadata field errors
            {
                "code": String, # error code
                "message": String # Description and additional contextual details of
the error
            }
        ]
    },
    "errors": [ # image errors
        {

```

```
    "code": String, # error code
    "message": String # Description and additional contextual details of the
error
  }
]
}
```

Exemple d'erreur de ligne JSON

La ligne JSON de localisation d'objets suivante (formatée pour des raisons de lisibilité) affiche une [ERROR_BOUNDING_BOX_TOO_SMALL](#) erreur. Dans cet exemple, les dimensions du cadre de délimitation (hauteur et largeur) ne sont pas supérieures à 1 x 1.

```
{
  "source-ref": "s3://bucket/Manifests/images/199940-1791.jpg",
  "bounding-box": {
    "image_size": [
      {
        "width": 3000,
        "height": 3000,
        "depth": 3
      }
    ],
    "annotations": [
      {
        "class_id": 1,
        "top": 0,
        "left": 0,
        "width": 1,
        "height": 1,
        "errors": [
          {
            "code": "ERROR_BOUNDING_BOX_TOO_SMALL",
            "message": "The height and width of the bounding box is too
small."
          }
        ]
      }
    ],
    {
      "class_id": 0,
      "top": 65,
      "left": 86,
      "width": 220,
```

```
        "height": 334
      }
    ]
  },
  "bounding-box-metadata": {
    "objects": [
      {
        "confidence": 1
      },
      {
        "confidence": 1
      }
    ],
    "class-map": {
      "0": "Echo",
      "1": "Echo Dot"
    },
    "type": "groundtruth/object-detection",
    "human-annotated": "yes",
    "creation-date": "2019-11-20T02:57:28.288286",
    "job-name": "my job"
  }
}
```

Obtenir les résultats de validation

Les résultats de validation contiennent des informations d'erreur pour [Erreurs de contenu du manifeste du terminal](#) et [Erreurs de validation de lignes JSON non terminales](#). Il existe trois fichiers de résultats de validation.

- `training_manifest_with_validation.json` : copie du fichier manifeste du jeu de données d'entraînement avec les informations d'erreur JSON Line ajoutées.
- `testing_manifest_with_validation.json` — Une copie du fichier manifeste de l'ensemble de données de test avec les informations d'erreur de ligne JSON ajoutées.
- `manifest_summary.json` : résumé des erreurs de contenu du manifeste et des erreurs de ligne JSON détectées dans les ensembles de données d'entraînement et de test. Pour plus d'informations, veuillez consulter [Présentation du récapitulatif du manifeste](#).

Pour plus d'informations sur le contenu des manifestes de validation de la formation et des tests, reportez-vous à la section [Déboguer un modèle d'entraînement ayant échoué](#).

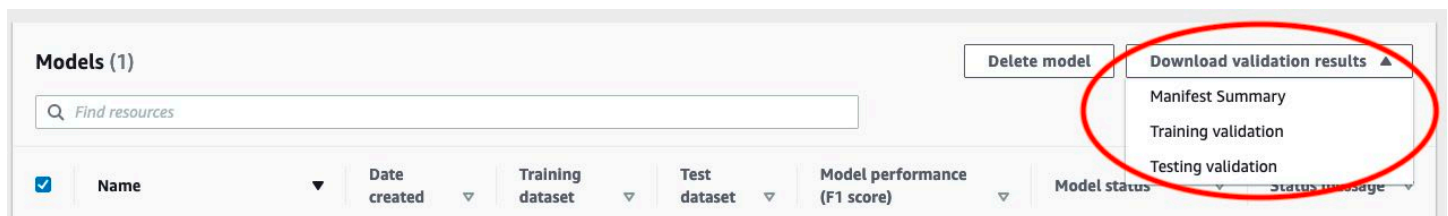
Note

- Les résultats de validation sont créés uniquement si aucun n'[Erreurs du fichier manifeste du terminal](#) est généré pendant l'entraînement.
- Si une [erreur de service](#) se produit après la validation du manifeste de formation et de test, les résultats de validation sont créés, mais la réponse de [DescribeProjectVersions](#) n'inclut pas les emplacements des fichiers de résultats de validation.

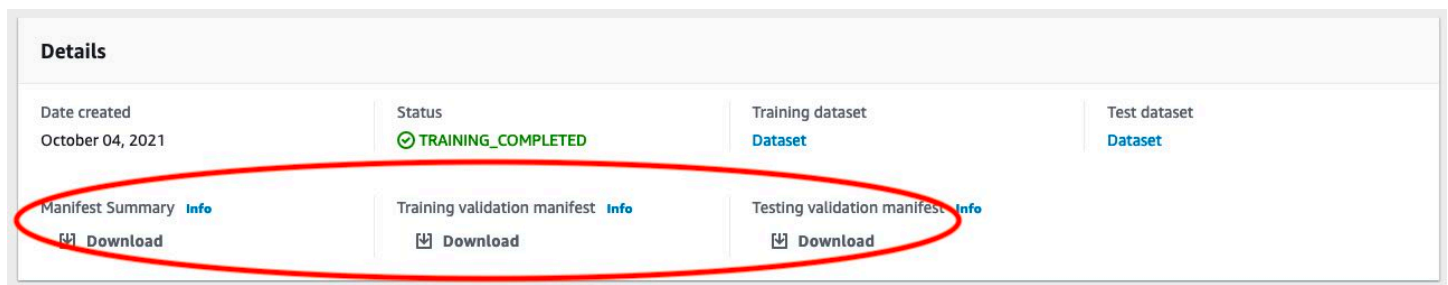
Une fois la formation terminée ou échouée, vous pouvez télécharger les résultats de validation à l'aide de la console Amazon Rekognition Custom Labels ou obtenir l'emplacement du bucket Amazon S3 en appelant l'API. [DescribeProjectVersions](#)

Obtenir les résultats de validation (console)

Si vous utilisez la console pour entraîner votre modèle, vous pouvez télécharger les résultats de validation à partir de la liste des modèles d'un projet, comme illustré dans le schéma suivant.



Vous pouvez également accéder au téléchargement des résultats de validation depuis la page de détails d'un modèle.



Pour plus d'informations, veuillez consulter [Entraînement d'un modèle \(console\)](#).

Obtenir les résultats de validation (SDK)

Une fois la formation du modèle terminée, Amazon Rekognition Custom Labels stocke les résultats de validation dans le compartiment Amazon S3 spécifié lors de la formation. Vous pouvez obtenir

l'emplacement du bucket S3 en appelant l'[DescribeProjectVersions](#) API, une fois la formation terminée. Pour entraîner un modèle, reportez-vous à la section [Entraînement d'un modèle \(SDK\)](#).

Un [ValidationData](#) objet est renvoyé pour l'ensemble de données d'entraînement ([TrainingDataResult](#)) et l'ensemble de données de test ([TestingDataResult](#)). Le résumé du manifeste est renvoyé dans `ManifestSummary`.

Une fois que vous avez obtenu l'emplacement du compartiment Amazon S3, vous pouvez télécharger les résultats de validation. Pour de plus amples informations, [veuillez consulter Comment télécharger un objet à partir d'un compartiment S3 ?](#) . Vous pouvez également utiliser l'[GetObject](#) opération.

Pour obtenir des données de validation (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS CLI AWS SDK. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez l'exemple suivant pour connaître l'emplacement des résultats de validation.

Python

`project_arn` Remplacez-le par l'Amazon Resource Name (ARN) du projet qui contient le modèle. Pour plus d'informations, veuillez consulter [Gestion d'un projet Amazon Rekognition Custom Labels](#). Remplacez `version_name` par le nom de la version du modèle. Pour plus d'informations, veuillez consulter [Entraînement d'un modèle \(SDK\)](#).

```
import boto3
import io
from io import BytesIO
import sys
import json

def describe_model(project_arn, version_name):

    client=boto3.client('rekognition')

    response=client.describe_project_versions(ProjectArn=project_arn,
        VersionNames=[version_name])

    for model in response['ProjectVersionDescriptions']:
        print(json.dumps(model,indent=4,default=str))
```

```
def main():  
  
    project_arn='project_arn'  
    version_name='version_name'  
  
    describe_model(project_arn, version_name)  
  
if __name__ == "__main__":  
    main()
```

3. Dans la sortie du programme, notez le `Validation` champ dans les `TrainingDataResult` objets `TestingDataResult` et. Le résumé du manifeste est disponible `ManifestSummary`.

Corriger les erreurs d'entraînement

Vous utilisez le résumé du manifeste pour identifier [Erreurs de contenu du manifeste du terminal](#) les personnes [Erreurs de validation de lignes JSON non terminales](#) rencontrées pendant l'entraînement. Vous devez corriger les erreurs de contenu du manifeste. Nous recommandons de corriger également les erreurs de ligne JSON non terminale. Pour plus d'informations sur des erreurs spécifiques, consultez [Erreurs de validation de lignes JSON non terminales](#) et [Erreurs de contenu du manifeste du terminal](#).

Vous pouvez apporter des corrections au jeu de données d'entraînement ou de test utilisé pour l'entraînement. Vous pouvez également apporter les corrections nécessaires dans les fichiers manifestes de validation d'entraînement et de test et les utiliser pour entraîner le modèle.

Une fois les corrections apportées, vous devez importer le ou les manifestes mis à jour et réentraîner le modèle. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).

La procédure suivante vous montre comment utiliser le résumé de manifeste pour corriger les erreurs de contenu de terminal. La procédure explique également comment localiser et corriger les erreurs de ligne JSON dans les manifestes de validation d'entraînement et de test.

Pour corriger les erreurs d'entraînement sur les étiquettes personnalisées Amazon Rekognition

1. Téléchargez les fichiers de résultats de validation. Les noms des fichiers sont `training_manifest_with_validation.json`, `testing_manifest_with_validation.json` et `manifest_summary.json`. Pour plus d'informations, veuillez consulter [Obtenir les résultats de validation](#).

2. Ouvrez le fichier récapitulatif du manifeste (manifest_summary.json).
3. Corrigez les erreurs éventuelles dans le résumé du manifeste. Pour plus d'informations, veuillez consulter [Présentation du récapitulatif du manifeste](#).
4. Dans le résumé du manifeste, parcourez le error_line_indices tableau training et corrigez les erreurs training_manifest_with_validation.json aux numéros de ligne JSON correspondants. Pour plus d'informations, veuillez consulter [the section called "Comprendre les manifestes de résultats de validation des formations et des tests"](#).
5. Parcourez le error_line_indices tableau testing et corrigez les erreurs aux numéros testing_manifest_with_validation.json de ligne JSON correspondants.
6. Réentraînez le modèle à l'aide des fichiers manifestes de validation en tant qu'ensembles de données d'entraînement et de test. Pour plus d'informations, veuillez consulter [the section called "Entraînement d'un modèle"](#).

Si vous utilisez le AWS SDK et que vous choisissez de corriger les erreurs dans les fichiers manifestes de données d'entraînement ou de validation des tests, utilisez l'emplacement des fichiers manifestes de données de validation dans les paramètres [TestingData](#) d'entrée [TrainingData](#) et pour [CreateProjectVersion](#). Pour plus d'informations, veuillez consulter [Entraînement d'un modèle \(SDK\)](#).

Priorité des erreurs de ligne JSON

Les erreurs de ligne JSON suivantes sont détectées en premier. Si l'une de ces erreurs se produit, la validation des erreurs de ligne JSON est arrêtée. Vous devez corriger ces erreurs avant de pouvoir corriger les autres erreurs de ligne JSON.

- RÉFÉRENCE SOURCE_MANQUANTE
- ERROR_SOURCE_REF_FORMAT NON VALIDE
- ERROR_NO_LABEL_ATTRIBUTES
- ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT
- ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT
- ERROR_MISSING_BOUNDING_BOX_CONFIDENCE
- ERROR_MISSING_CLASS_MAP_ID
- ERROR_INVALID_JSON_LINE

Erreurs du fichier manifeste du terminal

Cette rubrique décrit les [Erreurs du fichier manifeste du terminal](#). Aucun code d'erreur n'est associé aux erreurs du fichier manifeste. Les manifestes des résultats de validation ne sont pas créés lorsqu'une erreur survient dans le fichier manifeste du terminal. Pour plus d'informations, veuillez consulter [Présentation du récapitulatif du manifeste](#). Les erreurs du manifeste du terminal empêchent le signalement de [Erreurs de validation de lignes JSON non terminales](#).

L'extension ou le contenu du fichier manifeste ne sont pas valides.

Le fichier manifeste de formation ou de test ne possède pas d'extension de fichier ou son contenu n'est pas valide.

Pour corriger l'erreur L'extension ou le contenu du fichier manifeste ne sont pas valides.

- Vérifiez les causes possibles suivantes dans les fichiers manifestes d'entraînement et de test.
 - Le fichier manifeste ne possède pas d'extension de fichier. Par convention, l'extension du fichier est `.manifest`.
 - Impossible de trouver le compartiment ou la clé Amazon S3 pour le fichier manifeste.

Le fichier manifeste est vide.

Le fichier manifeste d'entraînement ou de test utilisé pour l'entraînement existe, mais il est vide. Le fichier manifeste nécessite une ligne JSON pour chaque image que vous utilisez pour la formation et les tests.

Pour corriger l'erreur Le fichier manifeste est vide.

1. Vérifiez lesquels des manifestes de formation ou de test sont vides.
2. Ajoutez des lignes JSON au fichier manifeste vide. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#). Vous pouvez également créer un nouvel ensemble de données à l'aide de la console. Pour plus d'informations, veuillez consulter [the section called "Création de jeux de données avec des images"](#).

La taille du fichier manifeste dépasse la taille maximale prise en charge.

La taille du fichier manifeste d'entraînement ou de test (en octets) est trop importante. Pour plus d'informations, veuillez consulter [Amazon Rekognition](#). Un fichier manifeste peut contenir moins que le nombre maximum de lignes JSON tout en dépassant la taille de fichier maximale.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger l'erreur La taille du fichier manifeste dépasse la taille maximale prise en charge.

Pour corriger l'erreur La taille du fichier manifeste dépasse la taille maximale prise en charge.

1. Vérifiez quels manifestes d'entraînement et de test dépassent la taille de fichier maximale.
2. Réduisez le nombre de lignes JSON trop volumineuses dans les fichiers manifestes. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).

Les autorisations du compartiment S3 sont incorrectes.

Amazon Rekognition Custom Labels n'est pas autorisé à accéder à un ou plusieurs compartiments contenant les fichiers manifestes de formation et de test.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

Pour corriger l'erreur Les autorisations du compartiment S3 sont incorrectes.

- Vérifiez les autorisations pour le ou les compartiments contenant les manifestes de formation et de test. Pour plus d'informations, veuillez consulter [Étape 2 : Configurer les autorisations d'accès à la console Étiquettes personnalisées Amazon Rekognition](#).

Impossible d'écrire dans le compartiment S3 de sortie.

Le service n'est pas en mesure de générer les fichiers de sortie de formation.

Pour corriger l'erreur Impossible d'écrire dans le compartiment S3 de sortie.

- Vérifiez que les informations relatives au compartiment Amazon S3 figurant dans le paramètre [OutputConfig](#)d'entrée to [CreateProjectVersion](#)sont correctes.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

Erreurs de contenu du manifeste du terminal

Cette rubrique décrit ce qui est [Erreurs de contenu du manifeste du terminal](#) indiqué dans le résumé du manifeste. Le résumé du manifeste inclut un code d'erreur et un message pour chaque erreur détectée. Pour plus d'informations, veuillez consulter [Présentation du récapitulatif du manifeste](#). Les erreurs de contenu du manifeste du terminal n'empêchent pas le signalement de [Erreurs de validation de lignes JSON non terminales](#).

ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST

Error message (Message d'erreur)

Le fichier manifeste contient trop de lignes non valides.

En savoir plus

Une ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST erreur se produit si trop de lignes JSON contiennent du contenu non valide.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger une erreur.

ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST

Pour corriger ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST

1. Vérifiez que le manifeste ne contient pas d'erreurs de ligne JSON. Pour plus d'informations, veuillez consulter [Comprendre les manifestes de résultats de validation des formations et des tests](#).
2. Corriger les lignes JSON contenant des erreurs Pour plus d'informations, consultez [Erreurs de validation de lignes JSON non terminales](#).

IMAGES_ERROR_IN_MULTIPLE_S3_BUCKETS

Error message (Message d'erreur)

Le fichier manifeste contient des images provenant de plusieurs compartiments S3.

En savoir plus

Un manifeste ne peut faire référence qu'à des images stockées dans un seul bucket. Chaque ligne JSON stocke l'emplacement Amazon S3 d'un emplacement d'image dans la valeur `source-ref`. Dans l'exemple suivant, le nom du bucket est `my-bucket`.

```
"source-ref": "s3://my-bucket/images/sunrise.png"
```

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

Pour réparer **ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS**

- Assurez-vous que toutes vos images se trouvent dans le même compartiment Amazon S3 et que la valeur de chaque ligne JSON fait référence au compartiment `source-ref` dans lequel vos images sont stockées. Vous pouvez également choisir un compartiment Amazon S3 préféré et supprimer les lignes JSON qui `source-ref` ne font pas référence à votre compartiment préféré.

ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET

Error message (Message d'erreur)

Les autorisations pour le compartiment d'images S3 ne sont pas valides.

En savoir plus

Les autorisations sur le compartiment Amazon S3 qui contient les images ne sont pas correctes.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

Pour réparer **ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET**

- Vérifiez les autorisations du bucket contenant les images. La valeur du `source-ref` pour une image contient l'emplacement du bucket.

ERROR_INVALID_IMAGES_S3_BUCKET_OWNER

Error message (Message d'erreur)

Identifiant de propriétaire non valide pour le compartiment Images S3.

En savoir plus

Le propriétaire du bucket qui contient les images d'entraînement ou de test est différent du propriétaire du bucket qui contient le manifeste de formation ou de test. Pour connaître le propriétaire d'un compartiment, vous pouvez utiliser la commande :

```
aws s3api get-bucket-acl --bucket bucket name
```

Le OWNER ID doit correspondre aux compartiments qui stockent les images et les fichiers manifestes.

Pour corriger ERROR_INVALID_IMAGES_S3_BUCKET_OWNER

1. Choisissez le propriétaire souhaité pour les compartiments de formation, de test, de sortie et d'image. Le propriétaire doit être autorisé à utiliser les étiquettes personnalisées Amazon Rekognition.
2. Pour chaque compartiment qui n'est pas actuellement détenu par le propriétaire souhaité, créez un nouveau compartiment Amazon S3 appartenant au propriétaire préféré.
3. Copiez le contenu de l'ancien compartiment dans le nouveau compartiment. Pour plus d'informations, consultez [Comment puis-je copier des objets entre des compartiments Amazon S3 ?](#).

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT

Error message (Message d'erreur)

Le fichier manifeste ne contient pas suffisamment d'images étiquetées par étiquette pour effectuer un découpage automatique.

En savoir plus

Pendant l'entraînement du modèle, vous pouvez créer un jeu de données de test en utilisant 20 % des images du jeu de données d'entraînement.

ERROR_INENSUFFUIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT se produit lorsqu'il n'y a pas assez d'images pour créer un jeu de données de test acceptable.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

Pour corriger ERROR_INENSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT

- Ajoutez d'autres images étiquetées à votre jeu de données d'entraînement. Vous pouvez ajouter des images dans la console Amazon Rekognition Custom Labels en ajoutant des images au jeu de données d'entraînement ou en ajoutant des lignes JSON à votre manifeste d'entraînement. Pour plus d'informations, veuillez consulter [Gestion des jeux de données](#).

ERROR_MANIFEST_TOO_FEW_LABELS

Error message (Message d'erreur)

Le fichier manifeste comporte trop peu d'étiquettes.

En savoir plus

Les ensembles de données de formation et de test comportent un nombre minimum d'étiquettes requis. Le minimum varie selon que le jeu de données entraîne ou teste un modèle pour détecter les étiquettes au niveau de l'image (classification) ou si le modèle détecte la position des objets. Si le jeu de données d'entraînement est divisé pour créer un jeu de données de test, le nombre d'étiquettes du jeu de données est déterminé après la division du jeu de données d'entraînement. Pour plus d'informations, veuillez consulter [Amazon Rekognition](#).

Pour corriger ERROR_MANIFEST_TOO_FEW_LABELS (console)

1. Ajoutez d'autres nouvelles étiquettes à l'ensemble de données. Pour plus d'informations, veuillez consulter [Gestion des étiquettes](#).
2. Ajoutez les nouvelles étiquettes aux images du jeu de données. Si votre modèle détecte des étiquettes au niveau de l'image, reportez-vous à la section [Attribution d'étiquettes au niveau de l'image à une image](#). Si votre modèle détecte la position d'objets, reportez-vous à la section [the section called "Étiquetage des objets à l'aide de cadres de délimitation"](#).

Pour corriger ERROR_MANIFEST_TOO_FEW_LABELS (ligne JSON)

- Ajoutez des lignes JSON pour les nouvelles images portant de nouvelles étiquettes. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#). Si votre modèle détecte des

étiquettes au niveau de l'image, vous ajoutez de nouveaux noms d'étiquettes dans le `class-name` champ. Par exemple, l'étiquette de l'image suivante est Sunrise.

```
{
  "source-ref": "s3://bucket/images/sunrise.png",
  "testdataset-classification_Sunrise": 1,
  "testdataset-classification_Sunrise-metadata": {
    "confidence": 1,
    "job-name": "labeling-job/testdataset-classification_Sunrise",
    "class-name": "Sunrise",
    "human-annotated": "yes",
    "creation-date": "2018-10-18T22:18:13.527256",
    "type": "groundtruth/image-classification"
  }
}
```

Si votre modèle détecte l'emplacement d'objets, ajoutez de nouvelles étiquettes `class-map`, comme le montre l'exemple suivant.

```
{
  "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
  "bounding-box": {
    "image_size": [{
      "width": 640,
      "height": 480,
      "depth": 3
    }],
    "annotations": [{
      "class_id": 1,
      "top": 251,
      "left": 399,
      "width": 155,
      "height": 101
    }, {
      "class_id": 0,
      "top": 65,
      "left": 86,
      "width": 220,
      "height": 334
    }
  ],
  "bounding-box-metadata": {
```

```
"objects": [{
  "confidence": 1
}, {
  "confidence": 1
}],
"class-map": {
  "0": "Echo",
  "1": "Echo Dot"
},
"type": "groundtruth/object-detection",
"human-annotated": "yes",
"creation-date": "2018-10-18T22:18:13.527256",
"job-name": "my job"
}
```

Vous devez mapper la table de la carte des classes aux annotations du cadre de délimitation. Pour plus d'informations, veuillez consulter [Localisation d'objets dans les fichiers manifestes](#).

ERROR_MANIFEST_TOO_MANY_LABELS

Error message (Message d'erreur)

Le fichier manifeste comporte trop d'étiquettes.

En savoir plus

Le nombre d'étiquettes uniques dans le manifeste (ensemble de données) est supérieur à la limite autorisée. Si le jeu de données d'entraînement est scindé pour créer un jeu de données de test, le nombre d'étiquettes est déterminé après le fractionnement.

Pour corriger ERROR_MANIFEST_TOO_MANY_LABELS (console)

- Supprimez les étiquettes du jeu de données. Pour plus d'informations, veuillez consulter [Gestion des étiquettes](#). Les étiquettes sont automatiquement supprimées des images et des cadres de délimitation de votre jeu de données.

Pour corriger ERROR_MANIFEST_TOO_MANY_LABELS (ligne JSON)

- Manifestes avec lignes JSON au niveau de l'image : si l'image comporte une seule étiquette, supprimez les lignes JSON pour les images qui utilisent l'étiquette souhaitée. Si la ligne JSON contient plusieurs étiquettes, supprimez uniquement l'objet JSON correspondant à l'étiquette souhaitée. Pour plus d'informations, veuillez consulter [Ajout de plusieurs étiquettes au niveau de l'image à une image](#).

Manifestes avec lignes JSON de localisation de l'objet : supprimez le cadre de délimitation et les informations d'étiquette associées pour l'étiquette que vous souhaitez supprimer. Procédez ainsi pour chaque ligne JSON contenant l'étiquette souhaitée. Vous devez supprimer l'étiquette du `class-map` tableau et des objets correspondants dans le `annotations` tableau `objects` et. Pour plus d'informations, veuillez consulter [Localisation d'objets dans les fichiers manifestes](#).

ERROR_INSUFFICIENT_LABEL_OVERLAP

Error message (Message d'erreur)

Moins de {} % de chevauchement d'étiquettes entre les fichiers manifestes d'entraînement et de test.

En savoir plus

Il y a moins de 50 % de chevauchement entre les noms des étiquettes des ensembles de données de test et les noms des étiquettes des ensembles de données d'entraînement.

Pour corriger ERROR_INSUFFICIENT_LABEL_OVERLAP (console)

- Supprimez les étiquettes du jeu de données d'entraînement. Vous pouvez également ajouter des étiquettes plus courantes à votre ensemble de données de test. Pour plus d'informations, veuillez consulter [Gestion des étiquettes](#). Les étiquettes sont automatiquement supprimées des images et des cadres de délimitation de votre jeu de données.

Pour corriger ERROR_INSUFFICIENT_LABEL_OVERLAP en supprimant les étiquettes du jeu de données d'entraînement (ligne JSON)

- Manifestes avec lignes JSON au niveau de l'image : si l'image comporte une seule étiquette, supprimez la ligne JSON de l'image qui utilise l'étiquette souhaitée. Si la ligne JSON contient plusieurs étiquettes, supprimez uniquement l'objet JSON correspondant à l'étiquette souhaitée.

Pour plus d'informations, veuillez consulter [Ajout de plusieurs étiquettes au niveau de l'image à une image](#). Procédez ainsi pour chaque ligne JSON du manifeste qui contient l'étiquette que vous souhaitez supprimer.

Manifestes avec lignes JSON de localisation de l'objet : supprimez le cadre de délimitation et les informations d'étiquette associées pour l'étiquette que vous souhaitez supprimer. Procédez ainsi pour chaque ligne JSON contenant l'étiquette souhaitée. Vous devez supprimer l'étiquette du `class-map` tableau et des objets correspondants dans le `annotations` tableau `objects` et. Pour plus d'informations, veuillez consulter [Localisation d'objets dans les fichiers manifestes](#).

Pour corriger `ERROR_INSUFFICIENT_LABEL_OVERLAP` en ajoutant des étiquettes communes à l'ensemble de données de test (ligne JSON)

- Ajoutez des lignes JSON au jeu de données de test qui incluent des images étiquetées avec des étiquettes déjà présentes dans le jeu de données d'entraînement. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).

ERROR_MANIFEST_TOO_FEW_LABELS_UTILISABLES

Error message (Message d'erreur)

Le fichier manifeste comporte trop peu d'étiquettes utilisables.

En savoir plus

Un manifeste d'entraînement peut contenir des lignes JSON au format d'étiquette au niveau de l'image et au format de localisation de l'objet. Selon le type de lignes JSON trouvées dans le manifeste d'apprentissage, Amazon Rekognition Custom Labels choisit de créer un modèle qui détecte les étiquettes au niveau de l'image ou un modèle qui détecte l'emplacement des objets. Les étiquettes personnalisées Amazon Rekognition filtrent les enregistrements JSON valides pour les lignes JSON qui ne sont pas au format choisi. `ERROR_MANIFEST_TOO_FEW_USABLE_LABELS` se produit lorsque le nombre d'étiquettes dans le manifeste du type de modèle choisi est insuffisant pour entraîner le modèle.

Au moins une étiquette est requise pour entraîner un modèle à détecter les étiquettes au niveau de l'image. Un minimum de deux étiquettes est requis pour entraîner un modèle à localiser des objets.

Pour corriger ERROR_MANIFEST_TOO_FEW_USABLE_LABELS (console)

1. Vérifiez le `use_case` champ dans le résumé du manifeste.
2. Ajoutez des étiquettes supplémentaires au jeu de données d'entraînement pour le cas d'utilisation (niveau de l'image ou localisation de l'objet) qui correspond à la valeur `use_case`. Pour plus d'informations, veuillez consulter [Gestion des étiquettes](#). Les étiquettes sont automatiquement supprimées des images et des cadres de délimitation de votre jeu de données.

Pour corriger ERROR_MANIFEST_TOO_FEW_USABLE_LABELS (ligne JSON)

1. Vérifiez le `use_case` champ dans le résumé du manifeste.
2. Ajoutez des étiquettes supplémentaires au jeu de données d'entraînement pour le cas d'utilisation (niveau de l'image ou localisation de l'objet) qui correspond à la valeur `use_case`. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).

ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP

Error message (Message d'erreur)

Moins de {} % de chevauchement d'étiquettes utilisables entre les fichiers manifestes d'entraînement et de test.

En savoir plus

Un manifeste d'entraînement peut contenir des lignes JSON au format d'étiquette au niveau de l'image et au format de localisation de l'objet. En fonction des formats trouvés dans le manifeste de formation, Amazon Rekognition Custom Labels choisit de créer un modèle qui détecte les étiquettes au niveau de l'image ou un modèle qui détecte l'emplacement des objets. Amazon Rekognition Custom Labels n'utilise pas d'enregistrements JSON valides pour les lignes JSON qui ne sont pas dans le format de modèle choisi. L'erreur `ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP` se produit lorsqu'il y a moins de 50 % de chevauchement entre les étiquettes de test et d'entraînement utilisées.

Pour corriger ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP (console)

- Supprimez les étiquettes du jeu de données d'entraînement. Vous pouvez également ajouter des étiquettes plus courantes à votre ensemble de données de test. Pour plus d'informations, veuillez

consulter [Gestion des étiquettes](#). Les étiquettes sont automatiquement supprimées des images et des cadres de délimitation de votre jeu de données.

Pour corriger ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP en supprimant les étiquettes du jeu de données d'entraînement (ligne JSON)

- Ensembles de données utilisés pour détecter les étiquettes au niveau de l'image : si l'image comporte une seule étiquette, supprimez la ligne JSON de l'image qui utilise l'étiquette souhaitée. Si la ligne JSON contient plusieurs étiquettes, supprimez uniquement l'objet JSON correspondant à l'étiquette souhaitée. Pour plus d'informations, veuillez consulter [Ajout de plusieurs étiquettes au niveau de l'image à une image](#). Procédez ainsi pour chaque ligne JSON du manifeste qui contient l'étiquette que vous souhaitez supprimer.

Ensembles de données utilisés pour détecter l'emplacement des objets : supprimez le cadre de sélection et les informations d'étiquette associées pour l'étiquette que vous souhaitez supprimer. Procédez ainsi pour chaque ligne JSON contenant l'étiquette souhaitée. Vous devez supprimer l'étiquette du class-map tableau et des objets correspondants dans le annotations tableau objects et. Pour plus d'informations, veuillez consulter [Localisation d'objets dans les fichiers manifestes](#).

Pour corriger ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP en ajoutant des étiquettes communes à l'ensemble de données de test (ligne JSON)

- Ajoutez des lignes JSON au jeu de données de test qui incluent des images étiquetées avec des étiquettes déjà présentes dans le jeu de données d'entraînement. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).

ERROR_FAILED_IMAGES_S3_COPY

Error message (Message d'erreur)

Impossible de copier les images depuis le compartiment S3.

En savoir plus

Le service n'a pu copier aucune des images de votre jeu de données.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

Pour corriger **ERROR_FAILED_IMAGES_S3_COPY**

1. Vérifiez les autorisations de vos images.
2. Si vous en utilisez AWS KMS, vérifiez la politique relative aux compartiments. Pour plus d'informations, veuillez consulter [Déchiffrement de fichiers chiffrés avec AWS Key Management Service](#).

Le fichier manifeste contient trop d'erreurs de terminal.

Il y a trop de lignes JSON contenant des erreurs de contenu dans le terminal.

Pour réparer **ERROR_TOO_MANY_RECORDS_IN_ERROR**

- Réduisez le nombre de lignes JSON (images) contenant des erreurs de contenu dans le terminal. Pour plus d'informations, veuillez consulter [Erreurs de contenu du manifeste du terminal](#).

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

Erreurs de validation de lignes JSON non terminales

Cette rubrique répertorie les erreurs de validation de lignes JSON non terminales signalées par Amazon Rekognition Custom Labels au cours de la formation. Les erreurs sont signalées dans le manifeste de validation de la formation et des tests. Pour plus d'informations, veuillez consulter [Comprendre les manifestes de résultats de validation des formations et des tests](#). Vous pouvez corriger une erreur de ligne JSON non terminale en mettant à jour la ligne JSON dans le fichier manifeste d'apprentissage ou de test. Vous pouvez également supprimer la ligne JSON du manifeste, mais cela risque de réduire la qualité de votre modèle. S'il existe de nombreuses erreurs de validation non liées au terminal, il vous sera peut-être plus facile de recréer le fichier manifeste. Les erreurs de validation se produisent généralement dans les fichiers manifestes créés manuellement. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#). Pour plus d'informations sur la correction des erreurs de validation, consultez [Corriger les erreurs d'entraînement](#). Certaines erreurs peuvent être corrigées à l'aide de la console Amazon Rekognition Custom Labels.

ERROR_MISSING_SOURCE_REF

Error message (Message d'erreur)

La clé source-ref est manquante.

En savoir plus

Le `source-ref` champ JSON Line fournit l'emplacement Amazon S3 d'une image. Cette erreur se produit lorsque la `source-ref` clé est manquante ou mal orthographiée. Cette erreur se produit généralement dans les fichiers manifestes créés manuellement. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).

Pour réparer **ERROR_MISSING_SOURCE_REF**

1. Vérifiez que la `source-ref` clé est présente et qu'elle est correctement orthographiée. Une `source-ref` clé et une valeur complètes sont similaires aux suivantes. `is"source-ref": "s3://bucket/path/image"`.
2. Mettez à jour la `source-ref` clé dans la ligne JSON. Vous pouvez également supprimer la ligne JSON du fichier manifeste.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

ERROR_SOURCE_REF_FORMAT_NON_VALIDÉ

Error message (Message d'erreur)

Le format de la valeur de référence source n'est pas valide.

En savoir plus

La `source-ref` clé est présente dans la ligne JSON, mais le schéma du chemin Amazon S3 est incorrect. Par exemple, le chemin est `https://...` au lieu de `S3://...`. Une erreur `ERROR_INVALID_SOURCE_REF_FORMAT` se produit généralement dans les fichiers manifestes créés manuellement. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).

Pour réparer **ERROR_INVALID_SOURCE_REF_FORMAT**

1. Vérifiez que le schéma est "source-ref": "s3://bucket/path/image". Par exemple, "source-ref": "s3://custom-labels-console-us-east-1-1111111111/images/000000242287.jpg".
2. Mettez à jour ou supprimez la ligne JSON dans le fichier manifeste.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour résoudre ce problème. **ERROR_INVALID_SOURCE_REF_FORMAT**

ERROR_NO_LABEL_ATTRIBUTES

Error message (Message d'erreur)

Aucun attribut d'étiquette n'a été trouvé.

En savoir plus

L'attribut d'étiquette ou le nom de -metadata clé de l'attribut d'étiquette (ou les deux) n'est pas valide ou est manquant. Dans l'exemple suivant, **ERROR_NO_LABEL_ATTRIBUTES** se produit chaque fois que la bounding-box-metadata touche bounding-box ou (ou les deux) est manquante. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).

```
{
  "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
  "bounding-box": {
    "image_size": [{
      "width": 640,
      "height": 480,
      "depth": 3
    }],
    "annotations": [{
      "class_id": 1,
      "top": 251,
      "left": 399,
      "width": 155,
      "height": 101
    }, {
      "class_id": 0,
      "top": 65,
      "left": 86,
```

```
    "width": 220,  
    "height": 334  
  ]]  
},  
"bounding-box-metadata": {  
  "objects": [{  
    "confidence": 1  
  }, {  
    "confidence": 1  
  }],  
  "class-map": {  
    "0": "Echo",  
    "1": "Echo Dot"  
  },  
  "type": "groundtruth/object-detection",  
  "human-annotated": "yes",  
  "creation-date": "2018-10-18T22:18:13.527256",  
  "job-name": "my job"  
}  
}
```

Une `ERROR_NO_LABEL_ATTRIBUTES` erreur se produit généralement dans un fichier manifeste créé manuellement. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).

Pour réparer **ERROR_NO_LABEL_ATTRIBUTES**

1. Vérifiez que l'identifiant d'attribut d'étiquette et les `-metadata` clés d'identifiant d'attribut d'étiquette sont présents et que les noms des clés sont correctement orthographiés.
2. Mettez à jour ou supprimez la ligne JSON dans le fichier manifeste.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger. `ERROR_NO_LABEL_ATTRIBUTES`

ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT

Error message (Message d'erreur)

Le format de l'attribut d'étiquette `{}` n'est pas valide.

En savoir plus

Le schéma de la clé d'attribut d'étiquette est manquant ou non valide. Une erreur `ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT` se produit généralement dans les fichiers manifestes créés manuellement. Pour plus d'informations, consultez [Création d'un fichier manifeste](#)

Pour réparer `ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT`

1. Vérifiez que la section JSON Line correspondant à la clé d'attribut du label est correcte. Dans l'exemple d'emplacement d'objet suivant, les annotations objets `image_size` et doivent être corrects. La clé d'attribut de l'étiquette est nommée `bounding-box`.

```
"bounding-box": {
  "image_size": [{
    "width": 640,
    "height": 480,
    "depth": 3
  }],
  "annotations": [{
    "class_id": 1,
    "top": 251,
    "left": 399,
    "width": 155,
    "height": 101
  }, {
    "class_id": 0,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
  }]
},
```

2. Mettez à jour ou supprimez la ligne JSON dans le fichier manifeste.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT

Error message (Message d'erreur)

Le format des métadonnées de l'attribut d'étiquette n'est pas valide.

En savoir plus

Le schéma de la clé de métadonnées de l'attribut d'étiquette est manquant ou non valide. Une erreur `ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT` se produit généralement dans les fichiers manifestes créés manuellement. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).

Pour réparer **ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT**

1. Vérifiez que le schéma de ligne JSON pour la clé de métadonnées de l'attribut d'étiquette est similaire à celui de l'exemple suivant. La clé de métadonnées de l'attribut d'étiquette est nommée `bounding-box-metadata`.

```
"bounding-box-metadata": {
  "objects": [{
    "confidence": 1
  }, {
    "confidence": 1
  }],
  "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
}
```

2. Mettez à jour ou supprimez la ligne JSON dans le fichier manifeste.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

ERROR_NO_VALID_LABEL_ATTRIBUTES

Error message (Message d'erreur)

Aucun attribut d'étiquette valide n'a été trouvé.

En savoir plus

Aucun attribut d'étiquette valide n'a été trouvé dans la ligne JSON. Amazon Rekognition Custom Labels vérifie à la fois l'attribut d'étiquette et l'identifiant de l'attribut d'étiquette. Une erreur `ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT` se produit généralement dans les fichiers manifestes créés manuellement. Pour plus d'informations, consultez [Création d'un fichier manifeste](#)

Si une ligne JSON n'est pas dans un format de SageMaker manifeste pris en charge, Amazon Rekognition Custom Labels marque la ligne JSON comme non valide et une `ERROR_NO_VALID_LABEL_ATTRIBUTES` erreur est signalée. À l'heure actuelle, Amazon Rekognition Custom Labels prend en charge les formats de tâches de classification et de cadre de sélection. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).

Pour réparer **ERROR_NO_VALID_LABEL_ATTRIBUTES**

1. Vérifiez que le JSON de la clé d'attribut d'étiquette et des métadonnées de l'attribut d'étiquette est correct.
2. Mettez à jour ou supprimez la ligne JSON dans le fichier manifeste. Pour plus d'informations, veuillez consulter [the section called "Création d'un fichier manifeste"](#).

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

ERROR_MISSING_BOUNDING_BOX_CONFIDENCE

Error message (Message d'erreur)

Une ou plusieurs zones de délimitation présentent une valeur de confiance manquante.

En savoir plus

La clé de confiance est manquante pour un ou plusieurs cadres de délimitation de l'emplacement des objets. La clé de confiance pour un cadre de sélection se trouve dans les métadonnées de l'étiquette, comme le montre l'exemple suivant. Une erreur `ERROR_MISSING_BOUNDING_BOX_CONFIDENCE` se produit généralement dans les fichiers

manifestes créés manuellement. Pour plus d'informations, veuillez consulter [the section called "Localisation d'objets dans les fichiers manifestes"](#).

```
"bounding-box-metadata": {
  "objects": [{
    "confidence": 1
  }, {
    "confidence": 1
  }],
```

Pour réparer **ERROR_MISSING_BOUNDING_BOX_CONFIDENCE**

1. Vérifiez que le `objects` tableau de l'attribut `label` contient le même nombre de clés de confiance que le tableau d'objets du `annotations` tableau d'attributs `label`.
2. Mettez à jour ou supprimez la ligne JSON dans le fichier manifeste.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

ERROR_MISSING_CLASS_MAP_ID

Error message (Message d'erreur)

Un ou plusieurs identifiants de classe sont absents de la carte des classes.

En savoir plus

L'objet `class_id` dans une annotation (cadre de délimitation) n'a pas d'entrée correspondante dans la classe de métadonnées de l'attribut d'étiquette `map()` `class-map`. Pour plus d'informations, veuillez consulter [Localisation d'objets dans les fichiers manifestes](#). Une erreur `ERROR_MISSING_CLASS_MAP_ID` se produit généralement dans les fichiers manifestes créés manuellement.

Pour corriger **ERROR_MISSING_CLASS_MAP_ID**

1. Vérifiez que la `class_id` valeur de chaque objet d'annotation (cadre de délimitation) correspond à une valeur dans le `class-map` tableau, comme indiqué dans l'exemple suivant. Le `annotations` tableau et le `class_map` tableau doivent comporter le même nombre d'éléments.

```
{
```



```
"source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
"bounding-box": {
  "image_size": [{
    "width": 640,
    "height": 480,
    "depth": 3
  }],
  "annotations": [{
    "class_id": 1,
    "top": 251,
    "left": 399,
    "width": 155,
    "height": 101
  }, {
    "class_id": 0,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
  }]
},
"bounding-box-metadata": {
  "objects": [{
    "confidence": 1
  }, {
    "confidence": 1
  }],
  "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
}
}
```

2. Mettez à jour ou supprimez la ligne JSON dans le fichier manifeste.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

ERROR_INVALID_JSON_LINE

Error message (Message d'erreur)

Le format de la ligne JSON n'est pas valide.

En savoir plus

Un caractère inattendu a été trouvé dans la ligne JSON. La ligne JSON est remplacée par une nouvelle ligne JSON qui contient uniquement les informations d'erreur. Une erreur `ERROR_INVALID_JSON_LINE` se produit généralement dans les fichiers manifestes créés manuellement. Pour plus d'informations, veuillez consulter [the section called “Localisation d'objets dans les fichiers manifestes”](#).

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

Pour réparer **ERROR_INVALID_JSON_LINE**

1. Ouvrez le fichier manifeste et accédez à la ligne JSON où se produit l'erreur `ERROR_INVALID_JSON_LINE`.
2. Vérifiez que la ligne JSON ne contient pas de caractères non valides et que des `,` caractères obligatoires `;` ou obligatoires ne sont pas manquants.
3. Mettez à jour ou supprimez la ligne JSON dans le fichier manifeste.

ERREUR_INVALID_IMAGE

Error message (Message d'erreur)

L'image n'est pas valide. Vérifiez le chemin S3 et/ou les propriétés de l'image.

En savoir plus

Le fichier référencé par `n'source-ref` n'est pas une image valide. Les causes potentielles incluent le rapport hauteur/largeur de l'image, la taille de l'image et le format de l'image.

Pour plus d'informations, veuillez consulter [Con.](#)

Pour réparer **ERROR_INVALID_IMAGE**

1. Vérifiez les éléments suivants :

- Le rapport hauteur/largeur de l'image est inférieur à 20:1.
 - La taille de l'image est supérieure à 15 Mo
 - L'image est au format PNG ou JPEG.
 - Le chemin d'accès à l'image `source-ref` est correct.
 - La dimension minimale de l'image est supérieure à 64 pixels x 64 pixels.
 - La dimension maximale de l'image est inférieure à 4 096 pixels x 4 096 pixels.
2. Mettez à jour ou supprimez la ligne JSON dans le fichier manifeste.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

DIMENSION DE L'IMAGE ERROR_NON VALIDE

Error message (Message d'erreur)

Les dimensions de l'image ne sont pas conformes aux dimensions autorisées.

En savoir plus

L'image référencée par `source-ref` n'est pas conforme aux dimensions d'image autorisées.

La dimension minimale est de 64 pixels. La dimension maximale est de 4 096 pixels.

`ERROR_INVALID_IMAGE_DIMENSION` est indiqué pour les images comportant des cadres de délimitation.

Pour plus d'informations, veuillez consulter [Con.](#)

Pour corriger **ERROR_INVALID_IMAGE_DIMENSION** (console)

1. Mettez à jour l'image dans le compartiment Amazon S3 avec les dimensions que les étiquettes personnalisées Amazon Rekognition peuvent traiter.
2. Dans la console Amazon Rekognition Custom Labels, procédez comme suit :
 - a. Supprimez les cadres de délimitation existants de l'image.
 - b. Ajoutez à nouveau les cadres de délimitation à l'image.
 - c. Enregistrez vos modifications.

Pour plus d'informations, veuillez consulter [Étiquetage des objets à l'aide de cadres de délimitation](#).

Pour corriger **ERROR_INVALID_IMAGE_DIMENSION** (SDK)

1. Mettez à jour l'image dans le compartiment Amazon S3 avec les dimensions que les étiquettes personnalisées Amazon Rekognition peuvent traiter.
2. Obtenez la ligne JSON existante pour l'image en appelant [ListDatasetEntries](#). Pour le paramètre `SourceRefContains` d'entrée, spécifiez l'emplacement Amazon S3 et le nom de fichier de l'image.
3. Appelez [UpdateDatasetEntries](#) et fournissez la ligne JSON pour l'image. Assurez-vous que la valeur de `source-ref` correspond à l'emplacement de l'image dans le compartiment Amazon S3. Mettez à jour les annotations du cadre de sélection pour qu'elles correspondent aux dimensions du cadre de sélection requises pour l'image mise à jour.

```
{
  "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
  "bounding-box": {
    "image_size": [{
      "width": 640,
      "height": 480,
      "depth": 3
    }],
    "annotations": [{
      "class_id": 1,
      "top": 251,
      "left": 399,
      "width": 155,
      "height": 101
    }, {
      "class_id": 0,
      "top": 65,
      "left": 86,
      "width": 220,
      "height": 334
    }
  ]
},
"bounding-box-metadata": {
  "objects": [{
```

```
"confidence": 1
}, {
  "confidence": 1
}],
"class-map": {
  "0": "Echo",
  "1": "Echo Dot"
},
"type": "groundtruth/object-detection",
"human-annotated": "yes",
"creation-date": "2013-11-18T02:53:27",
"job-name": "my job"
}
}
```

BOÎTE DE DÉLIMITATION D'ERREUR NON VALIDE

Error message (Message d'erreur)

Le cadre de sélection contient des valeurs hors cadre.

En savoir plus

Les informations du cadre de sélection indiquent une image qui se trouve en dehors du cadre de l'image ou qui contient des valeurs négatives.

Pour plus d'informations, veuillez consulter [Con.](#)

Pour réparer **ERROR_INVALID_BOUNDING_BOX**

1. Vérifiez les valeurs des cadres de délimitation du annotations tableau.

```
"bounding-box": {
  "image_size": [{
    "width": 640,
    "height": 480,
    "depth": 3
  }],
  "annotations": [{
    "class_id": 1,
    "top": 251,
```

```
"left": 399,  
"width": 155,  
"height": 101  
}]  
},
```

2. Mettez à jour ou supprimez la ligne JSON du fichier manifeste.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

ERREUR_NO_VALIDATION_ANNOTATIONS

Error message (Message d'erreur)

Aucune annotation valide n'a été trouvée.

En savoir plus

Aucun des objets d'annotation de la ligne JSON ne contient d'informations de cadre de délimitation valides.

Pour réparer **ERROR_NO_VALID_ANNOTATIONS**

1. Mettez à jour le annotations tableau pour inclure des objets de cadre de sélection valides. Vérifiez également que les informations du cadre de délimitation (`confidenceetc``class_map`) correspondant dans les métadonnées de l'attribut d'étiquette sont correctes. Pour plus d'informations, veuillez consulter [Localisation d'objets dans les fichiers manifestes](#).

```
{  
  "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",  
  "bounding-box": {  
    "image_size": [{  
      "width": 640,  
      "height": 480,  
      "depth": 3  
    }],  
    "annotations": [  
      {  
        "class_id": 1,    #annotation object  
        "top": 251,  
        "left": 399,
```

```
    "width": 155,
    "height": 101
  }, {
    "class_id": 0,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
  ]
},
"bounding-box-metadata": {
  "objects": [
    >{
      "confidence": 1          #confidence object
    },
    {
      "confidence": 1
    }
  ],
  "class-map": {
    "0": "Echo",    #label
    "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
}
}
```

2. Mettez à jour ou supprimez la ligne JSON du fichier manifeste.

Vous ne pouvez pas utiliser la console Amazon Rekognition Custom Labels pour corriger cette erreur.

ERROR_BOUNDING_BOX_TOO_SMALL

Error message (Message d'erreur)

La hauteur et la largeur du cadre de délimitation sont trop petites.

En savoir plus

Les dimensions du cadre de délimitation (hauteur et largeur) doivent être supérieures à 1 x 1 pixel.

Pendant la formation, Amazon Rekognition Custom Labels redimensionne une image si l'une de ses dimensions est supérieure à 1 280 pixels (les images source ne sont pas affectées). Les hauteurs et largeurs du cadre de délimitation qui en résultent doivent être supérieures à 1 x 1 pixel. L'emplacement d'un cadre de délimitation est stocké dans le annotations tableau d'une ligne JSON d'emplacement d'objet. Pour de plus amples informations, consultez [Localisation d'objets dans les fichiers manifestes](#).

```
"bounding-box": {
  "image_size": [{
    "width": 640,
    "height": 480,
    "depth": 3
  }],
  "annotations": [{
    "class_id": 1,
    "top": 251,
    "left": 399,
    "width": 155,
    "height": 101
  }]
},
```

Les informations d'erreur sont ajoutées à l'objet d'annotation.

Pour corriger ERROR_BOUNDING_BOX_TOO_SMALL

- Sélectionnez l'une des options suivantes :
 - Augmentez la taille des cadres de délimitation qui sont trop petits.
 - Supprimez les cadres de délimitation qui sont trop petits. Pour plus d'informations sur la suppression d'un cadre de sélection, reportez-vous à la section [ERREUR_TOO_MANY_BOUNDING_BOXES](#).
 - Supprimez l'image (ligne JSON) du manifeste.


```
  },
  {
    "confidence": 1
  }],
  "class-map": {
    "0": "Echo",    #label
    "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
}
}
```

ENREGISTREMENT D'AVERTISSEMENT NON ANNOTÉ

Message d'avertissement

L'enregistrement n'est pas annoté.

En savoir plus

Une image ajoutée à un ensemble de données à l'aide de la console Amazon Rekognition Custom Labels n'était pas étiquetée. La ligne JSON de l'image n'est pas utilisée pour l'entraînement.

```
{
  "source-ref": "s3://bucket/images/IMG_1186.png",
  "warnings": [
    {
      "code": "WARNING_UNANNOTATED_RECORD",
      "message": "Record is unannotated."
    }
  ]
}
```

Pour corriger WARNING_UNANNOTATED_RECORD

- Étiquetez l'image à l'aide de la console Amazon Rekognition Custom Labels. Pour des instructions, consultez [Attribution d'étiquettes au niveau de l'image à une image](#).

AVERTISSEMENT_NO_ANNOTATIONS

Message d'avertissement

Aucune annotation n'a été fournie.

En savoir plus

Une ligne JSON au format de localisation d'objets ne contient aucune information de cadre de délimitation, bien qu'elle soit annotée par un humain (`human-annotated = yes`). La ligne JSON est valide, mais elle n'est pas utilisée pour l'entraînement. Pour plus d'informations, veuillez consulter [Comprendre les manifestes de résultats de validation des formations et des tests](#).

```
{
  "source-ref": "s3://bucket/images/IMG_1186.png",
  "bounding-box": {
    "image_size": [
      {
        "width": 640,
        "height": 480,
        "depth": 3
      }
    ],
    "annotations": [
    ],
    "warnings": [
      {
        "code": "WARNING_NO_ATTRIBUTE_ANNOTATIONS",
        "message": "No attribute annotations were found."
      }
    ]
  },
  "bounding-box-metadata": {
    "objects": [
    ],
    "class-map": {
    },
    "type": "groundtruth/object-detection",
    "human-annotated": "yes",
  }
}
```

```
    "creation-date": "2013-11-18 02:53:27",
    "job-name": "my job"
  },
  "warnings": [
    {
      "code": "WARNING_NO_ANNOTATIONS",
      "message": "No annotations were found."
    }
  ]
}
```

Pour corriger WARNING_NO_ANNOTATIONS

- Sélectionnez l'une des options suivantes :
 - Ajoutez les informations du cadre de délimitation (annotations) à la ligne JSON. Pour plus d'informations, veuillez consulter [Localisation d'objets dans les fichiers manifestes](#).
 - Supprimez l'image (ligne JSON) du manifeste.

AVERTISSEMENT_NO_ATTRIBUTE_ANNOTATIONS

Message d'avertissement

Aucune annotation d'attribut n'est fournie.

En savoir plus

Une ligne JSON au format Object Localization ne contient aucune information d'annotation encadrée, bien qu'elle soit annotée par un humain (`human-annotated = yes`). Le tableau `annotations` n'est pas présent ou n'est pas renseigné. La ligne JSON est valide, mais elle n'est pas utilisée pour l'entraînement. Pour plus d'informations, veuillez consulter [Comprendre les manifestes de résultats de validation des formations et des tests](#).

```
{
  "source-ref": "s3://bucket/images/IMG_1186.png",
  "bounding-box": {
    "image_size": [
      {
        "width": 640,
```

```
        "height": 480,
        "depth": 3
    }
],
"annotations": [

],
"warnings": [
    {
        "code": "WARNING_NO_ATTRIBUTE_ANNOTATIONS",
        "message": "No attribute annotations were found."
    }
]
},
"bounding-box-metadata": {
    "objects": [

],
    "class-map": {

    },
    "type": "groundtruth/object-detection",
    "human-annotated": "yes",
    "creation-date": "2013-11-18 02:53:27",
    "job-name": "my job"
},
"warnings": [
    {
        "code": "WARNING_NO_ANNOTATIONS",
        "message": "No annotations were found."
    }
]
}
```

Pour corriger WARNING_NO_ATTRIBUTE_ANNOTATIONS

- Sélectionnez l'une des options suivantes :
 - Ajoutez un ou plusieurs annotation objets de cadre de délimitation à la ligne JSON. Pour plus d'informations, veuillez consulter [Localisation d'objets dans les fichiers manifestes](#).
 - Supprimez l'attribut du cadre de délimitation.

- Supprimez l'image (ligne JSON) du manifeste. Si d'autres attributs de cadre de délimitation valides existent dans la ligne JSON, vous pouvez supprimer uniquement l'attribut de cadre de délimitation non valide de la ligne JSON.

TYPE DE CAS D'UTILISATION D'ERREUR NON PRIS EN CHARGE

Message d'avertissement

En savoir plus

La valeur du type champ n'est pas `groundtruth/image-classification` ou `groundtruth/object-detection`. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).

```
{
  "source-ref": "s3://bucket/test_normal_8.jpg",
  "BB": {
    "annotations": [
      {
        "left": 1768,
        "top": 1007,
        "width": 448,
        "height": 295,
        "class_id": 0
      },
      {
        "left": 1794,
        "top": 1306,
        "width": 432,
        "height": 411,
        "class_id": 1
      },
      {
        "left": 2568,
        "top": 1346,
        "width": 710,
        "height": 305,
        "class_id": 2
      },
      {
        "left": 2571,
        "top": 1020,
        "width": 644,
```

```
        "height": 312,
        "class_id": 3
      }
    ],
    "image_size": [
      {
        "width": 4000,
        "height": 2667,
        "depth": 3
      }
    ]
  },
  "BB-metadata": {
    "job-name": "labeling-job/BB",
    "class-map": {
      "0": "comparator",
      "1": "pot_resistor",
      "2": "ir_phototransistor",
      "3": "ir_led"
    },
    "human-annotated": "yes",
    "objects": [
      {
        "confidence": 1
      },
      {
        "confidence": 1
      },
      {
        "confidence": 1
      },
      {
        "confidence": 1
      }
    ],
    "creation-date": "2021-06-22T09:58:34.811Z",
    "type": "groundtruth/wrongtype",
    "cl-errors": [
      {
        "code": "ERROR_UNSUPPORTED_USE_CASE_TYPE",
        "message": "The use case type of the BB-metadata label attribute
metadata is unsupported. Check the type field."
      }
    ]
  ]
}
```



```
  },
  "cl-metadata": {
    "is_labeled": true
  },
  "cl-errors": [
    {
      "code": "ERROR_NO_VALID_LABEL_ATTRIBUTES",
      "message": "No valid label attributes found."
    }
  ]
}
```

Pour corriger ERROR_UNSUPPORTED_USE_CASE_TYPE

- Choisissez l'une des options suivantes :
 - Remplacez la valeur du type champ par `groundtruth/image-classification` ou `groundtruth/object-detection`, selon le type de modèle que vous souhaitez créer. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).
 - Supprimez l'image (ligne JSON) du manifeste.

ERROR_INVALID_LABEL_NAME_LENGTH

En savoir plus

La longueur du nom d'une étiquette est trop longue. La longueur maximale est de 256 caractères.

Pour corriger ERROR_INVALID_LABEL_NAME_LENGTH

- Choisissez l'une des options suivantes :
 - Réduisez la longueur du nom de l'étiquette à 256 caractères ou moins.
 - Supprimez l'image (ligne JSON) du manifeste.

Amélioration d'un modèle d'étiquettes personnalisées

Amazon Rekognition certifié

Lorsque la formation est terminée, vous évaluez les performances du modèle. Pour vous aider, Amazon Rekognition Custom Labels fournit des statistiques récapitulatives et des mesures d'évaluation pour chaque étiquette. Pour plus d'informations sur les métriques disponibles, veuillez consulter [Métriques pour évaluer votre modèle](#). Pour améliorer votre modèle à l'aide de mesures, consultez [Amélioration d'un modèle Amazon Rekognition Custom Labels](#).

Si vous êtes satisfait de la précision de votre modèle, vous pouvez commencer à l'utiliser. Pour plus d'informations, veuillez consulter [Exécution d'un modèle Étiquettes personnalisées Amazon Rekognition entraîné](#).

Rubriques

- [Métriques pour évaluer votre modèle](#)
- [Accès aux mesures d'évaluation \(console\)](#)
- [Accès aux métriques d'évaluation \(SDK\) d'Amazon Rekognition Custom Labels](#)
- [Amélioration d'un modèle Amazon Rekognition Custom Labels](#)

Métriques pour évaluer votre modèle

Une fois votre modèle entraîné, Amazon Rekognition Custom Labels renvoie des mesures issues des tests du modèle, que vous pouvez utiliser pour évaluer les performances de votre modèle. Cette rubrique décrit les mesures à votre disposition et explique comment déterminer si votre modèle entraîné fonctionne correctement.

La console Amazon Rekognition Custom Labels fournit les mesures suivantes sous forme de résumé des résultats de formation et de mesures pour chaque étiquette :

- [Précision](#)
- [Sensibilité](#)
- [F1](#)

Chaque métrique que nous fournissons est une métrique couramment utilisée pour évaluer les performances d'un modèle Machine Learning. Amazon Rekognition Custom Labels renvoie des métriques pour les résultats des tests sur l'ensemble de données de test, ainsi que des métriques pour chaque étiquette personnalisée. Vous pouvez également vérifier les performances de votre modèle personnalisé entraîné pour chaque image de votre jeu de données de test. Pour plus d'informations, veuillez consulter [Accès aux mesures d'évaluation \(console\)](#).

Évaluation des performances de modèle

Lors des tests, Amazon Rekognition Custom Labels prédit si une image de test contient une étiquette personnalisée. Le score de confiance est une valeur qui quantifie la certitude de la prédiction du modèle.

Si le score de confiance d'une étiquette personnalisée dépasse la valeur seuil, la sortie du modèle inclura cette étiquette. Les prévisions peuvent être classées de la manière suivante :

- **Vrai positif** : le modèle Amazon Rekognition Custom Labels prédit correctement la présence de l'étiquette personnalisée dans l'image de test. En d'autres termes, l'étiquette prédite est également une étiquette de « vérité fondamentale » pour cette image. Par exemple, Amazon Rekognition Custom Labels renvoie correctement une étiquette de ballon de football lorsqu'un ballon de football est présent dans une image.
- **Faux positif** : le modèle d'étiquettes personnalisées Amazon Rekognition prédit de manière incorrecte la présence d'une étiquette personnalisée dans une image de test. En d'autres termes, l'étiquette prédite n'est pas une étiquette de base pour l'image. Par exemple, Amazon Rekognition Custom Labels renvoie une étiquette de ballon de football, mais aucune étiquette de ballon de football n'est exacte pour cette image.
- **Faux négatif** : le modèle Amazon Rekognition Custom Labels ne prédit pas la présence d'une étiquette personnalisée dans l'image, mais la « réalité de base » de cette image inclut cette étiquette. Par exemple, Amazon Rekognition Custom Labels ne renvoie pas d'étiquette personnalisée « ballon de football » pour une image contenant un ballon de football.
- **Vrai négatif** : le modèle d'étiquettes personnalisées Amazon Rekognition prédit correctement qu'aucune étiquette personnalisée n'est présente dans l'image de test. Par exemple, Amazon Rekognition Custom Labels ne renvoie pas d'étiquette de ballon de football pour une image qui ne contient pas de ballon de football.

La console permet d'accéder aux valeurs de vrai positif, de faux positif et de faux négatif pour chaque image de votre jeu de données de test. Pour plus d'informations, veuillez consulter [Accès aux mesures d'évaluation \(console\)](#).

Ces résultats de prédiction sont utilisés pour calculer les mesures suivantes pour chaque étiquette et un agrégat pour l'ensemble de votre ensemble de tests. Les mêmes définitions s'appliquent aux prédictions effectuées par le modèle au niveau du cadre de délimitation, à la différence que toutes les mesures sont calculées sur chaque zone de délimitation (prédiction ou réalité de terrain) dans chaque image de test.

Intersection au-dessus de l'Union (IoU) et détection d'objets

Intersection over Union (IoU) mesure le pourcentage de chevauchement entre deux zones de délimitation d'objets sur leur surface combinée. La plage est comprise entre 0 (chevauchement le plus faible) et 1 (chevauchement complet). Pendant les tests, une zone de délimitation prédite est correcte lorsque l'IoU de la zone de délimitation réelle au sol et de la zone de délimitation prédite est d'au moins 0,5.

Seuil supposé

Amazon Rekognition Custom Labels calcule automatiquement une valeur seuil supposée (0-1) pour chacune de vos étiquettes personnalisées. Vous ne pouvez pas définir la valeur de seuil supposée pour une étiquette personnalisée. Le seuil supposé pour chaque étiquette est la valeur au-dessus de laquelle une prédiction est considérée comme un vrai ou un faux positif. Il est défini en fonction de votre ensemble de données de test. Le seuil supposé est calculé sur la base du meilleur score F1 obtenu sur l'ensemble de données de test lors de l'entraînement du modèle.

Vous pouvez obtenir la valeur du seuil supposé pour une étiquette à partir des résultats d'entraînement du modèle. Pour plus d'informations, veuillez consulter [Accès aux mesures d'évaluation \(console\)](#).

Les modifications des valeurs seuils supposées sont généralement utilisées pour améliorer la précision et le rappel d'un modèle. Pour plus d'informations, veuillez consulter [Amélioration d'un modèle Amazon Rekognition Custom Labels](#). Comme vous ne pouvez pas définir le seuil supposé d'un modèle pour une étiquette, vous pouvez obtenir les mêmes résultats en analysant une image à l'aide d'un paramètre `MinConfidence` d'entrée `DetectCustomLabels` et en le spécifiant. Pour plus d'informations, veuillez consulter [Analyse d'une image avec un modèle entraîné](#).

Précision

Amazon Rekognition Custom Labels fournit des mesures de précision pour chaque étiquette et une métrique de précision moyenne pour l'ensemble de données de test.

La précision est la fraction des prédictions correctes (vrais positifs) par rapport à toutes les prédictions du modèle (vrais et faux positifs) au seuil supposé pour une étiquette individuelle. À mesure que le seuil augmente, le modèle peut faire moins de prévisions. En général, cependant, le ratio de vrais positifs par rapport aux faux positifs sera plus élevé par rapport à un seuil inférieur. Les valeurs de précision possibles sont comprises entre 0 et 1, et des valeurs plus élevées indiquent une précision plus élevée.

Par exemple, lorsque le modèle prédit la présence d'un ballon de football dans une image, à quelle fréquence cette prédiction est-elle correcte ? Supposons qu'il y ait une image avec 8 ballons de football et 5 pierres. Si le modèle prédit 9 ballons de football (8 correctement prédits et 1 faux positif), la précision de cet exemple est de 0,89. Toutefois, si le modèle a prédit 13 ballons de football dans l'image avec 8 prédictions correctes et 5 incorrectes, la précision résultante est inférieure.

Pour plus d'informations, consultez [Précision et rappel](#).

Sensibilité

Amazon Rekognition Custom Labels fournit des mesures de rappel moyennes pour chaque étiquette et une métrique de rappel moyenne pour l'ensemble de données de test.

Le rappel est la fraction des étiquettes de votre ensemble de test qui ont été correctement prédites au-dessus du seuil supposé. Il s'agit d'une mesure de la fréquence à laquelle le modèle peut prédire correctement une étiquette personnalisée alors qu'elle est réellement présente dans les images de votre ensemble de test. La plage de rappel est comprise entre 0 et 1. Des valeurs plus élevées indiquent un rappel plus élevé.

Par exemple, si une image contient 8 ballons de football, combien d'entre eux sont détectés correctement ? Dans cet exemple où une image contient 8 ballons de football et 5 pierres, si le modèle détecte 5 ballons de football, la valeur de rappel est de 0,62. Si, après le réentraînement, le nouveau modèle détecte 9 ballons de football, y compris les 8 présents sur l'image, la valeur de rappel est de 1,0.

Pour plus d'informations, consultez [Précision et rappel](#).

F1

Amazon Rekognition Custom Labels utilise la métrique de score F1 pour mesurer les performances moyennes du modèle de chaque étiquette et les performances moyennes du modèle de l'ensemble de données de test.

Les performances du modèle sont une mesure agrégée qui prend en compte à la fois la précision et le rappel sur toutes les étiquettes. (par exemple, score F1 ou précision moyenne). La note de performances du modèle est une valeur comprise entre 0 et 1. Plus la valeur est élevée, meilleures sont les performances du modèle en termes de rappel et de précision. Plus précisément, les performances du modèle pour les tâches de classification sont généralement mesurées par le score F1. Ce score est la moyenne harmonique des scores de précision et de rappel au seuil supposé. Par exemple, pour un modèle avec une précision de 0,9 et un rappel de 1,0, le score F1 est de 0,947.

Une valeur élevée pour le score F1 indique que le modèle fonctionne bien à la fois en termes de précision et de rappel. Si le modèle ne fonctionne pas bien, par exemple avec une faible précision de 0,30 et un rappel élevé de 1,0, le score F1 est de 0,46. De même, si la précision est élevée (0,95) et le rappel faible (0,20), le score F1 est de 0,33. Dans les deux cas, le score F1 est faible et indique des problèmes avec le modèle.

Pour plus d'informations, consultez [Score de F1](#).

Utilisation de métriques

Pour un modèle donné que vous avez entraîné et en fonction de votre application, vous pouvez trouver un compromis entre précision et rappel en utilisant le paramètre `MinConfidence` d'entrée `toDetectCustomLabels`. À une `MinConfidence` valeur plus élevée, vous obtenez généralement une plus grande précision (des prévisions plus correctes concernant les ballons de football), mais un rappel plus faible (un plus grand nombre de ballons de football réels seront oubliés). À une `MinConfidence` valeur plus faible, vous obtenez un meilleur rappel (plus de ballons de football réels sont correctement prédits), mais une précision moindre (un plus grand nombre de ces prédictions seront erronées). Pour plus d'informations, veuillez consulter [Analyse d'une image avec un modèle entraîné](#).

Les mesures vous informent également sur les mesures que vous pourriez prendre pour améliorer les performances du modèle si nécessaire. Pour plus d'informations, veuillez consulter [Amélioration d'un modèle Amazon Rekognition Custom Labels](#).

Note

`DetectCustomLabels` renvoie des prédictions comprises entre 0 et 100, qui correspondent à la plage métrique de 0 à 1.

Accès aux mesures d'évaluation (console)

Au cours des tests, les performances du modèle sont évaluées par rapport à l'ensemble de données de test. Les étiquettes de l'ensemble de données de test sont considérées comme une « vérité fondamentale » car elles représentent ce que représente l'image réelle. Pendant les tests, le modèle fait des prédictions à l'aide de l'ensemble de données de test. Les étiquettes prédites sont comparées aux étiquettes réelles et les résultats sont disponibles sur la page d'évaluation de la console.

La console Amazon Rekognition Custom Labels affiche des statistiques récapitulatives pour l'ensemble du modèle et des statistiques pour des étiquettes individuelles. Les mesures disponibles dans la console sont le rappel de précision, le score F1, la confiance et le seuil de confiance. Pour plus d'informations, veuillez consulter [Amélioration d'un modèle d'étiquettes personnalisées Amazon Rekognition certifié](#).

Vous pouvez utiliser la console pour vous concentrer sur des indicateurs individuels. Par exemple, pour étudier les problèmes de précision liés à une étiquette, vous pouvez filtrer les résultats d'apprentissage par étiquette et par faux positifs. Pour plus d'informations, veuillez consulter [Métriques pour évaluer votre modèle](#).

Après l'entraînement, le jeu de données d'entraînement est en lecture seule. Si vous décidez d'améliorer le modèle, vous pouvez copier le jeu de données d'apprentissage dans un nouveau jeu de données. Vous utilisez la copie du jeu de données pour entraîner une nouvelle version du modèle.

Dans cette étape, vous utilisez la console pour accéder aux résultats d'entraînement dans la console.

Pour accéder aux métriques d'évaluation (console)

1. Ouvrez la console Amazon Rekognition à l'[adresse https://console.aws.amazon.com/rekognition/](https://console.aws.amazon.com/rekognition/).
2. Choisissez Utiliser des étiquettes personnalisées.
3. Sélectionnez Get started (Démarrer).
4. Dans le panneau de navigation de gauche, choisissez Projects.

- Sur la page Projets, choisissez le projet qui contient le modèle entraîné que vous souhaitez évaluer.
- Dans Modèles, choisissez le modèle que vous souhaitez évaluer.
- Cliquez sur l'onglet Évaluation pour voir les résultats de l'évaluation. Pour plus d'informations sur l'évaluation d'un modèle, veuillez consulter [Amélioration d'un modèle d'étiquettes personnalisées Amazon Rekognition certifié](#).
- Choisissez Afficher les résultats des tests pour voir les résultats des images de test individuelles. Pour plus d'informations, veuillez consulter [Métriques pour évaluer votre modèle](#).

rooms_19 [Info](#) Delete model

Evaluate | Model details | Use Model | Tags

Evaluation results

View test results

F1 score Info 0.902	Average precision Info 0.893	Overall recall Info 0.928
Date completed July 13, 2021 Trained in 1.223 hours	Training dataset 10 labels, 61 images	Testing dataset 10 labels, 56 images

Per label performance (10)

Find labels < 1 >

Label name ▲	F1 score ▼	Test images ▼	Precision ▼	Recall ▼	Assumed threshold ▼
backyard	0.857	4	1.000	0.750	0.286
bathroom	0.889	9	0.889	0.889	0.185
bedroom	0.900	11	1.000	0.818	0.262
closet	1.000	2	1.000	1.000	0.169
entry_way	1.000	3	1.000	1.000	0.149
floor_plan	1.000	2	1.000	1.000	0.685

- Après avoir consulté les résultats du test, choisissez le nom du projet pour revenir à la page du modèle.

Custom Labels > Projects > rooms_19 > **rooms_19.2021-07-13T10.36.30** Performance

Evaluate image ✕

Review the test results of your trained model for individual images. Below each image is information about the model's predicted label compared with the label assigned to the image in the test dataset, noted by result type. You can also filter by label and result types.

Filter by label

Choose labels
Choose labels to filter images

True positive


False positive

False negative

Images (56) [Info](#)


< 1 2 3 4 ... >

backyard2.jpeg



Labels	Confidence
front_yard False positive	30.3%
backyard False negative	21.6%

backyard4.jpeg



Labels	Confidence
backyard True positive	46.3%

10. Utilisez les mesures pour évaluer les performances du modèle. Pour plus d'informations, veuillez consulter [Amélioration d'un modèle Amazon Rekognition Custom Labels](#).

Accès aux métriques d'évaluation (SDK) d'Amazon Rekognition Custom Labels

L'[DescribeProjectVersions](#) opération permet d'accéder à des mesures autres que celles fournies dans la console.

À l'instar de la console, elle `DescribeProjectVersions` permet d'accéder aux mesures suivantes sous forme d'informations récapitulatives pour les résultats des tests et de résultats de test pour chaque étiquette :

- [Précision](#)
- [Sensibilité](#)
- [F1](#)

Le seuil moyen pour toutes les étiquettes et le seuil pour les étiquettes individuelles sont renvoyés.

DescribeProjectVersions permet également d'accéder aux mesures suivantes pour la classification et la détection d'images (emplacement de l'objet sur l'image).

- Matrice de confusion pour la classification des images. Pour plus d'informations, veuillez consulter [Affichage de la matrice de confusion d'un modèle](#).
- Précision moyenne (mAP) pour la détection d'images.
- Rappel moyen (MaR) pour la détection d'images.

DescribeProjectVersions permet également d'accéder aux valeurs vraies positives, fausses positives, fausses négatives et vraies négatives. Pour plus d'informations, veuillez consulter [Métriques pour évaluer votre modèle](#).

La métrique du score F1 agrégé est renvoyée directement par DescribeProjectVersions. Les autres mesures sont accessibles à partir de [Aperçu du manifeste d'évaluation](#) fichiers [Fichier récapitulatif](#) et stockés dans un compartiment Amazon S3. Pour plus d'informations, veuillez consulter [Accès au fichier récapitulatif et à l'instantané du manifeste d'évaluation \(SDK\)](#).

Rubriques

- [Fichier récapitulatif](#)
- [Aperçu du manifeste d'évaluation](#)
- [Accès au fichier récapitulatif et à l'instantané du manifeste d'évaluation \(SDK\)](#)
- [Affichage de la matrice de confusion d'un modèle](#)
- [Référence : fichier récapitulatif des résultats de formation](#)

Fichier récapitulatif

Le fichier récapitulatif contient les résultats de l'évaluation, des informations sur le modèle dans son ensemble et des mesures pour chaque étiquette. Les mesures sont la précision, le rappel, le score F1. La valeur seuil du modèle est également fournie. L'emplacement du fichier récapitulatif est accessible depuis l'`EvaluationResult` objet renvoyé par DescribeProjectVersions. Pour plus d'informations, veuillez consulter [Référence : fichier récapitulatif des résultats de formation](#).

L'exemple suivant est un exemple de fichier récapitulatif.

```
{
  "Version": 1,
```

```
"AggregatedEvaluationResults": {
  "ConfusionMatrix": [
    {
      "GroundTruthLabel": "CAP",
      "PredictedLabel": "CAP",
      "Value": 0.9948717948717949
    },
    {
      "GroundTruthLabel": "CAP",
      "PredictedLabel": "WATCH",
      "Value": 0.008547008547008548
    },
    {
      "GroundTruthLabel": "WATCH",
      "PredictedLabel": "CAP",
      "Value": 0.1794871794871795
    },
    {
      "GroundTruthLabel": "WATCH",
      "PredictedLabel": "WATCH",
      "Value": 0.7008547008547008
    }
  ],
  "F1Score": 0.9726959470546408,
  "Precision": 0.9719115848331294,
  "Recall": 0.9735042735042735
},
"EvaluationDetails": {
  "EvaluationEndTimestamp": "2019-11-21T07:30:23.910943",
  "Labels": [
    "CAP",
    "WATCH"
  ],
  "NumberOfTestingImages": 624,
  "NumberOfTrainingImages": 5216,
  "ProjectVersionArn": "arn:aws:rekognition:us-east-1:nnnnnnnn:project/my-project/
version/v0/1574317227432"
},
"LabelEvaluationResults": [
  {
    "Label": "CAP",
    "Metrics": {
      "F1Score": 0.9794344473007711,
      "Precision": 0.9819587628865979,
```

```
    "Recall": 0.9769230769230769,  
    "Threshold": 0.9879502058029175  
  },  
  "NumberOfTestingImages": 390  
},  
{  
  "Label": "WATCH",  
  "Metrics": {  
    "F1Score": 0.9659574468085106,  
    "Precision": 0.961864406779661,  
    "Recall": 0.9700854700854701,  
    "Threshold": 0.014450683258473873  
  },  
  "NumberOfTestingImages": 234  
}  
]  
}
```

Aperçu du manifeste d'évaluation

L'instantané du manifeste d'évaluation contient des informations détaillées sur les résultats des tests. L'instantané inclut l'indice de confiance pour chaque prédiction. Elle inclut également la classification de la prédiction par rapport à la classification réelle de l'image (vrai positif, vrai négatif, faux positif ou faux négatif).

Les fichiers sont un instantané puisque seules les images pouvant être utilisées pour les tests et la formation sont incluses. Les images qui ne peuvent pas être vérifiées, telles que les images dont le format est incorrect, ne sont pas incluses dans le manifeste. L'emplacement du snapshot de test est accessible depuis l'`TestingDataResult` objet renvoyé par `DescribeProjectVersions`. L'emplacement du cliché d'entraînement est accessible à partir de `TrainingDataResult` l'objet renvoyé par `DescribeProjectVersions`.

L'instantané est au format de sortie du manifeste SageMaker Ground Truth avec des champs ajoutés pour fournir des informations supplémentaires, telles que le résultat de la classification binaire d'une détection. L'extrait suivant montre les champs supplémentaires.

```
"rekognition-custom-labels-evaluation-details": {  
  "version": 1,  
  "is-true-positive": true,  
  "is-true-negative": false,  
  "is-false-positive": false,
```

```
"is-false-negative": false,
"is-present-in-ground-truth": true
"ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"]
}
```

- `version` : version du format de `rekognition-custom-labels-evaluation-details` champ dans l'instantané du manifeste.
- `is-true-positive...` — Classification binaire de la prédiction basée sur la comparaison entre le score de confiance et le seuil minimum pour l'étiquette.
- `is-present-in-ground-truth` — Vrai si la prédiction faite par le modèle est présente dans les informations de base utilisées pour l'entraînement, sinon faux. Cette valeur ne dépend pas du fait que le score de confiance dépasse le seuil minimum calculé par le modèle.
- `ground-truth-labeling-jobs`— Une liste des champs de base du manifeste qui sont utilisés pour la formation.

Pour plus d'informations sur le format du manifeste SageMaker Ground Truth, voir [Output](#).

Voici un exemple de capture instantanée du manifeste de test qui présente des mesures pour la classification des images et la détection d'objets.

```
// For image classification
{
  "source-ref": "s3://test-bucket/dataset/beckham.jpeg",
  "rekognition-custom-labels-training-0": 1,
  "rekognition-custom-labels-training-0-metadata": {
    "confidence": 1.0,
    "job-name": "rekognition-custom-labels-training-job",
    "class-name": "Football",
    "human-annotated": "yes",
    "creation-date": "2019-09-06T00:07:25.488243",
    "type": "groundtruth/image-classification"
  },
  "rekognition-custom-labels-evaluation-0": 1,
  "rekognition-custom-labels-evaluation-0-metadata": {
    "confidence": 0.95,
    "job-name": "rekognition-custom-labels-evaluation-job",
    "class-name": "Football",
    "human-annotated": "no",
    "creation-date": "2019-09-06T00:07:25.488243",
    "type": "groundtruth/image-classification",
  }
}
```

```
"rekognition-custom-labels-evaluation-details": {
  "version": 1,
  "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"],
  "is-true-positive": true,
  "is-true-negative": false,
  "is-false-positive": false,
  "is-false-negative": false,
  "is-present-in-ground-truth": true
}
}
}

// For object detection
{
  "source-ref": "s3://test-bucket/dataset/beckham.jpeg",
  "rekognition-custom-labels-training-0": {
    "annotations": [
      {
        "class_id": 0,
        "width": 39,
        "top": 409,
        "height": 63,
        "left": 712
      },
      ...
    ],
    "image_size": [
      {
        "width": 1024,
        "depth": 3,
        "height": 768
      }
    ]
  },
  "rekognition-custom-labels-training-0-metadata": {
    "job-name": "rekognition-custom-labels-training-job",
    "class-map": {
      "0": "Cap",
      ...
    },
    "human-annotated": "yes",
    "objects": [
      {
```

```
    "confidence": 1.0
  },
  ...
],
"creation-date": "2019-10-21T22:02:18.432644",
"type": "groundtruth/object-detection"
},
"rekognition-custom-labels-evaluation": {
  "annotations": [
    {
      "class_id": 0,
      "width": 39,
      "top": 409,
      "height": 63,
      "left": 712
    },
    ...
  ],
  "image_size": [
    {
      "width": 1024,
      "depth": 3,
      "height": 768
    }
  ]
},
"rekognition-custom-labels-evaluation-metadata": {
  "confidence": 0.95,
  "job-name": "rekognition-custom-labels-evaluation-job",
  "class-map": {
    "0": "Cap",
    ...
  },
  "human-annotated": "no",
  "objects": [
    {
      "confidence": 0.95,
      "rekognition-custom-labels-evaluation-details": {
        "version": 1,
        "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"],
        "is-true-positive": true,
        "is-true-negative": false,
        "is-false-positive": false,
        "is-false-negative": false,
      }
    }
  ]
}
```

```
        "is-present-in-ground-truth": true
    }
},
...
],
"creation-date": "2019-10-21T22:02:18.432644",
"type": "groundtruth/object-detection"
}
}
```

Accès au fichier récapitulatif et à l'instantané du manifeste d'évaluation (SDK)

Pour obtenir les résultats de l'entraînement, vous appelez [DescribeProjectVersions](#). Pour obtenir un exemple de code, veuillez consulter [Décrire un modèle \(SDK\)](#).

L'emplacement des métriques est renvoyé dans le formulaire de `ProjectVersionDescription` réponse `DescribeProjectVersions`.

- `EvaluationResult`— Emplacement du fichier récapitulatif.
- `TestingDataResult`— Emplacement de l'instantané du manifeste d'évaluation utilisé pour les tests.

Le score de F1 et l'emplacement du fichier de synthèse sont renvoyés dans `EvaluationResult`. Par exemple :

```
"EvaluationResult": {
    "F1Score": 1.0,
    "Summary": {
        "S3Object": {
            "Bucket": "echo-dot-scans",
            "Name": "test-output/EvaluationResultSummary-my-echo-dots-
project-v2.json"
        }
    }
}
```

L'instantané du manifeste d'évaluation est stocké à l'emplacement spécifié dans le paramètre `--output-config` d'entrée que vous avez spécifié [Entraînement d'un modèle \(SDK\)](#).

Note

La quantité de temps, en secondes, durant laquelle vous êtes facturé pour la formation est restituée `BillableTrainingTimeInSeconds`.

Pour plus d'informations sur les statistiques renvoyées par les étiquettes personnalisées Amazon Rekognition, consultez [Accès aux métriques d'évaluation \(SDK\) d'Amazon Rekognition Custom Labels](#).

Affichage de la matrice de confusion d'un modèle

Une matrice de confusion vous permet de voir les étiquettes que votre modèle confond avec d'autres étiquettes de votre modèle. En utilisant une matrice de confusion, vous pouvez concentrer vos améliorations sur le modèle.

Lors de l'évaluation du modèle, les étiquettes personnalisées Amazon Rekognition créent une matrice de confusion en utilisant les images de test pour identifier les étiquettes mal identifiées (confuses). Les étiquettes personnalisées Amazon Rekognition ne créent une matrice de confusion que pour les modèles de classification. La matrice de classification est accessible à partir du fichier récapitulatif créé par Amazon Rekognition Custom Labels lors de la formation des modèles. Vous ne pouvez pas consulter la matrice de confusion dans la console Amazon Rekognition Custom Labels.

Rubriques

- [Utiliser une matrice de confusion](#)
- [Obtenir la matrice de confusion pour un modèle](#)

Utiliser une matrice de confusion

Le tableau suivant présente la matrice de confusion pour l'exemple de projet de [classification d'images Rooms](#). Les en-têtes de colonne sont les étiquettes (étiquettes de base) attribuées aux images de test. Les en-têtes de ligne sont les étiquettes que le modèle prédit pour les images de test. Chaque cellule représente le pourcentage de prédictions pour une étiquette (ligne) qui doit être l'étiquette de base (colonne). Par exemple, 67 % des prévisions concernant les salles de bains étaient correctement étiquetées comme des salles de bains. 33 % des salles de bains étaient mal étiquetées comme des cuisines. Un modèle très performant possède des valeurs de cellule élevées lorsque l'étiquette prédite correspond à l'étiquette de base. Vous pouvez les voir sous la forme d'une

ligne diagonale allant de la première à la dernière étiquette prédite et de vérité fondamentale. Si la valeur d'une cellule est 0, aucune prédiction n'a été faite pour l'étiquette prédite de la cellule qui devrait être l'étiquette fondamentale de la cellule.

Note

Les modèles n'étant pas déterministes, les valeurs des cellules de la matrice de confusion que vous obtenez lors de l'entraînement au projet Rooms peuvent différer de celles du tableau suivant.

La matrice de confusion identifie les domaines sur lesquels se concentrer. Par exemple, la matrice de confusion montre que 50 % du temps, le modèle a confondu placards et chambres à coucher. Dans ce cas, vous devez ajouter d'autres images de placards et de chambres à votre jeu de données d'entraînement. Vérifiez également que les images existantes du placard et de la chambre sont correctement étiquetées. Cela devrait aider le modèle à mieux distinguer les deux étiquettes. Pour ajouter d'autres images à un jeu de données, consultez [Ajouter d'autres images à un jeu de données](#).

Bien que la matrice de confusion soit utile, il est important de prendre en compte d'autres indicateurs. Par exemple, 100 % des prédictions ont correctement trouvé l'étiquette `floor_plan`, ce qui indique d'excellentes performances. Toutefois, le jeu de données de test ne contient que 2 images portant l'étiquette `floor_plan`. Il contient également 11 images portant le label `living_space`. Ce déséquilibre se retrouve également dans l'ensemble de données d'entraînement (13 images `living_space` et 2 images de placard). Pour obtenir une évaluation plus précise, équilibrez les ensembles de données d'entraînement et de test en ajoutant des images supplémentaires d'étiquettes sous-représentées (plans d'étage dans cet exemple). Pour obtenir le nombre d'images de test par étiquette, voir [Accès aux mesures d'évaluation \(console\)](#).

Étiquetage Ground Truth

Étiquette prédite	cour arrière	salle de bain	chambre	placard	voie d'entrée	plan_étage	jardin à l'avant	cuisine	espace de vie	patio
cour arrière	75%	0%	0%	0%	0%	0%	33%	0%	0%	0%

Étiquetage Ground Truth

Étiquette prédite	cour arrière	salle de bain	chambre	placard	voie d'entrée	plan_étage	jardin à l'avant	cuisine	espace de vie	patio
salle de bain	0%	67%	0%	0%	0%	0%	0%	0%	0%	0%
chambre	0%	0%	82%	50%	0%	0%	0%	0%	9%	0%
placard	0%	0%	0%	50%	0%	0%	0%	0%	0%	0%
voie d'entrée	0%	0%	0%	0%	33%	0%	0%	0%	0%	0%
plan_étage	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
jardin à l'avant	25%	0%	0%	0%	0%	0%	67%	0%	0%	0%
cuisine	0%	33%	0%	0%	0%	0%	0%	88%	0%	0%
espace de vie	0%	0%	18%	0%	67%	0%	0%	12%	91%	33%
patio	0%	0%	0%	0%	0%	0%	0%	0%	0%	67%

Obtenir la matrice de confusion pour un modèle

Le code suivant utilise les [DescribeProjectVersions](#) opérations [DescribeProjects](#) et pour obtenir le [fichier récapitulatif](#) d'un modèle. Il utilise ensuite le fichier récapitulatif pour afficher la matrice de confusion du modèle.

Pour afficher la matrice de confusion d'un modèle (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les SDK AWS CLI et les AWS SDK. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez le code suivant pour afficher la matrice de confusion d'un modèle. Fournissez les arguments de ligne de commande suivants :
 - `project_name`— le nom du projet que vous souhaitez utiliser. Vous pouvez obtenir le nom du projet sur la page des projets de la console Amazon Rekognition Custom Labels.
 - `version_name`— la version du modèle que vous souhaitez utiliser. Vous pouvez obtenir le nom de la version sur la page des détails du projet dans la console Amazon Rekognition Custom Labels.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose

Shows how to display the confusion matrix for an Amazon Rekognition Custom labels
image
classification model.
"""

import json
import argparse
import logging
import boto3
import pandas as pd
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def get_model_summary_location(rek_client, project_name, version_name):
    """
    Get the summary file location for a model.

    :param rek_client: A Boto3 Rekognition client.
```

```
:param project_arn: The Amazon Resource Name (ARN) of the project that contains
the model.
:param model_arn: The Amazon Resource Name (ARN) of the model.
:return: The location of the model summary file.
"""

try:
    logger.info(
        "Getting summary file for model %s in project %s.", version_name,
        project_name)

    summary_location = ""

    # Get the project ARN from the project name.
    response = rek_client.describe_projects(ProjectNames=[project_name])

    assert len(response['ProjectDescriptions']) > 0, \
        f"Project {project_name} not found."

    project_arn = response['ProjectDescriptions'][0]['ProjectArn']

    # Get the summary file location for the model.
    describe_response =
rek_client.describe_project_versions(ProjectArn=project_arn,
VersionNames=[version_name])
    assert len(describe_response['ProjectVersionDescriptions']) > 0, \
        f"Model {version_name} not found."

    model=describe_response['ProjectVersionDescriptions'][0]

    evaluation_results=model['EvaluationResult']

    summary_location=(f"s3://{evaluation_results['Summary']['S3Object']
['Bucket']}"
                    f"/{evaluation_results['Summary']['S3Object']
['Name']}")

    return summary_location

except ClientError as err:
    logger.exception(
        "Couldn't get summary file location: %s", err.response['Error']
['Message'])
```

```
raise

def show_confusion_matrix(summary):
    """
    Shows the confusion matrix for an Amazon Rekognition Custom Labels
    image classification model.
    :param summary: The summary file JSON object.
    """
    pd.options.display.float_format = '{:.0%}'.format

    # Load the model summary JSON into a DataFrame.

    summary_df = pd.DataFrame(
        summary['AggregatedEvaluationResults']['ConfusionMatrix'])

    # Get the confusion matrix.
    confusion_matrix = summary_df.pivot_table(index='PredictedLabel',
                                               columns='GroundTruthLabel',
                                               fill_value=0.0).astype(float)

    # Display the confusion matrix.
    print(confusion_matrix)

def get_summary(s3_resource, summary):
    """
    Gets the summary file.
    : return: The summary file in bytes.
    """
    try:
        summary_bucket, summary_key = summary.replace(
            "s3://", "").split("/", 1)

        bucket = s3_resource.Bucket(summary_bucket)
        obj = bucket.Object(summary_key)
        body = obj.get()['Body'].read()
        logger.info(
            "Got summary file '%s' from bucket '%s'.",
            obj.key, obj.bucket_name)
    except ClientError:
        logger.exception(
            "Couldn't get summary file '%s' from bucket '%s'.",
            obj.key, obj.bucket_name)
```

```
        raise
    else:
        return body

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    : param parser: The command line parser.
    """

    parser.add_argument(
        "project_name", help="The ARN of the project in which the model resides."
    )
    parser.add_argument(
        "version_name", help="The version of the model that you want to describe."
    )

def main():
    """
    Entry point for script.
    """

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get the command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Showing confusion matrix for: {args.version_name} for project
{args.project_name}.")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")
        s3_resource = session.resource('s3')

        # Get the summary file for the model.
```

```
summary_location = get_model_summary_location(rekognition_client,
args.project_name,
                                             args.version_name
                                             )
summary = json.loads(get_summary(s3_resource, summary_location))

# Check that the confusion matrix is available.
assert 'ConfusionMatrix' in summary['AggregatedEvaluationResults'], \
    "Confusion matrix not found in summary. Is the model a classification
model?"

# Show the confusion matrix.
show_confusion_matrix(summary)
print("Done")

except ClientError as err:
    logger.exception("Problem showing confusion matrix: %s", err)
    print(f"Problem describing model: {err}")

except AssertionError as err:
    logger.exception(
        "Error: %s.\n", err)
    print(
        f"Error: {err}\n")

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

Référence : fichier récapitulatif des résultats de formation

Le résumé des résultats de formation contient des mesures que vous pouvez utiliser pour évaluer votre modèle. Le fichier récapitulatif est également utilisé pour afficher les mesures sur la page des résultats de formation de la console. Le fichier récapitulatif est stocké dans un compartiment Amazon S3 après l'entraînement. Pour obtenir le fichier récapitulatif, appelez `DescribeProjectVersion`. Pour obtenir un exemple de code, veuillez consulter [Accès au fichier récapitulatif et à l'instantané du manifeste d'évaluation \(SDK\)](#).

Fichier récapitulatif

Le format JSON suivant correspond au format du fichier récapitulatif.

EvaluationDetails (article 3)

Informations générales sur la tâche d'entraînement. Cela inclut l'ARN du projet auquel appartient le modèle (`ProjectVersionArn`), la date et l'heure de fin de la formation, la version du modèle qui a été évaluée (`EvaluationEndTimeStamp`) et la liste des étiquettes détectées pendant la formation (`Labels`). Le nombre d'images utilisées pour la formation (`NumberOfTrainingImages`) et l'évaluation (`NumberOfTestingImages`) est également inclus.

AggregatedEvaluationResults (article 1)

Vous pouvez utiliser `AggregatedEvaluationResults` pour évaluer les performances globales du modèle entraîné lorsqu'il est utilisé avec le jeu de données de test. Des mesures agrégées sont incluses pour `PrecisionRecall`, et `F1Score` des métriques. Pour la détection d'objets (l'emplacement de l'objet sur une image), des métriques `AverageRecall` (MaR) et `AveragePrecision` (mAP) sont renvoyées. Pour la classification (le type d'objet dans une image), une métrique de matrice de confusion est renvoyée.

LabelEvaluationResults (article 2)

Vous pouvez utiliser `LabelEvaluationResults` pour évaluer les performances de chaque étiquette. Les étiquettes sont triées en fonction du score F1 de chaque étiquette. Les mesures incluses sont `Precision`, `Recall`, `F1Score`, et `Threshold` (utilisées pour la classification).

Le nom du fichier est formaté comme suit : `EvaluationSummary-ProjectName-VersionName.json`.

```
{
  "Version": "integer",
  // section-3
  "EvaluationDetails": {
    "ProjectVersionArn": "string",
    "EvaluationEndTimeStamp": "string",
    "Labels": "[string]",
    "NumberOfTrainingImages": "int",
    "NumberOfTestingImages": "int"
  },
  // section-1
  "AggregatedEvaluationResults": {
    "Metrics": {
      "Precision": "float",
      "Recall": "float",
```

```
"F1Score": "float",
// The following 2 fields are only applicable to object detection
"AveragePrecision": "float",
"AverageRecall": "float",
// The following field is only applicable to classification
"ConfusionMatrix":[
  {
    "GroundTruthLabel": "string",
    "PredictedLabel": "string",
    "Value": "float"
  },
  ...
],
}
},
// section-2
"LabelEvaluationResults": [
  {
    "Label": "string",
    "NumberOfTestingImages", "int",
    "Metrics": {
      "Threshold": "float",
      "Precision": "float",
      "Recall": "float",
      "F1Score": "float"
    },
  },
  ...
]
}
```

Amélioration d'un modèle Amazon Rekognition Custom Labels

Les performances des modèles d'apprentissage automatique dépendent largement de facteurs tels que la complexité et la variabilité de vos étiquettes personnalisées (les objets et scènes spécifiques qui vous intéressent), la qualité et la puissance représentative du jeu de données d'apprentissage que vous fournissez, ainsi que les cadres de modèles et les méthodes d'apprentissage automatique utilisés pour entraîner le modèle.

Amazon Rekognition Custom Labels simplifie ce processus et aucune expertise en matière d'apprentissage automatique n'est requise. Cependant, le processus de création d'un bon modèle implique souvent des itérations sur les données et des améliorations du modèle pour atteindre les

performances souhaitées. Vous trouverez ci-dessous des informations sur la manière d'améliorer votre modèle.

non structurées

En général, vous pouvez améliorer la qualité de votre modèle avec de plus grandes quantités de données de meilleure qualité. Utilisez des images d'apprentissage qui montrent clairement l'objet ou la scène et qui ne sont pas encombrées d'objets inutiles. Pour délimiter des zones autour d'objets, utilisez des images d'apprentissage qui montrent l'objet entièrement visible et qui ne sont pas masqués par d'autres objets.

Assurez-vous que vos ensembles de données d'entraînement et de test correspondent au type d'images sur lesquelles vous allez éventuellement effectuer des inférences. Pour les objets, tels que les logos, pour lesquels vous ne disposez que de quelques exemples d'apprentissage, vous devez placer des cadres de délimitation autour du logo dans vos images de test. Ces images représentent ou illustrent les scénarios dans lesquels vous souhaitez localiser l'objet.

Pour ajouter d'autres images à un jeu de données d'entraînement ou de test, consultez [Ajouter d'autres images à un jeu de données](#).

Réduire les faux positifs (meilleure précision)

- Tout d'abord, vérifiez si l'augmentation du seuil supposé vous permet de conserver les prévisions correctes, tout en diminuant le nombre de faux positifs. À un moment donné, cela se traduit par des gains décroissants en raison du compromis entre précision et rappel pour un modèle donné. Vous ne pouvez pas définir le seuil supposé pour une étiquette, mais vous pouvez obtenir le même résultat en spécifiant une valeur élevée pour le paramètre `MinConfidence` d'entrée `toDetectCustomLabels`. Pour plus d'informations, veuillez consulter [Analyse d'une image avec un modèle entraîné](#).
- Il se peut qu'une ou plusieurs de vos étiquettes personnalisées qui vous intéressent (A) soient constamment confondues avec la même classe d'objets (mais pas une étiquette qui vous intéresse) (B). Pour vous aider, ajoutez B comme étiquette de classe d'objet à votre jeu de données d'apprentissage (avec les images sur lesquelles vous avez obtenu le faux positif). En fait, vous aidez le modèle à apprendre à prédire B et non A grâce aux nouvelles images d'entraînement. Pour ajouter des images à un jeu de données d'entraînement, consultez [Ajouter d'autres images à un jeu de données](#).
- Vous constaterez peut-être que le modèle est confus à cause de deux de vos étiquettes personnalisées (A et B) : l'image de test portant l'étiquette A est censée porter l'étiquette B

et vice versa. Dans ce cas, vérifiez d'abord qu'il n'y a pas d'images mal étiquetées dans vos kits d'entraînement et de test. Utilisez la galerie de jeux de données pour gérer les étiquettes attribuées à un ensemble de données. Pour plus d'informations, veuillez consulter [Gestion des étiquettes](#). En outre, l'ajout d'images d'apprentissage liées à ce type de confusion aidera un modèle réentraîné à mieux faire la distinction entre A et B. Pour ajouter des images à un jeu de données d'apprentissage, voir [Ajouter d'autres images à un jeu de données](#).

Réduire les faux négatifs (meilleure mémorisation)

- Utilisez une valeur inférieure pour le seuil supposé. Vous ne pouvez pas définir le seuil supposé pour une étiquette, mais vous pouvez obtenir le même résultat en spécifiant un paramètre `MinConfidence` d'entrée inférieur à `DetectCustomLabels`. Pour plus d'informations, veuillez consulter [Analyse d'une image avec un modèle entraîné](#).
- Utilisez de meilleurs exemples pour modéliser la variété de l'objet et des images dans lesquelles il apparaît.
- Divisez votre étiquette en deux catégories plus faciles à apprendre. Par exemple, au lieu de bons et de mauvais cookies, vous pouvez préférer de bons cookies, des cookies brûlés et des cookies cassés pour aider le modèle à mieux comprendre chaque concept unique.

Exécution d'un modèle Étiquettes personnalisées Amazon Rekognition entraîné

Lorsque vous êtes satisfait des performances du modèle, vous pouvez commencer à l'utiliser. Vous pouvez démarrer et arrêter un modèle à l'aide de la console ou du kit AWS SDK. La console inclut également des exemples d'opérations du kit SDK que vous pouvez utiliser.

Rubriques

- [Unités d'inférence](#)
- [Zones de disponibilité](#)
- [Démarrage d'un modèle Étiquettes personnalisées Amazon Rekognition](#)
- [Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition](#)
- [Rapport sur la durée d'exécution et les unités d'inférence utilisées](#)

Unités d'inférence

Lorsque vous démarrez votre modèle, vous spécifiez le nombre de ressources de calcul, appelées unités d'inférence, utilisées par le modèle.

Important

Vous êtes facturé en fonction du nombre d'heures d'exécution de votre modèle et du nombre d'unités d'inférence qu'il utilise pendant son exécution, en fonction de la façon dont vous configurez l'exécution de votre modèle. Par exemple, si vous démarrez le modèle avec deux unités d'inférence et que vous l'utilisez pendant 8 heures, 16 heures d'inférence vous sont facturées (8 heures d'exécution * deux unités d'inférence). Pour plus d'informations, consultez [Heures d'inférence](#). Si vous n'**arrêtez pas explicitement votre modèle**, des frais vous seront facturés même si vous n'analysez pas activement les images avec votre modèle.

Les transactions par seconde (TPS) prises en charge par une seule unité d'inférence sont affectées par les facteurs suivants.

- Un modèle qui détecte des étiquettes au niveau de l'image (classification) a généralement un nombre TPS plus élevé qu'un modèle qui détecte et localise des objets à l'aide de cadres de délimitation (détection d'objets).
- La complexité du modèle.
- Une image avec une résolution plus élevée nécessite plus de temps pour l'analyse.
- Un plus grand nombre d'objets dans une image nécessite plus de temps pour l'analyse.
- Les petites images sont analysées plus rapidement que les images plus grandes.
- Une image transmise sous forme d'octets d'image est analysée plus rapidement que lorsque vous l'avez d'abord téléchargée dans un compartiment Amazon S3, puis que vous la référencez. Les images transmises sous forme d'octets d'image doivent avoir une taille inférieure à 4 Mo. Nous vous recommandons d'utiliser des octets d'image pour le traitement des images en temps quasi réel et lorsque la taille de l'image est inférieure à 4 Mo. Par exemple, des images capturées par une caméra IP.
- Le traitement des images stockées dans un compartiment Amazon S3 est plus rapide que de télécharger des images, les convertir en octets d'image, puis les transmettre à des fins d'analyse.
- L'analyse d'une image déjà stockée dans un compartiment Amazon S3 est probablement plus rapide que d'analyser la même image transmise sous forme d'octets d'image. Cela est particulièrement vrai si la taille de l'image est plus grande.

Si le nombre d'appels à `DetectCustomLabels` dépasse le nombre TPS maximum pris en charge par la somme des unités d'inférence utilisées par un modèle, Étiquettes personnalisées Amazon Rekognition renvoie une exception `ProvisionedThroughputExceededException`.

Gestion du débit à l'aide d'unités d'inférence

Vous pouvez augmenter ou diminuer le débit de votre modèle en fonction des exigences de votre application. Pour augmenter le débit, utilisez des unités d'inférence supplémentaires. Chaque unité d'inférence supplémentaire augmente votre vitesse de traitement d'une unité d'inférence. Pour plus d'informations sur le calcul du nombre d'unités d'inférence dont vous avez besoin, consultez [Calculate inference units for Amazon Rekognition Custom Labels and Amazon Lookout for Vision models](#). Si vous souhaitez modifier le débit pris en charge par votre modèle, deux options s'offrent à vous :

Ajout ou suppression des unités d'inférence manuellement

[Arrêtez](#) le modèle, puis [redémarrez-le](#) avec le nombre d'unités d'inférence requis. L'inconvénient de cette approche est que le modèle ne peut pas recevoir de demandes pendant le redémarrage et qu'il ne peut pas être utilisé pour gérer les pics de demande. Utilisez cette approche si le débit de votre modèle est stable et que votre cas d'utilisation peut tolérer 10 à 20 minutes d'indisponibilité. Par exemple, si vous souhaitez effectuer des appels groupés vers votre modèle selon un calendrier hebdomadaire.

Mise à l'échelle automatique d'unités d'inférence

Si votre modèle doit faire face à des pics de demande, la fonctionnalité Étiquettes personnalisées Amazon Rekognition peut automatiquement mettre à l'échelle le nombre d'unités d'inférence utilisées par votre modèle. À mesure que la demande augmente, la fonctionnalité Étiquettes personnalisées Amazon Rekognition ajoute des unités d'inférence supplémentaires au modèle et les supprime lorsque la demande diminue.

Pour permettre à Étiquettes personnalisées Amazon Rekognition de mettre à l'échelle automatiquement les unités d'inférence d'un modèle, [démarez](#) le modèle et définissez le nombre maximum d'unités d'inférence qu'il peut utiliser à l'aide du paramètre `MaxInferenceUnits`. La définition d'un nombre maximum d'unités d'inférence vous permet de gérer le coût d'exécution du modèle en limitant le nombre d'unités d'inférence disponibles. Si vous ne spécifiez pas de nombre maximum d'unités, la fonctionnalité Étiquettes personnalisées Amazon Rekognition ne mettra pas automatiquement à l'échelle votre modèle, mais utilisera uniquement le nombre d'unités d'inférence avec lequel vous avez commencé. Pour plus d'informations concernant le nombre maximum d'unités d'inférence, consultez [Service Quotas](#).

Vous pouvez également spécifier un nombre minimum d'unités d'inférence à l'aide du paramètre `MinInferenceUnits`. Cela vous permet de spécifier le débit minimum pour votre modèle, où une seule unité d'inférence représente 1 heure de traitement.

Note

Vous ne pouvez pas définir le nombre maximum d'unités d'inférence avec la console Étiquettes personnalisées Amazon Rekognition. Spécifiez plutôt le paramètre d'entrée `MaxInferenceUnits` de l'opération `StartProjectVersion`.

Les étiquettes personnalisées Amazon Rekognition fournissent les métriques CloudWatch Amazon Logs suivantes que vous pouvez utiliser pour déterminer l'état actuel du dimensionnement automatique d'un modèle.

Métrique	Description
<code>DesiredInferenceUnits</code>	Le nombre d'unités d'inférence par rapport auxquelles la fonctionnalité Étiquettes personnalisées Amazon Rekognition est augmentée ou réduite.
<code>InServiceInferenceUnits</code>	Le nombre d'unités d'inférence utilisées par le modèle.

Si `DesiredInferenceUnits` = `InServiceInferenceUnits`, la fonctionnalité Étiquettes personnalisées Amazon Rekognition ne met pas actuellement à l'échelle le nombre d'unités d'inférence.

Si `DesiredInferenceUnits` > `InServiceInferenceUnits`, la fonctionnalité Étiquettes personnalisées Amazon Rekognition augmente à la valeur de `DesiredInferenceUnits`.

Si `DesiredInferenceUnits` < `InServiceInferenceUnits`, la fonctionnalité Étiquettes personnalisées Amazon Rekognition est réduite à la valeur de `DesiredInferenceUnits`.

[Pour plus d'informations concernant les métriques renvoyées par les étiquettes personnalisées Amazon Rekognition et les dimensions de filtrage, CloudWatch consultez les métriques pour Rekognition.](#)

Pour connaître le nombre maximum d'unités d'inférence que vous avez demandées pour un modèle, appelez `DescribeProjectsVersion` et vérifiez le champ `MaxInferenceUnits` dans la réponse. Pour obtenir un exemple de code, consultez [Décrire un modèle \(SDK\)](#).

Zones de disponibilité

La fonctionnalité Étiquettes personnalisées Amazon Rekognition distribue les unités d'inférence dans plusieurs zones de disponibilité au sein d'une région AWS afin d'augmenter la disponibilité. Pour plus d'informations, consultez [Zones de disponibilité](#). Pour protéger vos modèles de production contre les

pannes de zone de disponibilité et les défaillances des unités d'inférence, démarrez vos modèles de production avec au moins deux unités d'inférence.

En cas de panne de la zone de disponibilité, toutes les unités d'inférence de la zone de disponibilité ne sont pas disponibles et la capacité du modèle est réduite. Les appels à [DetectCustomLabels](#) sont redistribués entre les unités d'inférence restantes. Ces appels réussissent s'ils ne dépassent pas les transactions par seconde (TPS) prises en charge par les unités d'inférence restantes. Une fois qu'AWS a réparé la zone de disponibilité, les unités d'inférence sont redémarrées et leur capacité maximum est rétablie.

Si une seule unité d'inférence échoue, la fonctionnalité Étiquettes personnalisées Amazon Rekognition lance automatiquement une nouvelle unité d'inférence dans la même zone de disponibilité. La capacité du modèle est réduite jusqu'au démarrage de la nouvelle unité d'inférence.

Démarrage d'un modèle Étiquettes personnalisées Amazon Rekognition

Vous pouvez commencer à exécuter un modèle d'étiquettes personnalisées Amazon Rekognition en utilisant la console ou en utilisant l'opération. [StartProjectVersion](#)

Important

Vous êtes facturé en fonction du nombre d'heures d'exécution de votre modèle et du nombre d'unités d'inférence qu'il utilise pendant son exécution. Pour plus d'informations, consultez [Exécution d'un modèle Étiquettes personnalisées Amazon Rekognition entraîné](#).

Le démarrage d'un modèle peut prendre quelques minutes. Pour vérifier l'état actuel de préparation du modèle, consultez la page de détails du projet ou de l'utilisation [DescribeProjectVersions](#).

Une fois le modèle démarré, vous utilisez [DetectCustomLabels](#), pour analyser les images à l'aide du modèle. Pour plus d'informations, consultez [Analyse d'une image avec un modèle entraîné](#). La console fournit également un exemple de code pour appeler `DetectCustomLabels`.

Rubriques

- [Démarrage d'un modèle Étiquettes personnalisées Amazon Rekognition \(Console\)](#)
- [Démarrage d'un modèle Étiquettes personnalisées Amazon Rekognition \(kit SDK\)](#)

Démarrage d'un modèle Étiquettes personnalisées Amazon Rekognition (Console)

Utilisez la procédure suivante pour commencer à exécuter un modèle Étiquettes personnalisées Amazon Rekognition avec la console. Vous pouvez démarrer le modèle directement depuis la console ou utiliser le code des kits AWS SDK fourni par la console.

Pour démarrer un modèle (console)

1. Ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
2. Choisissez Utiliser Custom Labels.
3. Choisissez Démarrer.
4. Dans le volet de navigation de gauche, choisissez Projets.
5. Sur la page des ressources Projets, choisissez le projet qui contient le modèle entraîné que vous souhaitez démarrer.
6. Dans la section Modèles, choisissez le modèle que vous souhaitez démarrer.
7. Choisissez l'onglet Utiliser le modèle.
8. Effectuez l'une des actions suivantes :

Start model using the console

Dans la section Démarrer ou arrêter le modèle, procédez comme suit :

1. Sélectionnez le nombre d'unités d'inférence que vous souhaitez utiliser. Pour plus d'informations, consultez [Exécution d'un modèle Étiquettes personnalisées Amazon Rekognition entraîné](#).
2. Sélectionnez Démarrer.
3. Dans la boîte de dialogue Démarrer le modèle, choisissez Démarrer.

Start model using the AWS SDK

Dans la section Utiliser votre modèle, procédez comme suit :

1. Choisissez Code de l'API.
2. Choisissez AWS CLI ou Python.

3. Dans Démarrer le modèle, copiez l'exemple de code.
4. Utilisez l'exemple de code pour démarrer votre modèle. Pour plus d'informations, consultez [Démarrage d'un modèle Étiquettes personnalisées Amazon Rekognition \(kit SDK\)](#).
9. Pour revenir à la page de présentation du projet, choisissez le nom de votre projet en haut de la page.
10. Dans la section Modèle, vérifiez le statut du modèle. Lorsque le statut du modèle est EN COURS D'EXÉCUTION, vous pouvez utiliser le modèle pour analyser des images. Pour plus d'informations, consultez [Analyse d'une image avec un modèle entraîné](#).

Démarrage d'un modèle Étiquettes personnalisées Amazon Rekognition (kit SDK)

Vous démarrez un modèle en appelant l'[StartProjectVersion](#) API et en transmettant le nom de ressource Amazon (ARN) du modèle dans le paramètre `ProjectVersionArn` d'entrée. Vous spécifiez également le nombre d'unités d'inférence que vous souhaitez utiliser. Pour plus d'informations, consultez [Exécution d'un modèle Étiquettes personnalisées Amazon Rekognition entraîné](#).

Le démarrage d'un modèle peut prendre un certain temps. Les exemples Python et Java présentés dans cette rubrique utilisent des programmes d'attente pour attendre le démarrage du modèle. Un programme d'attente est un utilitaire qui attend qu'un statut particulier survienne. Vous pouvez également vérifier l'état actuel en appelant [DescribeProjectVersions](#).

Pour démarrer un modèle (kit SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez l'AWS CLI et les kits AWS SDK. Pour plus d'informations, consultez [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez l'exemple de code suivant pour démarrer un modèle.

CLI

Remplacez la valeur de `project-version-arn` par l'ARN du modèle que vous souhaitez démarrer. Remplacez la valeur de `--min-inference-units` par le nombre d'unités d'inférence que vous souhaitez utiliser. (Facultatif) Modifiez `--max-inference-units` par le nombre maximum d'unités d'inférence que la fonctionnalité Étiquettes personnalisées Amazon Rekognition peut utiliser pour automatiquement mettre à l'échelle le modèle.

```
aws rekognition start-project-version --project-version-arn model_arn \  
--min-inference-units minimum number of units \  
--max-inference-units maximum number of units \  
--profile custom-labels-access
```

Python

Fournissez les paramètres de ligne de commande suivants :

- `project_arn` : l'ARN du projet contenant le modèle que vous souhaitez démarrer.
- `model_arn` : l'ARN du modèle que vous souhaitez démarrer.
- `min_inference_units` : le nombre d'unités d'inférence que vous souhaitez utiliser.
- (Facultatif) `--max_inference_units` : le nombre maximum d'unités d'inférence que la fonctionnalité Étiquettes personnalisées Amazon Rekognition peut utiliser pour automatiquement mettre à l'échelle le modèle.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Shows how to start running an Amazon Lookout for Vision model.  
"""  
  
import argparse  
import logging  
import boto3  
from botocore.exceptions import ClientError  
  
logger = logging.getLogger(__name__)  
  
def get_model_status(rek_client, project_arn, model_arn):  
    """  
    Gets the current status of an Amazon Rekognition Custom Labels model  
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.  
    :param project_name: The name of the project that you want to use.  
    :param model_arn: The name of the model that you want the status for.  
    """
```

```
:return: The model status
"""

logger.info("Getting status for %s.", model_arn)

# Extract the model version from the model arn.
version_name = (model_arn.split("version/", 1)[1]).rpartition('/')[0]

models = rek_client.describe_project_versions(ProjectArn=project_arn,
                                             VersionNames=[version_name])

for model in models['ProjectVersionDescriptions']:

    logger.info("Status: %s", model['StatusMessage'])
    return model["Status"]

error_message = f"Model {model_arn} not found."
logger.exception(error_message)
raise Exception(error_message)

def start_model(rek_client, project_arn, model_arn, min_inference_units,
               max_inference_units=None):
    """
    Starts the hosting of an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: The name of the project that contains the
    model that you want to start hosting.
    :param min_inference_units: The number of inference units to use for
    hosting.
    :param max_inference_units: The number of inference units to use for auto-
    scaling
    the model. If not supplied, auto-scaling does not happen.
    """

    try:
        # Start the model
        logger.info(f"Starting model: {model_arn}. Please wait....")

        if max_inference_units is None:
            rek_client.start_project_version(ProjectVersionArn=model_arn,
            MinInferenceUnits=int(min_inference_units))
        else:
```

```
        rek_client.start_project_version(ProjectVersionArn=model_arn,
                                        MinInferenceUnits=int(
                                            min_inference_units),
                                        MaxInferenceUnits=int(max_inference_units))

    # Wait for the model to be in the running state
    version_name = (model_arn.split("version/", 1)[1]).rpartition('/')[0]
    project_version_running_waiter = rek_client.get_waiter(
        'project_version_running')
    project_version_running_waiter.wait(
        ProjectArn=project_arn, VersionNames=[version_name])

    # Get the running status
    return get_model_status(rek_client, project_arn, model_arn)

except ClientError as err:
    logger.exception("Client error: Problem starting model: %s", err)
    raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains that the model
you want to start."
    )
    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to start."
    )
    parser.add_argument(
        "min_inference_units", help="The minimum number of inference units to
use."
    )
    parser.add_argument(
        "--max_inference_units", help="The maximum number of inference units to
use for auto-scaling the model.", required=False
    )
```

```
def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Start the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        status = start_model(rekognition_client,
                             args.project_arn, args.model_arn,
                             args.min_inference_units,
                             args.max_inference_units)

        print(f"Finished starting model: {args.model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        error_message = f"Client error: Problem starting model: {err}"
        logger.exception(error_message)
        print(error_message)

    except Exception as err:
        error_message = f"Problem starting model:{err}"
        logger.exception(error_message)
        print(error_message)

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

Java V2

Fournissez les paramètres de ligne de commande suivants :

- `project_arn` : l'ARN du projet contenant le modèle que vous souhaitez démarrer.

- `model_arn` : l'ARN du modèle que vous souhaitez démarrer.
- `min_inference_units` : le nombre d'unités d'inférence que vous souhaitez utiliser.
- (Facultatif) `max_inference_units` : le nombre maximum d'unités d'inférence que la fonctionnalité Étiquettes personnalisées Amazon Rekognition peut utiliser pour automatiquement mettre à l'échelle le modèle. Si vous ne spécifiez pas de valeur, la mise à l'échelle automatique ne se produit pas.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import
    software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.ProjectVersionStatus;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.StartProjectVersionRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartProjectVersionResponse;
import software.amazon.awssdk.services.rekognition.waiters.RekognitionWaiter;

import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;

public class StartModel {

    public static final Logger logger =
        Logger.getLogger(StartModel.class.getName());
```



```
public static int findForwardSlash(String modelArn, int n) {

    int start = modelArn.indexOf('/');
    while (start >= 0 && n > 1) {
        start = modelArn.indexOf('/', start + 1);
        n -= 1;
    }
    return start;

}

public static void startMyModel(RekognitionClient rekClient, String
projectArn, String modelArn,
    Integer minInferenceUnits, Integer maxInferenceUnits
    ) throws Exception, RekognitionException {

    try {

        logger.log(Level.INFO, "Starting model: {0}", modelArn);

        StartProjectVersionRequest startProjectVersionRequest = null;

        if (maxInferenceUnits == null) {
            startProjectVersionRequest =
StartProjectVersionRequest.builder()
                .projectVersionArn(modelArn)
                .minInferenceUnits(minInferenceUnits)
                .build();
        }
        else {
            startProjectVersionRequest =
StartProjectVersionRequest.builder()
                .projectVersionArn(modelArn)
                .minInferenceUnits(minInferenceUnits)
                .maxInferenceUnits(maxInferenceUnits)
                .build();
        }

        StartProjectVersionResponse response =
rekClient.startProjectVersion(startProjectVersionRequest);

        logger.log(Level.INFO, "Status: {0}", response.statusAsString() );
    }
}
```

```
// Get the model version

int start = findForwardSlash(modelArn, 3) + 1;
int end = findForwardSlash(modelArn, 4);

String versionName = modelArn.substring(start, end);

// wait until model starts

DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
    .versionNames(versionName)
    .projectArn(projectArn)
    .build();

RekognitionWaiter waiter = rekClient.waiter();

WaiterResponse<DescribeProjectVersionsResponse> waiterResponse =
waiter

.waitUntilProjectVersionRunning(describeProjectVersionsRequest);

Optional<DescribeProjectVersionsResponse> optionalResponse =
waiterResponse.matched().response();

DescribeProjectVersionsResponse describeProjectVersionsResponse =
optionalResponse.get();

for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
    .projectVersionDescriptions()) {
    if(projectVersionDescription.status() ==
ProjectVersionStatus.RUNNING) {
        logger.log(Level.INFO, "Model is running" );
    }
    else {
        String error = "Model training failed: " +
projectVersionDescription.statusAsString() + " "
            + projectVersionDescription.statusMessage() + " " +
modelArn;
```

```
        logger.log(Level.SEVERE, error);
        throw new Exception(error);
    }

}

} catch (RekognitionException e) {
    logger.log(Level.SEVERE, "Could not start model: {0}",
e.getMessage());
    throw e;
}

}

public static void main(String[] args) {

    String modelArn = null;
    String projectArn = null;
    Integer minInferenceUnits = null;
    Integer maxInferenceUnits = null;

    final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>
<min_inference_units> <max_inference_units>\n\n" + "Where:\n"
        + "    project_arn - The ARN of the project that contains the
model that you want to start. \n\n"
        + "    model_arn - The ARN of the model version that you want to
start.\n\n"
        + "    min_inference_units - The number of inference units to
start the model with.\n\n"
        + "    max_inference_units - The maximum number of inference
units that Custom Labels can use to "
        + "    automatically scale the model. If the value is null,
automatic scaling doesn't happen.\n\n";

    if (args.length < 3 || args.length >4) {
        System.out.println(USAGE);
        System.exit(1);
    }

    projectArn = args[0];
```

```
modelArn = args[1];
minInferenceUnits=Integer.parseInt(args[2]);

if (args.length == 4) {
    maxInferenceUnits = Integer.parseInt(args[3]);
}

try {

    // Get the Rekognition client.
    RekognitionClient rekClient = RekognitionClient.builder()
        .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
        .region(Region.US_WEST_2)
        .build();

    // Start the model.
    startMyModel(rekClient, projectArn, modelArn, minInferenceUnits,
maxInferenceUnits);

    System.out.println(String.format("Model started: %s", modelArn));

    rekClient.close();

} catch (RekognitionException rekError) {
    logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
    System.exit(1);
} catch (Exception rekError) {
    logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
    System.exit(1);
}

}

}
```

Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition

Vous pouvez arrêter d'exécuter un modèle d'étiquettes personnalisées Amazon Rekognition en utilisant la console ou en utilisant l'opération. [StopProjectVersion](#)

Rubriques

- [Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition \(console\)](#)
- [Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition \(kit SDK\)](#)

Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition (console)

Utilisez la procédure suivante pour arrêter d'exécuter un modèle Étiquettes personnalisées Amazon Rekognition avec la console. Vous pouvez arrêter le modèle directement depuis la console ou utiliser le code des kits AWS SDK fourni par la console.

Pour arrêter un modèle (console)

1. Ouvrez la console Amazon Rekognition à l'adresse <https://console.aws.amazon.com/rekognition/>.
2. Choisissez Utiliser Custom Labels.
3. Choisissez Démarrer.
4. Dans le volet de navigation de gauche, choisissez Projets.
5. Sur la page des ressources Projets, choisissez le projet qui contient le modèle entraîné que vous souhaitez arrêter.
6. Dans la section Modèles, choisissez le modèle que vous souhaitez arrêter.
7. Choisissez l'onglet Utiliser le modèle.
8. Stop model using the console
 1. Dans la section Démarrer ou arrêter le modèle, choisissez Arrêter.
 2. Dans la boîte de dialogue Arrêter le modèle, entrez stop pour confirmer que vous souhaitez arrêter le modèle.
 3. Choisissez Arrêter pour arrêter votre modèle.

Stop model using the AWS SDK

Dans la section Utiliser votre modèle, procédez comme suit :

1. Choisissez Code de l'API.
2. Choisissez AWS CLI ou Python.
3. Dans Arrêter le modèle, copiez l'exemple de code.
4. Utilisez l'exemple de code pour arrêter votre modèle. Pour plus d'informations, consultez [Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition \(kit SDK\)](#).
9. Choisissez le nom de votre projet en haut de la page pour revenir à la page de présentation du projet.
10. Dans la section Modèle, vérifiez le statut du modèle. Le modèle s'arrête lorsque le statut du modèle est ARRÊTÉ.

Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition (kit SDK)

Vous arrêtez un modèle en appelant l'[StopProjectVersion](#) API et en transmettant le nom de ressource Amazon (ARN) du modèle dans le paramètre `ProjectVersionArn` d'entrée.

L'arrêt d'un modèle peut prendre un certain temps. Pour vérifier le statut actuel, utilisez `DescribeProjectVersions`.

Pour arrêter un modèle (kit SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez l'AWS CLI et les kits AWS SDK. Pour plus d'informations, consultez [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez l'exemple de code suivant pour arrêter un modèle en cours d'exécution.

CLI

Remplacez la valeur de `project-version-arn` par l'ARN de la version du modèle que vous souhaitez arrêter.

```
aws rekognition stop-project-version --project-version-arn "model arn" \  
--profile custom-labels-access
```

Python

L'exemple suivant arrête un modèle déjà en cours d'exécution.

Fournissez les paramètres de ligne de commande suivants :

- `project_arn` : l'ARN du projet contenant le modèle que vous souhaitez arrêter.
- `model_arn` : l'ARN du modèle que vous souhaitez arrêter.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to stop a running Amazon Lookout for Vision model.
"""

import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def get_model_status(rek_client, project_arn, model_arn):
    """
    Gets the current status of an Amazon Rekognition Custom Labels model
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: The name of the project that you want to use.
    :param model_arn: The name of the model that you want the status for.
    """

    logger.info ("Getting status for %s.", model_arn)

    # Extract the model version from the model arn.
    version_name=(model_arn.split("version/",1)[1]).rpartition('/')[0]
```

```
# Get the model status.
models=rek_client.describe_project_versions(ProjectArn=project_arn,
VersionNames=[version_name])

for model in models['ProjectVersionDescriptions']:
    logger.info("Status: %s",model['StatusMessage'])
    return model["Status"]

# No model found.
logger.exception("Model %s not found.", model_arn)
raise Exception("Model %s not found.", model_arn)

def stop_model(rek_client, project_arn, model_arn):
    """
    Stops a running Amazon Rekognition Custom Labels Model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that you want to stop running.
    :param model_arn: The ARN of the model (ProjectVersion) that you want to
    stop running.
    """

    logger.info("Stopping model: %s", model_arn)

    try:
        # Stop the model.
        response=rek_client.stop_project_version(ProjectVersionArn=model_arn)

        logger.info("Status: %s", response['Status'])

        # stops when hosting has stopped or failure.
        status = ""
        finished = False

        while finished is False:

            status=get_model_status(rek_client, project_arn, model_arn)

            if status == "STOPPING":
                logger.info("Model stopping in progress...")
                time.sleep(10)
                continue
            if status == "STOPPED":
                logger.info("Model is not running.")
```



```
        finished = True
        continue

        error_message = f"Error stopping model. Unexpected state: {status}"
        logger.exception(error_message)
        raise Exception(error_message)

    logger.info("finished. Status %s", status)
    return status

except ClientError as err:
    logger.exception("Couldn't stop model - %s: %s",
                    model_arn, err.response['Error']['Message'])
    raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains the model that
you want to stop."
    )
    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to stop."
    )

def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Stop the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")
```

```
        status=stop_model(rekognition_client, args.project_arn, args.model_arn)

        print(f"Finished stopping model: {args.model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        logger.exception("Problem stopping model:%s",err)
        print(f"Failed to stop model: {err}")

    except Exception as err:
        logger.exception("Problem stopping model:%s", err)
        print(f"Failed to stop model: {err}")

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

Java V2

Fournissez les paramètres de ligne de commande suivants :

- `project_arn` : l'ARN du projet contenant le modèle que vous souhaitez arrêter.
- `model_arn` : l'ARN du modèle que vous souhaitez arrêter.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

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.DescribeProjectVersionsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import
    software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.ProjectVersionStatus;
```

```
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.StopProjectVersionRequest;
import
    software.amazon.awssdk.services.rekognition.model.StopProjectVersionResponse;

import java.util.logging.Level;
import java.util.logging.Logger;

public class StopModel {

    public static final Logger logger =
        Logger.getLogger(StopModel.class.getName());

    public static int findForwardSlash(String modelArn, int n) {

        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
        return start;
    }

    public static void stopMyModel(RekognitionClient rekClient, String
projectArn, String modelArn)
        throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Stopping {0}", modelArn);

            StopProjectVersionRequest stopProjectVersionRequest =
                StopProjectVersionRequest.builder()
                    .projectVersionArn(modelArn).build();

            StopProjectVersionResponse response =
                rekClient.stopProjectVersion(stopProjectVersionRequest);

            logger.log(Level.INFO, "Status: {0}", response.statusAsString());

            // Get the model version
```

```
int start = findForwardSlash(modelArn, 3) + 1;
int end = findForwardSlash(modelArn, 4);

String versionName = modelArn.substring(start, end);

// wait until model stops

DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
    .projectArn(projectArn).versionNames(versionName).build();

boolean stopped = false;

// Wait until create finishes

do {

    DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

    .describeProjectVersions(describeProjectVersionsRequest);

    for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
        .projectVersionDescriptions()) {

        ProjectVersionStatus status =
projectVersionDescription.status();

        logger.log(Level.INFO, "stopping model: {0} ", modelArn);

        switch (status) {

            case STOPPED:
                logger.log(Level.INFO, "Model stopped");
                stopped = true;
                break;

            case STOPPING:
                Thread.sleep(5000);
                break;

            case FAILED:
```

```
        String error = "Model stopping failed: " +
projectVersionDescription.statusAsString() + " "
        + projectVersionDescription.statusMessage() + "
" + modelArn;

        logger.log(Level.SEVERE, error);
        throw new Exception(error);

    default:
        String unexpectedError = "Unexpected stopping state: "
+ projectVersionDescription.statusAsString() + "
"
        + projectVersionDescription.statusMessage() + "
" + modelArn;

        logger.log(Level.SEVERE, unexpectedError);
        throw new Exception(unexpectedError);
    }
}

} while (stopped == false);

} catch (RekognitionException e) {
    logger.log(Level.SEVERE, "Could not stop model: {0}",
e.getMessage());
    throw e;
}

}

public static void main(String[] args) {

    String modelArn = null;
    String projectArn = null;

    final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>\n
\n" + "Where:\n"
        + "    project_arn - The ARN of the project that contains the
model that you want to stop. \n\n"
        + "    model_arn - The ARN of the model version that you want to
stop.\n\n";

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

```
    }

    projectArn = args[0];
    modelArn = args[1];

    try {

        // Get the Rekognition client.
        RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

        // Stop model
        stopMyModel(rekClient, projectArn, modelArn);

        System.out.println(String.format("Model stopped: %s", modelArn));

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        System.exit(1);
    } catch (Exception rekError) {
        logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
        System.exit(1);
    }

}

}
```

Rapport sur la durée d'exécution et les unités d'inférence utilisées

Si vous avez entraîné et démarré votre modèle après août 2022, vous pouvez utiliser la CloudWatch métrique `InServiceInferenceUnits` Amazon pour déterminer le nombre d'heures pendant lesquelles un modèle a fonctionné et le nombre d'[unités d'inférence](#) utilisées pendant ces heures.

Note

Si vous ne possédez qu'un seul modèle dans une AWS région, vous pouvez également connaître la durée de fonctionnement du modèle en effectuant le suivi des appels entrants `StartProjectVersion` et entrants `StopProjectVersion` réussis CloudWatch. Cette approche ne fonctionne pas si vous exécutez plusieurs modèles dans la région AWS, car les métriques n'incluent pas d'informations sur le modèle.

Vous pouvez également l'utiliser AWS CloudTrail pour suivre les appels vers `StartProjectVersion` et `StopProjectVersion` (ce qui inclut l'ARN du modèle dans le `requestParameters` champ de l'[historique des événements](#)). CloudTrail les événements sont limités à 90 jours, mais vous pouvez les stocker pendant une période maximale de 7 ans dans un [CloudTrail](#).

La procédure suivante permet de créer des graphiques pour ce qui suit :

- Le nombre d'heures pendant lequel un modèle a été exécuté.
- Le nombre d'unités d'inférence utilisées par le modèle.

Vous pouvez choisir une période remontant jusqu'aux 15 derniers mois. Pour plus d'informations sur la conservation des métriques, consultez [Conservation des métriques](#).

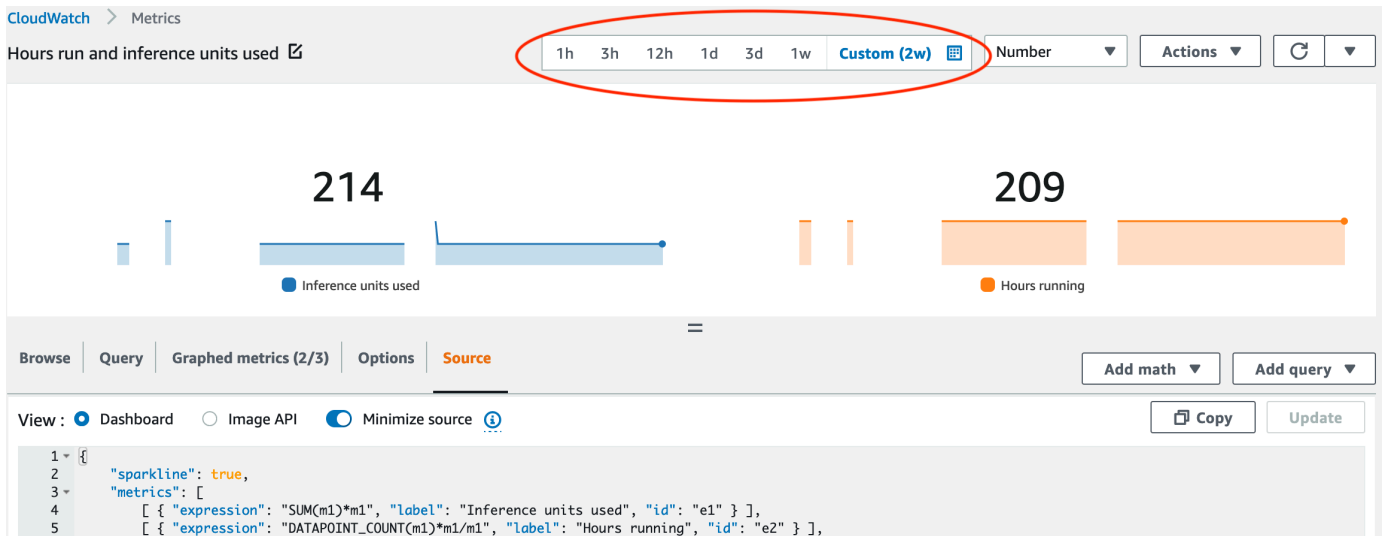
Pour déterminer la durée du modèle et les unités d'inférence utilisées pour un modèle

1. Connectez-vous à la CloudWatch console AWS Management Console et ouvrez-la à l'[adresse https://console.aws.amazon.com/cloudwatch/](https://console.aws.amazon.com/cloudwatch/).
2. Dans le volet de navigation de gauche, choisissez Toutes les métriques sous Métriques.
3. Dans le volet de contenu, choisissez l'onglet Source.
4. Assurez-vous que le bouton Tableau de bord est sélectionné.
5. Dans la zone d'édition, remplacez le fichier JSON existant par le fichier JSON suivant.
Remplacez les valeurs suivantes :
 - `Project_Name` : le projet contenant le modèle que vous souhaitez représenter graphiquement.
 - `Version_Name` : la version du modèle que vous souhaitez représenter graphiquement.

- `AWS_Region` : la région AWS qui contient le modèle. Assurez-vous que la CloudWatch console se trouve dans la même AWS région en cochant le sélecteur de région dans la barre de navigation en haut de la page. Mettez à jour si nécessaire.

```
{
  "sparkline": true,
  "metrics": [
    [
      {
        "expression": "SUM(m1)*m1",
        "label": "Inference units used",
        "id": "e1"
      }
    ],
    [
      {
        "expression": "DATAPoint_COUNT(m1)*m1/m1",
        "label": "Hours running",
        "id": "e2"
      }
    ],
    [
      "AWS/Rekognition",
      "InServiceInferenceUnits",
      "ProjectName",
      "Project_Name",
      "VersionName",
      "Version_Name",
      {
        "id": "m1",
        "visible": false
      }
    ]
  ],
  "view": "singleValue",
  "stacked": false,
  "region": "AWS_Region",
  "stat": "Average",
  "period": 3600,
  "title": "Hours run and inference units used"
}
```


6. Choisissez Mettre à jour.
7. Dans le haut de la page, choisissez une chronologie. Vous devriez voir les chiffres des unités d'inférence utilisées et les heures écoulées au cours de la chronologie. Les lacunes dans le graphique indiquent les moments où le modèle ne s'exécutait pas.



8. (Facultatif) Ajoutez le graphique à un tableau de bord en choisissant Actions, puis Ajouter au tableau de bord - amélioré.

Analyse d'une image avec un modèle entraîné

Pour analyser une image à l'aide d'un modèle d'étiquettes personnalisées Amazon Rekognition entraîné, vous devez appeler l'API. [DetectCustomLabels](#) Le résultat de `DetectCustomLabels` est une prédiction indiquant que l'image contient des objets, des scènes ou des concepts spécifiques.

Pour appeler `DetectCustomLabels`, vous spécifiez ce qui suit :

- L'Amazon Resource Name (ARN) du modèle Étiquettes personnalisées Amazon Rekognition que vous souhaitez utiliser.
- L'image avec laquelle vous souhaitez que le modèle fasse une prédiction. 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. Pour plus d'informations, consultez [Image](#).

Les étiquettes personnalisées sont renvoyées dans un tableau d'objets [Étiquette personnalisée](#). Chaque étiquette personnalisée représente un objet, une scène ou un concept unique présent dans l'image. Une étiquette personnalisée inclut :

- Une étiquette pour l'objet, la scène ou le concept présent dans l'image.
- Un cadre de délimitation pour les objets trouvés dans l'image. Les coordonnées du cadre de délimitation montrent où l'objet est situé sur l'image source. Les valeurs des coordonnées sont un ratio de la taille globale de l'image. Pour plus d'informations, consultez [BoundingBox](#). `DetectCustomLabels` renvoie des cadres de délimitation uniquement si le modèle est entraîné à détecter l'emplacement des objets.
- La confiance de la fonctionnalité Étiquettes personnalisées Amazon Rekognition quant à la précision de l'étiquette et du cadre de délimitation.

Pour filtrer les étiquettes en fonction du niveau de confiance de détection, spécifiez une valeur pour `MinConfidence` correspondant au niveau de confiance souhaité. Par exemple, si vous devez être très sûr de la prédiction, spécifiez une valeur élevée pour `MinConfidence`. Pour obtenir toutes les étiquettes, quel que soit le niveau de confiance, spécifiez une valeur `MinConfidence` de 0.

Les performances de votre modèle sont mesurées, en partie, par les métriques de rappel et de précision calculées lors de l'entraînement du modèle. Pour plus d'informations, consultez [Métriques pour évaluer votre modèle](#).

Pour augmenter la précision de votre modèle, définissez une valeur plus élevée pour `MinConfidence`. Pour plus d'informations, consultez [Réduire les faux positifs \(meilleure précision\)](#).

Pour augmenter le rappel de votre modèle, utilisez une valeur inférieure pour `MinConfidence`. Pour plus d'informations, consultez [Réduire les faux négatifs \(meilleure mémorisation\)](#).

Si vous ne spécifiez aucune valeur pour `MinConfidence`, la fonctionnalité Étiquettes personnalisées Amazon Rekognition renvoie une étiquette basée sur le seuil supposé pour cette étiquette. Pour plus d'informations, consultez [Seuil supposé](#). Vous pouvez obtenir la valeur du seuil supposé d'une étiquette à partir des résultats d'entraînement du modèle. Pour plus d'informations, consultez [Entraînement d'un modèle \(console\)](#).

En utilisant le paramètre d'entrée `MinConfidence`, vous spécifiez le seuil souhaité pour l'appel. Les étiquettes détectées avec un niveau de confiance inférieur à la valeur de `MinConfidence` ne sont pas renvoyées dans la réponse. De plus, le seuil supposé pour une étiquette n'affecte pas l'inclusion de l'étiquette dans la réponse.

Note

Les métriques Étiquettes personnalisées Amazon Rekognition expriment un seuil supposé sous la forme d'une valeur à virgule flottante comprise entre 0 et 1. La plage de `MinConfidence` normalise le seuil à une valeur en pourcentage (0-100). Les réponses de confiance de `DetectCustomLabels` sont également renvoyées sous forme de pourcentage.

Vous pouvez vouloir spécifier un seuil pour des étiquettes spécifiques. Par exemple, lorsque la métrique de précision est acceptable pour l'étiquette A, mais pas pour l'étiquette B. Lorsque vous spécifiez un seuil différent (`MinConfidence`), tenez compte des points suivants.

- Si vous n'êtes intéressé que par une seule étiquette (A), définissez la valeur de `MinConfidence` sur le seuil souhaité. Dans la réponse, les prédictions pour l'étiquette A sont renvoyées (ainsi que pour les autres étiquettes) uniquement si le niveau de confiance est supérieur à `MinConfidence`. Vous devez filtrer toutes les autres étiquettes renvoyées.
- Si vous souhaitez appliquer différents seuils à plusieurs étiquettes, procédez comme suit :
 1. Utilisez une valeur 0 pour `MinConfidence`. La valeur 0 garantit que toutes les étiquettes sont renvoyées, quel que soit le niveau de confiance de détection.
 2. Pour chaque étiquette renvoyée, appliquez le seuil souhaité en vérifiant que le niveau de confiance de l'étiquette est supérieur au seuil que vous souhaitez pour l'étiquette.

Pour plus d'informations, consultez [Amélioration d'un modèle d'étiquettes personnalisées Amazon Rekognition certifié](#).

Si vous trouvez que les valeurs de confiance renvoyées par `DetectCustomLabels` sont trop faibles, pensez à entraîner à nouveau le modèle. Pour plus d'informations, consultez [Entraînement d'un modèle Amazon Rekognition Custom Labels Custom Labels](#). Vous pouvez limiter le nombre d'étiquettes personnalisées renvoyées par `DetectCustomLabels` en spécifiant le paramètre d'entrée `MaxResults`. Les résultats sont renvoyés triés du niveau de confiance le plus élevé au niveau de confiance le plus faible.

Pour d'autres exemples appelant `DetectCustomLabels`, consultez [Exemples](#).

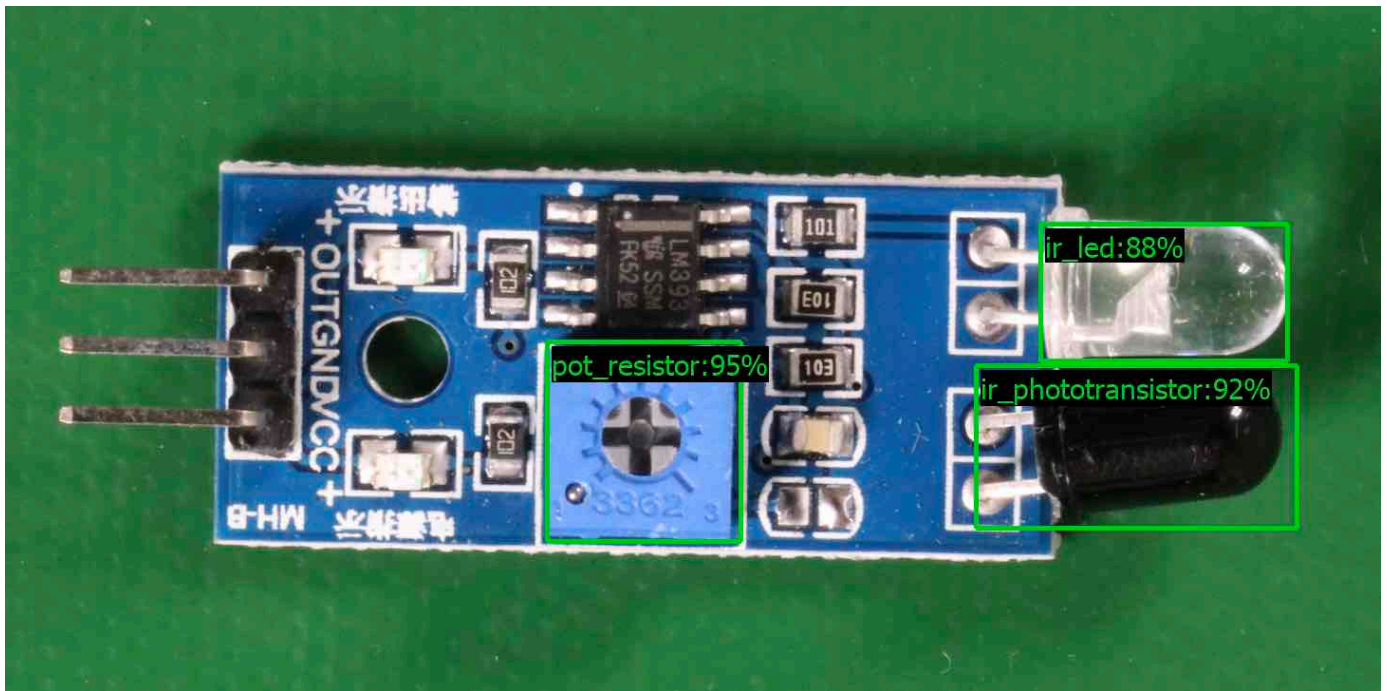
Pour plus d'informations sur la sécurisation de `DetectCustomLabels`, consultez [SécurisationDetectCustomLabels](#).

Pour détecter des étiquettes personnalisées (API)

1. Si vous ne l'avez pas déjà fait :
 - a. Veillez à disposer des autorisations `DetectCustomLabels` et `AmazonS3ReadOnlyAccess`. Pour plus d'informations, consultez [Configuration des autorisations du kit SDK](#).
 - b. Installez et configurez AWS CLI les AWS SDK. Pour plus d'informations, consultez [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Entraînez et déployez votre modèle. Pour plus d'informations, consultez [Création d'un modèle d'étiquettes personnalisées Amazon Rekognition](#).
3. Veillez à ce que l'utilisateur qui appelle `DetectCustomLabels` a accès au modèle que vous avez utilisé à l'étape 2. Pour plus d'informations, consultez [SécurisationDetectCustomLabels](#).
4. Chargez une image que vous souhaitez analyser dans un compartiment S3.

Pour en savoir plus, consultez [Chargement d'objets dans Amazon S3](#) dans le Guide de l'utilisateur Amazon Simple Storage Service. Les exemples Python, Java et Java 2 vous montrent également comment utiliser un fichier image local pour transmettre une image en utilisant des octets bruts. La taille du fichier doit être inférieure à 4 Mo.

5. Utilisez les exemples suivants pour appeler l'opération `DetectCustomLabels`. Les exemples Python et Java montrent l'image et superposent les résultats de l'analyse, comme dans l'image suivante.



AWS CLI

Cette AWS CLI commande affiche la sortie JSON pour l'opération DetectCustomLabels CLI. Modifiez les valeurs des paramètres d'entrée suivants.

- `bucket` avec le nom du compartiment Amazon S3 que vous avez utilisé à l'étape 4.
- `image` avec le nom du fichier image d'entrée que vous avez chargé à l'étape 4.
- `projectVersionArn` avec l'ARN du modèle que vous souhaitez utiliser.

```
aws rekognition detect-custom-labels --project-version-arn model_arn \  
  --image '{"S3Object":{"Bucket":"bucket","Name":"image"}}' \  
  --min-confidence 70 \  
  --profile custom-labels-access
```

Python

L'exemple de code suivant affiche les cadres de délimitation et les étiquettes de niveau d'image trouvés dans une image.

Pour analyser une image locale, exécutez le programme et fournissez les arguments de ligne de commande suivants :

- L'ARN du modèle avec lequel vous souhaitez analyser l'image.
- Le nom et l'emplacement d'un fichier image local.

Pour analyser une image stockée dans un compartiment Amazon S3, exécutez le programme et fournissez les arguments de ligne de commande suivants :

- L'ARN du modèle avec lequel vous souhaitez analyser l'image.
- Le nom et l'emplacement d'une image dans le compartiment Amazon S3 que vous avez utilisée à l'étape 4.
- `--bucket nom du compartiment` : le compartiment Amazon S3 que vous avez utilisé à l'étape 4.

Notez que cet exemple suppose que votre version de Pillow est $\geq 8.0.0$.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Amazon Rekognition Custom Labels detection example used in the service
documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/detecting-custom-
labels.html
Shows how to detect custom labels by using an Amazon Rekognition Custom Labels
model.
The image can be stored on your local computer or in an Amazon S3 bucket.
"""

import io
import logging
import argparse
import boto3
from PIL import Image, ImageDraw, ImageFont

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def analyze_local_image(rek_client, model, photo, min_confidence):
```

```
"""
Analyzes an image stored as a local file.
:param rek_client: The Amazon Rekognition Boto3 client.
:param s3_connection: The Amazon S3 Boto3 S3 connection object.
:param model: The ARN of the Amazon Rekognition Custom Labels model that you
want to use.
:param photo: The name and file path of the photo that you want to analyze.
:param min_confidence: The desired threshold/confidence for the call.
"""

try:
    logger.info("Analyzing local file: %s", photo)
    image = Image.open(photo)
    image_type = Image.MIME[image.format]

    if (image_type == "image/jpeg" or image_type == "image/png") is False:
        logger.error("Invalid image type for %s", photo)
        raise ValueError(
            f"Invalid file format. Supply a jpeg or png format file:
{photo}")
        )

    # get images bytes for call to detect_anomalies
    image_bytes = io.BytesIO()
    image.save(image_bytes, format=image.format)
    image_bytes = image_bytes.getvalue()

    response = rek_client.detect_custom_labels(Image={'Bytes': image_bytes},
                                                MinConfidence=min_confidence,
                                                ProjectVersionArn=model)

    show_image(image, response)
    return len(response['CustomLabels'])

except ClientError as client_err:
    logger.error(format(client_err))
    raise
except FileNotFoundError as file_error:
    logger.error(format(file_error))
    raise

def analyze_s3_image(rek_client, s3_connection, model, bucket, photo,
                    min_confidence):
```

```
"""
Analyzes an image stored in the specified S3 bucket.
:param rek_client: The Amazon Rekognition Boto3 client.
:param s3_connection: The Amazon S3 Boto3 S3 connection object.
:param model: The ARN of the Amazon Rekognition Custom Labels model that you
want to use.
:param bucket: The name of the S3 bucket that contains the image that you
want to analyze.
:param photo: The name of the photo that you want to analyze.
:param min_confidence: The desired threshold/confidence for the call.
"""

try:
    # Get image from S3 bucket.

    logger.info("analyzing bucket: %s image: %s", bucket, photo)
    s3_object = s3_connection.Object(bucket, photo)
    s3_response = s3_object.get()

    stream = io.BytesIO(s3_response['Body'].read())
    image = Image.open(stream)

    image_type = Image.MIME[image.format]

    if (image_type == "image/jpeg" or image_type == "image/png") is False:
        logger.error("Invalid image type for %s", photo)
        raise ValueError(
            f"Invalid file format. Supply a jpeg or png format file:
{photo}")

    ImageDraw.Draw(image)

    # Call DetectCustomLabels.
    response = rek_client.detect_custom_labels(
        Image={'S3Object': {'Bucket': bucket, 'Name': photo}},
        MinConfidence=min_confidence,
        ProjectVersionArn=model)

    show_image(image, response)
    return len(response['CustomLabels'])

except ClientError as err:
    logger.error(format(err))
    raise
```



```
def show_image(image, response):
    """
    Displays the analyzed image and overlays analysis results
    :param image: The analyzed image
    :param response: the response from DetectCustomLabels
    """
    try:
        font_size = 40
        line_width = 5

        img_width, img_height = image.size
        draw = ImageDraw.Draw(image)

        # Calculate and display bounding boxes for each detected custom label.
        image_level_label_height = 0

        for custom_label in response['CustomLabels']:
            confidence = int(round(custom_label['Confidence'], 0))
            label_text = f"{custom_label['Name']}:{confidence}%"
            fnt = ImageFont.truetype('Tahoma.ttf', font_size)
            text_left, text_top, text_right, text_bottom = draw.textbbox((0, 0),
label_text, fnt)
            text_width, text_height = text_right - text_left, text_bottom -
text_top

            logger.info("Label: %s", custom_label['Name'])
            logger.info("Confidence: %s", confidence)

            # Draw bounding boxes, if present
            if 'Geometry' in custom_label:
                box = custom_label['Geometry']['BoundingBox']
                left = img_width * box['Left']
                top = img_height * box['Top']
                width = img_width * box['Width']
                height = img_height * box['Height']

                logger.info("Bounding box")
                logger.info("\tLeft: {0:.0f}".format(left))
                logger.info("\tTop: {0:.0f}".format(top))
                logger.info("\tLabel Width: {0:.0f}".format(width))
                logger.info("\tLabel Height: {0:.0f}".format(height))
```

```
        points = (
            (left, top),
            (left + width, top),
            (left + width, top + height),
            (left, top + height),
            (left, top))
        # Draw bounding box and label text
        draw.line(points, fill="limegreen", width=line_width)
        draw.rectangle([(left + line_width, top+line_width),
            (left + text_width + line_width, top +
line_width + text_height)], fill="black")
        draw.text((left + line_width, top + line_width),
            label_text, fill="limegreen", font=fnt)

        # draw image-level label text.
    else:
        draw.rectangle([(10, image_level_label_height),
            (text_width + 10, image_level_label_height
+text_height)], fill="black")
        draw.text((10, image_level_label_height),
            label_text, fill="limegreen", font=fnt)

        image_level_label_height += text_height

    image.show()

except Exception as err:
    logger.error(format(err))
    raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to use."
    )

    parser.add_argument(
        "image", help="The path and file name of the image that you want to
analyze"
```

```
)
parser.add_argument(
    "--bucket", help="The bucket that contains the image. If not supplied,
image is assumed to be a local file.", required=False
)

def main():

    try:
        logging.basicConfig(level=logging.INFO,
                            format="%(levelname)s: %(message)s")

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        label_count = 0
        min_confidence = 50

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        if args.bucket is None:
            # Analyze local image.
            label_count = analyze_local_image(rekognition_client,
                                             args.model_arn,
                                             args.image,
                                             min_confidence)
        else:
            # Analyze image in S3 bucket.
            s3_connection = session.resource('s3')
            label_count = analyze_s3_image(rekognition_client,
                                          s3_connection,
                                          args.model_arn,
                                          args.bucket,
                                          args.image,
                                          min_confidence)

        print(f"Custom labels detected: {label_count}")

    except ClientError as client_err:
        print("A service client error occurred: " +
```

```
        format(client_err.response["Error"]["Message"]))

    except ValueError as value_err:
        print("A value error occurred: " + format(value_err))

    except FileNotFoundError as file_error:
        print("File not found error: " + format(file_error))

    except Exception as err:
        print("An error occurred: " + format(err))

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

Java

L'exemple de code suivant affiche les cadres de délimitation et les étiquettes de niveau d'image trouvés dans une image.

Pour analyser une image locale, exécutez le programme et fournissez les arguments de ligne de commande suivants :

- L'ARN du modèle avec lequel vous souhaitez analyser l'image.
- Le nom et l'emplacement d'un fichier image local.

Pour analyser une image stockée dans un compartiment Amazon S3, exécutez le programme et fournissez les arguments de ligne de commande suivants :

- L'ARN du modèle avec lequel vous souhaitez analyser l'image.
- Le nom et l'emplacement d'une image dans le compartiment Amazon S3 que vous avez utilisée à l'étape 4.
- Le compartiment Amazon S3 contenant l'image que vous avez utilisée à l'étape 4.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
```

```
package com.amazonaws.samples;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import java.io.FileNotFoundException;
import java.awt.font.FontRenderContext;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;

import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
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.CustomLabel;
import com.amazonaws.services.rekognition.model.DetectCustomLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectCustomLabelsResult;
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;

import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.util.IOUtils;

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

    private transient DetectCustomLabelsResult response;
```

```
private transient Dimension dimension;
private transient BufferedImage image;

public static final Logger logger =
Logger.getLogger(DetectCustomLabels.class.getName());

// Finds custom labels in an image stored in an S3 bucket.
public DetectCustomLabels(AmazonRekognition rekClient,
    AmazonS3 s3client,
    String projectVersionArn,
    String bucket,
    String key,
    Float minConfidence) throws AmazonRekognitionException,
AmazonS3Exception, IOException {

    logger.log(Level.INFO, "Processing S3 bucket: {0} image {1}", new
Object[] { bucket, key });

    // Get image from S3 bucket and create BufferedImage
    com.amazonaws.services.s3.model.S3Object s3object =
s3client.getObject(bucket, key);
    S3ObjectInputStream inputStream = s3object.getObjectContent();
    image = ImageIO.read(inputStream);

    // Set image size
    setWindowDimensions();

    DetectCustomLabelsRequest request = new DetectCustomLabelsRequest()
        .withProjectVersionArn(projectVersionArn)
        .withImage(new Image().withS3Object(new
S3Object().withName(key).withBucket(bucket)))
        .withMinConfidence(minConfidence);

    // Call DetectCustomLabels

    response = rekClient.detectCustomLabels(request);
    logFoundLabels(response.getCustomLabels());
    drawLabels();

}

// Finds custom label in a local image file.
public DetectCustomLabels(AmazonRekognition rekClient,
    String projectVersionArn,
```

```
        String photo,
        Float minConfidence)
        throws IOException, AmazonRekognitionException {

    logger.log(Level.INFO, "Processing local file: {0}", photo);

    // Get image bytes and buffered image
    ByteBuffer imageBytes;
    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);

    // Set image size
    setWindowDimensions();

    // Analyze image
    DetectCustomLabelsRequest request = new DetectCustomLabelsRequest()
        .withProjectVersionArn(projectVersionArn)
        .withImage(new Image()
            .withBytes(imageBytes))
        .withMinConfidence(minConfidence);

    response = rekClient.detectCustomLabels(request);

    logFoundLabels(response.getCustomLabels());

    drawLabels();
}

// Log the labels found by DetectCustomLabels
private void logFoundLabels(List<CustomLabel> customLabels) {
    logger.info("Custom labels found");
    if (customLabels.isEmpty()) {
        logger.log(Level.INFO, "No Custom Labels found. Consider lowering
min confidence.");
    }
}
```

```
    } else {
        for (CustomLabel customLabel : customLabels) {
            logger.log(Level.INFO, " Label: {0} Confidence: {1}",
                new Object[] { customLabel.getName(),
customLabel.getConfidence() });
        }
    }
}

// Sets window dimensions to 1/2 screen size, unless image is smaller
public void setWindowDimensions() {
    dimension = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

    dimension.width = (int) dimension.getWidth() / 2;
    if (image.getWidth() < dimension.width) {
        dimension.width = image.getWidth();
    }
    dimension.height = (int) dimension.getHeight() / 2;

    if (image.getHeight() < dimension.height) {
        dimension.height = image.getHeight();
    }

    setPreferredSize(dimension);
}

// Draws the image containing the bounding boxes and labels.
@Override
public void paintComponent(Graphics g) {

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

    // Draw the image.
    g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);
}

public void drawLabels() {
    // Draws bounding boxes (if present) and label text.

    int boundingBoxBorderWidth = 5;
    int imageHeight = image.getHeight(this);
```



```
int imageWidth = image.getWidth(this);

// Set up drawing
Graphics2D g2d = image.createGraphics();
g2d.setColor(Color.GREEN);
g2d.setFont(new Font("Tahoma", Font.PLAIN, 50));
Font font = g2d.getFont();
FontRenderContext frc = g2d.getFontRenderContext();
g2d.setStroke(new BasicStroke(boundingBoxBorderWidth));

List<CustomLabel> customLabels = response.getCustomLabels();

int imageLevelLabelHeight = 0;
for (CustomLabel customLabel : customLabels) {

    String label = customLabel.getName();

    int textWidth = (int) (font.getStringBounds(label, frc).getWidth());
    int textHeight = (int) (font.getStringBounds(label,
frc).getHeight());

    // Draw bounding box, if present
    if (customLabel.getGeometry() != null) {

        BoundingBox box = customLabel.getGeometry().getBoundingBox();
        float left = imageWidth * box.getLeft();
        float top = imageHeight * box.getTop();

        // Draw black rectangle
        g2d.setColor(Color.BLACK);
        g2d.fillRect(Math.round(left + (boundingBoxBorderWidth)),
Math.round(top + (boundingBoxBorderWidth)),
                    textWidth + boundingBoxBorderWidth, textHeight +
boundingBoxBorderWidth);

        // Write label onto black rectangle
        g2d.setColor(Color.GREEN);
        g2d.drawString(label, left + boundingBoxBorderWidth, (top +
textHeight));

        // Draw bounding box around label location
        g2d.drawRect(Math.round(left), Math.round(top),
Math.round((imageWidth * box.getWidth()))),
                    Math.round((imageHeight * box.getHeight())));
```

```
    }
    // Draw image level labels.
    else {
        // Draw black rectangle
        g2d.setColor(Color.BLACK);
        g2d.fillRect(10, 10 + imageLevelLabelHeight, textWidth,
textHeight);

        g2d.setColor(Color.GREEN);
        g2d.drawString(label, 10, textHeight + imageLevelLabelHeight);

        imageLevelLabelHeight += textHeight;
    }

}
g2d.dispose();

}

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

    String photo = null;
    String bucket = null;
    String projectVersionArn = null;
    float minConfidence = 50;

    final String USAGE = "\n" + "Usage: " + "<model_arn> <image> <bucket>\n"
\n" + "Where:\n"
        + "    model_arn - The ARN of the model that you want to use. \n"
\n"
        + "    image - The location of the image on your local file
system or within an S3 bucket.\n\n"
        + "    bucket - The S3 bucket that contains the image. Don't
specify if image is local.\n\n";

    // Collect the arguments. If 3 arguments are present, the image is
assumed to be
    // in an S3 bucket.

    if (args.length < 2 || args.length > 3) {
        System.out.println(USAGE);
        System.exit(1);
    }

    projectVersionArn = args[0];
```

```
    photo = args[1];

    if (args.length == 3) {
        bucket = args[2];
    }

    DetectCustomLabels panel = null;

    try {

        AWSCredentialsProvider provider =new
ProfileCredentialsProvider("custom-labels-access");

        AmazonRekognition rekClient =
AmazonRekognitionClientBuilder.standard()
            .withCredentials(provider)
            .withRegion(Regions.US_WEST_2)
            .build();

        AmazonS3 s3client = AmazonS3ClientBuilder.standard()
            .withCredentials(provider)
            .withRegion(Regions.US_WEST_2)
            .build();

        // Create frame and panel.
        JFrame frame = new JFrame("Custom Labels");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        if (args.length == 2) {
            // Analyze local image
            panel = new DetectCustomLabels(rekClient, projectVersionArn,
photo, minConfidence);
        } else {
            // Analyze image in S3 bucket
            panel = new DetectCustomLabels(rekClient, s3client,
projectVersionArn, bucket, photo, minConfidence);
        }

        frame.setContentPane(panel);
        frame.pack();
        frame.setVisible(true);

    } catch (AmazonRekognitionException rekError) {
```

```
        String errorMessage = "Rekognition client error: " +
rekError.getMessage();
        logger.log(Level.SEVERE, errorMessage);
        System.out.println(errorMessage);
        System.exit(1);
    } catch (FileNotFoundException fileError) {
        String errorMessage = "File not found: " + photo;
        logger.log(Level.SEVERE, errorMessage);
        System.out.println(errorMessage);
        System.exit(1);
    } catch (IOException fileError) {
        String errorMessage = "Input output exception: " +
fileError.getMessage();
        logger.log(Level.SEVERE, errorMessage);
        System.out.println(errorMessage);
        System.exit(1);
    } catch (AmazonS3Exception s3Error) {
        String errorMessage = "S3 error: " + s3Error.getErrorMessage();
        logger.log(Level.SEVERE, errorMessage);
        System.out.println(errorMessage);
        System.exit(1);
    }
}
}
```

Java V2

L'exemple de code suivant affiche les cadres de délimitation et les étiquettes de niveau d'image trouvés dans une image.

Pour analyser une image locale, exécutez le programme et fournissez les arguments de ligne de commande suivants :

- `projectVersionArn` : l'ARN du modèle avec lequel vous souhaitez analyser l'image.
- `photo` : le nom et l'emplacement d'un fichier image local.

Pour analyser une image stockée dans un compartiment S3, exécutez le programme et fournissez les arguments de ligne de commande suivants :

- L'ARN du modèle avec lequel vous souhaitez analyser l'image.

- Le nom et l'emplacement d'une image dans le compartiment S3 que vous avez utilisée à l'étape 4.
- Le compartiment Amazon S3 contenant l'image que vous avez utilisée à l'étape 4.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
 */

package com.example.rekognition;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
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.Image;
import
  software.amazon.awssdk.services.rekognition.model.DetectCustomLabelsRequest;
import
  software.amazon.awssdk.services.rekognition.model.DetectCustomLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.CustomLabel;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;

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.NoSuchBucketException;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;

import java.awt.*;
import java.awt.font.FontRenderContext;
```

```
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import javax.swing.*;

import java.util.logging.Level;
import java.util.logging.Logger;

// Calls DetectCustomLabels on an image. Displays bounding boxes or
// image level labels found in the image.
public class ShowCustomLabels extends JPanel {

    private transient BufferedImage image;
    private transient DetectCustomLabelsResponse response;
    private transient Dimension dimension;
    public static final Logger logger =
        Logger.getLogger(ShowCustomLabels.class.getName());

    // Finds custom labels in an image stored in an S3 bucket.
    public ShowCustomLabels(RekognitionClient rekClient,
        S3Client s3client,
        String projectVersionArn,
        String bucket,
        String key,
        Float minConfidence) throws RekognitionException,
        NoSuchBucketException, NoSuchKeyException, IOException {

        logger.log(Level.INFO, "Processing S3 bucket: {0} image {1}", new
            Object[] { bucket, key });
        // Get image from S3 bucket and create BufferedImage
        GetObjectRequest requestObject =
            GetObjectRequest.builder().bucket(bucket).key(key).build();
        ResponseBytes<GetObjectResponse> result =
            s3client.getObject(requestObject, ResponseTransformer.toBytes());
        ByteArrayInputStream bis = new
            ByteArrayInputStream(result.asByteArray());
        image = ImageIO.read(bis);

        // Set image size
        setWindowDimensions();

        // Construct request parameter for DetectCustomLabels
        S3Object s3Object = S3Object.builder().bucket(bucket).name(key).build();

        Image s3Image = Image.builder().s3Object(s3Object).build();
```

```
        DetectCustomLabelsRequest request =
DetectCustomLabelsRequest.builder().image(s3Image)

        .projectVersionArn(projectVersionArn).minConfidence(minConfidence).build();

        response = rekClient.detectCustomLabels(request);
        logFoundLabels(response.customLabels());
        drawLabels();

    }

    // Finds custom label in a local image file.
    public ShowCustomLabels(RekognitionClient rekClient,
        String projectVersionArn,
        String photo,
        Float minConfidence)
        throws IOException, RekognitionException {

        logger.log(Level.INFO, "Processing local file: {0}", photo);
        // Get image bytes and buffered image
        InputStream sourceStream = new FileInputStream(new File(photo));
        SdkBytes imageBytes = SdkBytes.fromInputStream(sourceStream);
        ByteArrayInputStream inputStream = new
ByteArrayInputStream(imageBytes.asByteArray());
        image = ImageIO.read(inputStream);

        setWindowDimensions();

        // Construct request parameter for DetectCustomLabels
        Image localImageBytes = Image.builder().bytes(imageBytes).build();

        DetectCustomLabelsRequest request =
DetectCustomLabelsRequest.builder().image(localImageBytes)

        .projectVersionArn(projectVersionArn).minConfidence(minConfidence).build();

        response = rekClient.detectCustomLabels(request);

        logFoundLabels(response.customLabels());
        drawLabels();

    }
}
```

```
// Sets window dimensions to 1/2 screen size, unless image is smaller
public void setWindowDimensions() {
    dimension = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

    dimension.width = (int) dimension.getWidth() / 2;
    if (image.getWidth() < dimension.width) {
        dimension.width = image.getWidth();
    }
    dimension.height = (int) dimension.getHeight() / 2;

    if (image.getHeight() < dimension.height) {
        dimension.height = image.getHeight();
    }

    setPreferredSize(dimension);
}

// Draws bounding boxes (if present) and label text.
public void drawLabels() {

    int boundingBoxBorderWidth = 5;
    int imageHeight = image.getHeight(this);
    int imageWidth = image.getWidth(this);

    // Set up drawing
    Graphics2D g2d = image.createGraphics();
    g2d.setColor(Color.GREEN);
    g2d.setFont(new Font("Tahoma", Font.PLAIN, 50));
    Font font = g2d.getFont();
    FontRenderContext frc = g2d.getFontRenderContext();
    g2d.setStroke(new BasicStroke(boundingBoxBorderWidth));

    List<CustomLabel> customLabels = response.customLabels();

    int imageLevelLabelHeight = 0;
    for (CustomLabel customLabel : customLabels) {

        String label = customLabel.name();

        int textWidth = (int) (font.getStringBounds(label, frc).getWidth());
        int textHeight = (int) (font.getStringBounds(label,
frc).getHeight());
```



```
// Draw bounding box, if present
if (customLabel.geometry() != null) {

    BoundingBox box = customLabel.geometry().boundingBox();
    float left = imageWidth * box.left();
    float top = imageHeight * box.top();

    // Draw black rectangle
    g2d.setColor(Color.BLACK);
    g2d.fillRect(Math.round(left + (boundingBoxBorderWidth)),
Math.round(top + (boundingBoxBorderWidth)),
                textWidth + boundingBoxBorderWidth, textHeight +
boundingBoxBorderWidth);

    // Write label onto black rectangle
    g2d.setColor(Color.GREEN);
    g2d.drawString(label, left + boundingBoxBorderWidth, (top +
textHeight));

    // Draw bounding box around label location
    g2d.drawRect(Math.round(left), Math.round(top),
Math.round((imageWidth * box.width())),
                Math.round((imageHeight * box.height())));
}
// Draw image level labels.
else {
    // Draw black rectangle
    g2d.setColor(Color.BLACK);
    g2d.fillRect(10, 10 + imageLevelLabelHeight, textWidth,
textHeight);

    g2d.setColor(Color.GREEN);
    g2d.drawString(label, 10, textHeight + imageLevelLabelHeight);

    imageLevelLabelHeight += textHeight;
}

}
g2d.dispose();

}

// Log the labels found by DetectCustomLabels
private void logFoundLabels(List<CustomLabel> customLabels) {
    logger.info("Custom labels found:");
}
```

```
        if (customLabels.isEmpty()) {
            logger.log(Level.INFO, "No Custom Labels found. Consider lowering
min confidence.");

        }
        else {
            for (CustomLabel customLabel : customLabels) {
                logger.log(Level.INFO, " Label: {0} Confidence: {1}",
                    new Object[] { customLabel.name(),
customLabel.confidence() } );
            }
        }
    }

    // Draws the image containing the bounding boxes and labels.
    @Override
    public void paintComponent(Graphics g) {

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

        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

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

        String photo = null;
        String bucket = null;
        String projectVersionArn = null;

        final String USAGE = "\n" + "Usage: " + "<model_arn> <image> <bucket>\n
\n" + "Where:\n"
            + "    model_arn - The ARN of the model that you want to use. \n
\n"
            + "    image - The location of the image on your local file
system or within an S3 bucket.\n\n"
            + "    bucket - The S3 bucket that contains the image. Don't
specify if image is local.\n\n";

        // Collect the arguments. If 3 arguments are present, the image is
assumed to be
        // in an S3 bucket.
```

```
    if (args.length < 2 || args.length > 3) {
        System.out.println(USAGE);
        System.exit(1);
    }

    projectVersionArn = args[0];
    photo = args[1];

    if (args.length == 3) {
        bucket = args[2];
    }

    float minConfidence = 50;

    ShowCustomLabels panel = null;

    try {
        // Get the Rekognition client

        // Get the Rekognition client.
        RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

        S3Client s3Client = S3Client.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

        // Create frame and panel.
        JFrame frame = new JFrame("Custom Labels");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        if (args.length == 2) {
            // Analyze local image
            panel = new ShowCustomLabels(rekClient, projectVersionArn,
photo, minConfidence);
        } else {
            // Analyze image in S3 bucket
```

```
        panel = new ShowCustomLabels(rekClient, s3Client,
projectVersionArn, bucket, photo, minConfidence);
    }

    frame.setContentPane(panel);
    frame.pack();
    frame.setVisible(true);

} catch (RekognitionException rekError) {

    String errorMessage = "Rekognition client error: " +
rekError.getMessage();
    logger.log(Level.SEVERE, errorMessage);
    System.out.println(errorMessage);
    System.exit(1);
} catch (FileNotFoundException fileError) {
    String errorMessage = "File not found: " + photo;
    logger.log(Level.SEVERE, errorMessage);
    System.out.println(errorMessage);
    System.exit(1);
} catch (IOException fileError) {
    String errorMessage = "Input output exception: " +
fileError.getMessage();
    logger.log(Level.SEVERE, errorMessage);
    System.out.println(errorMessage);
    System.exit(1);
} catch (NoSuchKeyException bucketError) {
    String errorMessage = String.format("Image not found: %s in bucket
%s.", photo, bucket);
    logger.log(Level.SEVERE, errorMessage);
    System.out.println(errorMessage);
    System.exit(1);
} catch (NoSuchBucketException bucketError) {
    String errorMessage = "Bucket not found: " + bucket;
    logger.log(Level.SEVERE, errorMessage);
    System.out.println(errorMessage);
    System.exit(1);
}

}

}
```

DetectCustomLabels demande d'opération

Dans le cadre de l'opération `DetectCustomLabels`, 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.

```
{
  "ProjectVersionArn": "string",
  "Image": {
    "S3Object": {
      "Bucket": "string",
      "Name": "string",
      "Version": "string"
    }
  },
  "MinConfidence": 90,
  "MaxLabels": 10,
}
```

DetectCustomLabels réponse à l'opération

La réponse JSON suivante de l'opération `DetectCustomLabels` montre les étiquettes personnalisées qui ont été détectées dans l'image suivante.

```
{
  "CustomLabels": [
    {
      "Name": "MyLogo",
      "Confidence": 77.7729721069336,
      "Geometry": {
        "BoundingBox": {
          "Width": 0.198987677693367,
          "Height": 0.31296101212501526,
          "Left": 0.07924537360668182,
          "Top": 0.4037395715713501
        }
      }
    }
  ]
}
```

```
}
```

Gestion des ressources Amazon Rekognition Custom Labels

Cette section donne un aperçu du flux de travail que vous utilisez pour former et utiliser un modèle d'étiquettes personnalisées Amazon Rekognition. Des informations générales sur l'utilisation du AWS SDK pour former et utiliser un modèle sont également incluses.

Gestion d'un projet Amazon Rekognition Custom Labels

Dans Amazon Rekognition Custom Labels, vous utilisez un projet pour gérer les modèles que vous créez pour un cas d'utilisation spécifique. Un projet gère les ensembles de données, la formation des modèles, les versions des modèles, l'évaluation des modèles et l'exécution des modèles de votre projet.

Rubriques

- [Suppression d'un projet Amazon Rekognition Custom Labels](#)
- [Décrire un projet \(SDK\)](#)
- [Création d'un projet avec AWS CloudFormation](#)

Suppression d'un projet Amazon Rekognition Custom Labels

Vous pouvez supprimer un projet en utilisant la console Amazon Rekognition ou en appelant l'[DeleteProject](#) API. Pour supprimer un projet, vous devez d'abord supprimer chaque modèle associé. Un projet ou un modèle supprimé ne peut pas être rétabli.

Rubriques

- [Suppression d'un projet Amazon Rekognition Custom Labels \(console\)](#)
- [Suppression d'un projet Amazon Rekognition Custom Labels \(SDK\)](#)

Suppression d'un projet Amazon Rekognition Custom Labels (console)

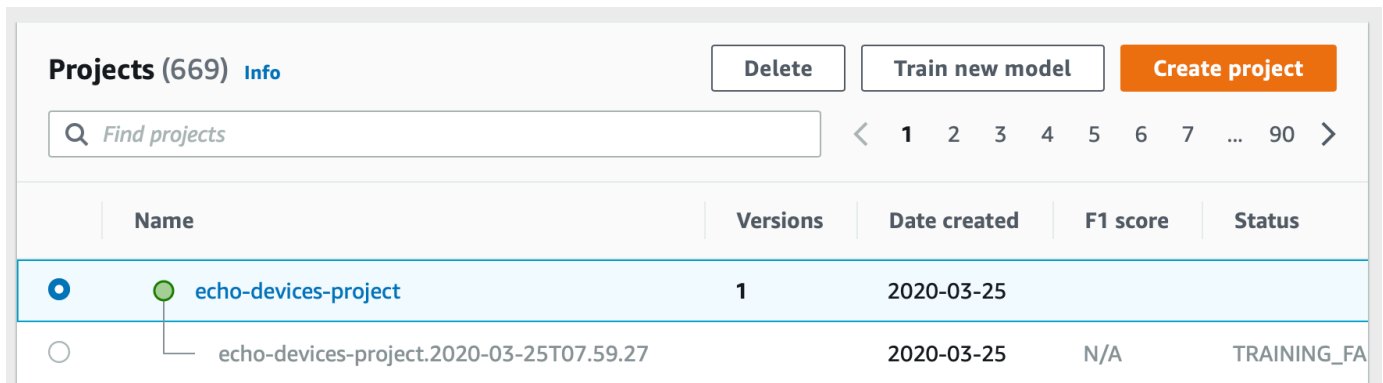
Vous pouvez supprimer un projet depuis la page des projets, ou vous pouvez supprimer un projet depuis la page détaillée d'un projet. La procédure suivante vous explique comment supprimer un projet à l'aide de la page des projets.

La console Amazon Rekognition Custom Labels supprime les modèles et les ensembles de données associés pour vous lors de la suppression du projet. Vous ne pouvez pas supprimer un projet si l'un


de ses modèles est en cours d'exécution ou en cours d'entraînement. Pour arrêter un modèle en cours d'exécution, consultez [Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition \(kit SDK\)](#). Si un modèle est en cours d'apprentissage, attendez qu'il soit terminé avant de supprimer le projet.

Pour supprimer un projet (console)

1. Ouvrez la console Amazon Rekognition à l'[adresse https://console.aws.amazon.com/rekognition/](https://console.aws.amazon.com/rekognition/).
2. Choisissez Utiliser des étiquettes personnalisées.
3. Sélectionnez Get started (Démarrer).
4. Dans le panneau de navigation de gauche, choisissez Projects (Projets).
5. Sur la page Projets, sélectionnez le bouton radio du projet que vous souhaitez supprimer.



6. En haut de la page, choisissez Supprimer. La boîte de dialogue Supprimer le projet s'affiche.
7. Si aucun modèle n'est associé au projet :
 - a. Entrez delete pour supprimer le projet.
 - b. Choisissez Supprimer pour supprimer le projet.
8. Si des modèles ou des ensembles de données sont associés au projet :
 - a. Entrez delete pour confirmer que vous souhaitez supprimer les modèles et les jeux de données.
 - b. Choisissez Supprimer les modèles associés, Supprimer les jeux de données associés ou Supprimer les jeux de données et les modèles associés, selon que le modèle comporte des jeux de données, des modèles ou les deux. La suppression du modèle peut prendre un certain temps.

 Note

La console ne peut pas supprimer les modèles en cours d'entraînement ou en cours d'exécution. Réessayez après avoir arrêté tous les modèles en cours d'exécution répertoriés et attendez que les modèles répertoriés comme étant d'entraînement soient terminés.


Si vous fermez la boîte de dialogue pendant la suppression du modèle, les modèles sont toujours supprimés. Vous pouvez supprimer le projet ultérieurement en répétant cette procédure.

Delete project

✕

Are you sure you want to delete:
echo-devices-project ?

All models in the project must be deleted before the project can be deleted. You cannot delete models which are running or being trained. [Learn more](#)



Delete models

To delete this project, all of its models must be deleted. Model deletion can take up to 5 minutes.

echo-devices-project.2020-03-30T09.28.17
TRAINING_COMPLETED

To confirm deletion, enter delete below.

Close


Delete associated models

- c. Entrez delete pour confirmer que vous souhaitez supprimer le projet.
- d. Choisissez Supprimer pour supprimer le projet.

Delete project ✕

Are you sure you want to delete:
echo-devices-project ?

All models in the project must be deleted before the project can be deleted. You cannot delete models which are running or being trained. [Learn more](#)

 **This project can be deleted**
This project has no models and can be deleted.

To confirm deletion, enter delete below.

Suppression d'un projet Amazon Rekognition Custom Labels (SDK)

Vous supprimez un projet Amazon Rekognition Custom Labels en appelant [DeleteProject](#) en fournissant l'Amazon Resource Name (ARN) du projet que vous souhaitez supprimer. Pour obtenir les ARN des projets enregistrés sur votre AWS compte, appelez [DescribeProjects](#). La réponse inclut un tableau d'[ProjectDescription](#) objets. L'ARN du projet est le `ProjectArn` terrain. Vous pouvez utiliser le nom du projet pour identifier l'ARN du projet. Par exemple, `arn:aws:rekognition:us-east-1:123456789010:project/project name/1234567890123`.

Avant de pouvoir supprimer un projet, vous devez d'abord supprimer tous les modèles et jeux de données du projet. Pour plus d'informations, consultez [Suppression d'un modèle Amazon Rekognition Custom Labels \(SDK\)](#) et [Supprimer un ensemble de données.](#)

La suppression du projet peut prendre quelques instants. Pendant ce temps, l'état du projet est `DELETING`. Le projet est supprimé si un appel ultérieur [DescribeProjects](#) n'inclut pas le projet que vous avez supprimé.

Pour supprimer un projet (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les SDK AWS CLI et les AWS SDK. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez le code suivant pour supprimer un projet.

AWS CLI

Remplacez la valeur `project-arn` de par le nom du projet que vous souhaitez supprimer.

```
aws rekognition delete-project --project-arn project_arn \  
  --profile custom-labels-access
```

Python

Utilisez le code suivant. Fournissez les paramètres de ligne de commande suivants :

- `project_arn`— l'ARN du projet que vous souhaitez supprimer.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Amazon Rekognition Custom Labels project example used in the service  
documentation:  
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/mp-delete-  
project.html  
Shows how to delete an existing Amazon Rekognition Custom Labels project.  
You must first delete any models and datasets that belong to the project.  
"""  
  
import argparse  
import logging  
import time  
import boto3
```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def find_forward_slash(input_string, n):
    """
    Returns the location of '/' after n number of occurrences.
    :param input_string: The string you want to search
    : n: the occurrence that you want to find.
    """
    position = input_string.find('/')
    while position >= 0 and n > 1:
        position = input_string.find('/', position + 1)
        n -= 1
    return position

def delete_project(rek_client, project_arn):
    """
    Deletes an Amazon Rekognition Custom Labels project.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that you want to delete.
    """

    try:
        # Delete the project
        logger.info("Deleting project: %s", project_arn)

        response = rek_client.delete_project(ProjectArn=project_arn)

        logger.info("project status: %s", response['Status'])

        deleted = False

        logger.info("waiting for project deletion: %s", project_arn)

        # Get the project name
        start = find_forward_slash(project_arn, 1) + 1
        end = find_forward_slash(project_arn, 2)
        project_name = project_arn[start:end]
```

```
project_names = [project_name]

while deleted is False:

    project_descriptions = rek_client.describe_projects(
        ProjectNames=project_names)['ProjectDescriptions']

    if len(project_descriptions) == 0:
        deleted = True

    else:
        time.sleep(5)

    logger.info("project deleted: %s",project_arn)

    return True

except ClientError as err:
    logger.exception(
        "Couldn't delete project - %s: %s",
        project_arn, err.response['Error']['Message'])
    raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that you want to delete."
    )

def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
```

```
add_arguments(parser)
args = parser.parse_args()

print(f"Deleting project: {args.project_arn}")

# Delete the project.
session = boto3.Session(profile_name='custom-labels-access')
rekognition_client = session.client("rekognition")

delete_project(rekognition_client,
               args.project_arn)

print(f"Finished deleting project: {args.project_arn}")

except ClientError as err:
    error_message = f"Problem deleting project: {err}"
    logger.exception(error_message)
    print(error_message)

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

Java V2

Utilisez le code suivant. Fournissez les paramètres de ligne de commande suivants :

- `project_arn`— l'ARN du projet que vous souhaitez supprimer.

```
/*
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.List;
import java.util.Objects;
import java.util.logging.Level;
import java.util.logging.Logger;

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.DeleteProjectRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteProjectResponse;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteProject {

    public static final Logger logger =
        Logger.getLogger(DeleteProject.class.getName());

    public static void deleteMyProject(RekognitionClient rekClient, String
        projectArn) throws InterruptedException {

        try {

            logger.log(Level.INFO, "Deleting project: {0}", projectArn);

            // Delete the project

            DeleteProjectRequest deleteProjectRequest =
                DeleteProjectRequest.builder().projectArn(projectArn).build();
            DeleteProjectResponse response =
                rekClient.deleteProject(deleteProjectRequest);

            logger.log(Level.INFO, "Status: {0}", response.status());

            // Wait until deletion finishes

            Boolean deleted = false;

            do {

                DescribeProjectsRequest describeProjectsRequest =
                    DescribeProjectsRequest.builder().build();
                DescribeProjectsResponse describeResponse =
                    rekClient.describeProjects(describeProjectsRequest);
                List<ProjectDescription> projectDescriptions =
                    describeResponse.projectDescriptions();
```

```
        deleted = true;

        for (ProjectDescription projectDescription :
projectDescriptions) {

            if (Objects.equals(projectDescription.projectArn(),
projectArn)) {

                deleted = false;
                logger.log(Level.INFO, "Not deleted: {0}",
projectDescription.projectArn());
                Thread.sleep(5000);
                break;
            }
        }

        } while (Boolean.FALSE.equals(deleted));

        logger.log(Level.INFO, "Project deleted: {0} ", projectArn);

    } catch (

        RekognitionException e) {
        logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
        throw e;
    }

}

public static void main(String[] args) {

    final String USAGE = "\n" + "Usage: " + "<project_arn>\n\n" + "Where:\n"
        + "    project_arn - The ARN of the project that you want to delete.
\n\n";

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

    String projectArn = args[0];

    try {
```



```
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .build();

        // Delete the project.
        deleteMyProject(rekClient, projectArn);

        System.out.println(String.format("Project deleted: %s",
projectArn));

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        System.exit(1);
    }

    catch (InterruptedException intError) {
        logger.log(Level.SEVERE, "Exception while sleeping: {0}",
intError.getMessage());
        System.exit(1);
    }
}
}
```

Décrire un projet (SDK)

Vous pouvez utiliser l'`DescribeProjectsAPI` pour obtenir des informations sur vos projets.

Pour décrire un projet (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les SDKAWS CLI et lesAWS SDK. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez l'exemple de code suivant pour décrire un projet. Remplacez `project_name` par le nom du projet que vous souhaitez décrire. Si vous ne spécifiez pas `--project-names`, les descriptions de tous les projets sont renvoyées.

AWS CLI

```
aws rekognition describe-projects --project-names project_name \  
  --profile custom-labels-access
```

Python

Utilisez le code suivant. Fournissez les paramètres de ligne de commande suivants :

- `project_name` — nom du projet que vous souhaitez décrire. Si vous ne spécifiez pas de nom, les descriptions de tous les projets sont renvoyées.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Shows how to describe an Amazon Rekognition Custom Labels project.  
"""  
  
import argparse  
import logging  
import json  
import boto3  
from botocore.exceptions import ClientError  
  
logger = logging.getLogger(__name__)  
  
def display_project_info(project):  
    """  
    Displays information about a Custom Labels project.  
    :param project: The project that you want to display information about.  
    """  
    print(f"Arn: {project['ProjectArn']}")  
    print(f"Status: {project['Status']}")  
  
    if len(project['Datasets']) == 0:  
        print("Datasets: None")  
    else:  
        print("Datasets:")
```

```
for dataset in project['Datasets']:
    print(f"\tCreated: {str(dataset['CreationTimestamp'])}")
    print(f"\tType: {dataset['DatasetType']}")
    print(f"\tARN: {dataset['DatasetArn']}")
    print(f"\tStatus: {dataset['Status']}")
    print(f"\tStatus message: {dataset['StatusMessage']}")
    print(f"\tStatus code: {dataset['StatusMessageCode']}")
    print()
print()

def describe_projects(rek_client, project_name):
    """
    Describes an Amazon Rekognition Custom Labels project, or all projects.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: The project you want to describe. Pass None to describe
    all projects.
    """

    try:
        # Describe the project
        if project_name is None:
            logger.info("Describing all projects.")
        else:
            logger.info("Describing project: %s.", project_name)

        if project_name is None:
            response = rek_client.describe_projects()
        else:
            project_names = json.loads('["' + project_name + '"]')
            response = rek_client.describe_projects(ProjectNames=project_names)

        print('Projects\n-----')
        if len(response['ProjectDescriptions']) == 0:
            print("Project(s) not found.")
        else:
            for project in response['ProjectDescriptions']:
                display_project_info(project)

        logger.info("Finished project description.")

    except ClientError as err:
        logger.exception(
            "Couldn't describe project - %s: %s",
```

```
        project_name, err.response['Error']['Message'] )
    raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "--project_name", help="The name of the project that you want to
describe.", required=False
    )

def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)

        args = parser.parse_args()

        print(f"Describing projects: {args.project_name}")

        # Describe the project.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        describe_projects(rekognition_client,
                          args.project_name)

        if args.project_name is None:
            print("Finished describing all projects.")
        else:
            print("Finished describing project %s.", args.project_name)

    except ClientError as err:
```

```
        error_message = f"Problem describing project: {err}"
        logger.exception(error_message)
        print(error_message)

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

Java V2

Utilisez le code suivant. Fournissez les paramètres de ligne de commande suivants :

- `project_name`— l'ARN du projet que vous souhaitez décrire. Si vous ne spécifiez pas de nom, les descriptions de tous les projets sont renvoyées.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetMetadata;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DescribeProjects {

    public static final Logger logger =
        Logger.getLogger(DescribeProjects.class.getName());
```

```
public static void describeMyProjects(RekognitionClient rekClient, String
projectName) {

    DescribeProjectsRequest descProjects = null;

    // If a single project name is supplied, build projectNames argument

    List<String> projectNames = new ArrayList<String>();

    if (projectName == null) {
        descProjects = DescribeProjectsRequest.builder().build();
    } else {
        projectNames.add(projectName);
        descProjects =
DescribeProjectsRequest.builder().projectNames(projectNames).build();
    }

    // Display useful information for each project.

    DescribeProjectsResponse resp =
rekClient.describeProjects(descProjects);

    for (ProjectDescription projectDescription : resp.projectDescriptions())
    {

        System.out.println("ARN: " + projectDescription.projectArn());
        System.out.println("Status: " +
projectDescription.statusAsString());
        if (projectDescription.hasDatasets()) {
            for (DatasetMetadata datasetDescription :
projectDescription.datasets()) {
                System.out.println("\tdataset Type: " +
datasetDescription.datasetTypeAsString());
                System.out.println("\tdataset ARN: " +
datasetDescription.datasetArn());
                System.out.println("\tdataset Status: " +
datasetDescription.statusAsString());
            }
        }
        System.out.println();
    }
}
```

```
public static void main(String[] args) {

    String projectArn = null;

    // Get command line arguments

    final String USAGE = "\n" + "Usage: " + "<project_name>\n\n" + "Where:
\n"
        + "    project_name - (Optional) The name of the project that you
want to describe. If not specified, all projects "
        + "are described.\n\n";

    if (args.length > 1) {
        System.out.println(USAGE);
        System.exit(1);
    }

    if (args.length == 1) {
        projectArn = args[0];
    }

    try {

        // Get the Rekognition client
        RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

        // Describe projects

        describeMyProjects(rekClient, projectArn);

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        System.exit(1);
    }

}
```

}

Création d'un projet avecAWS CloudFormation

Amazon Rekognition Custom Labels est intégré àAWS CloudFormation, un service qui vous aide à modéliser et à configurer vosAWS ressources afin que vous puissiez consacrer moins de temps à la création et à la gestion de vos ressources et de votre infrastructure. Vous créez un modèle qui décrit toutes lesAWS ressources que vous souhaitez etAWS CloudFormation s'occupe de la mise en service et de la configuration de ces ressources.

Vous pouvez l'utiliserAWS CloudFormation pour provisionner et configurer des projets Amazon Rekognition Custom Labels.

Lorsque vous utilisezAWS CloudFormation, vous pouvez réutiliser votre modèle pour configurer vos projets Amazon Rekognition Custom Labels de manière cohérente et répétée. Il suffit de décrire vos projets une fois, puis d'allouer les mêmes projets à l'infini dans plusieursAWS comptes et régions.

Labels etAWS CloudFormation modèles Amazon Rekognition personnalisés

Pour allouer et configurer des projets pour Amazon Rekognition Custom Labels et les services associés, vous devez maîtriser les [AWS CloudFormationmodèles](#). Les modèles sont des fichiers texte formatés en JSON ou YAML. Ces modèles décrivent les ressources que vous souhaitez allouer dans vos piles AWS CloudFormation. Si JSON ou YAML ne vous est pas familier, vous pouvez utiliser AWS CloudFormation Designer pour vous aider à démarrer avec des modèles AWS CloudFormation. Pour plus d'informations, consultez [Qu'est-ce que AWS CloudFormation Designer ?](#) dans le AWS CloudFormationGuide de l'utilisateur.

Pour des informations de référence sur les projets Amazon Rekognition Custom Labels, y compris des exemples de modèles JSON et YAML, consultez la [Référence de type de ressource Rekognition](#).

En savoir plus sur AWS CloudFormation

Pour en savoir plus sur AWS CloudFormation, consultez les ressources suivantes :

- [AWS CloudFormation](#)
- [Guide de l'utilisateur AWS CloudFormation](#)
- [Référence API AWS CloudFormation](#)
- [Guide de l'utilisateur de l'interface de ligne de commande AWS CloudFormation](#)

Gestion des jeux de données

Un jeu de données contient les images et les étiquettes attribuées que vous utilisez pour entraîner ou tester un modèle. Les rubriques de cette section vous montrent comment gérer un jeu de données à l'aide de la console Amazon Rekognition Custom Labels et du AWS SDK.

Rubriques

- [Ajouter un jeu de données à un projet](#)
- [Ajouter d'autres images à un jeu de données](#)
- [Création d'un jeu de données à l'aide d'un jeu de données existant \(SDK\)](#)
- [Décrire un ensemble de données \(SDK\)](#)
- [Répertorier les entrées du jeu de données \(SDK\)](#)
- [Distribution d'un jeu de données d'entraînement \(SDK\)](#)
- [Supprimer un ensemble de données.](#)

Ajouter un jeu de données à un projet

Vous pouvez ajouter un jeu de données d'entraînement ou un jeu de données de test à un projet existant. Si vous souhaitez remplacer un jeu de données existant, supprimez-le d'abord. Pour plus d'informations, veuillez consulter [Supprimer un ensemble de données.](#) Ajoutez ensuite le nouveau jeu de données.

Rubriques

- [Ajouter un jeu de données à un projet \(console\)](#)
- [Ajouter un jeu de données à un projet \(SDK\)](#)

Ajouter un jeu de données à un projet (console)

Vous pouvez ajouter un ensemble de données de formation ou de test à un projet à l'aide de la console Amazon Rekognition Custom Labels.

Pour ajouter un ensemble de données à un projet

1. Ouvrez la console Amazon Rekognition à l'[adresse https://console.aws.amazon.com/rekognition/](https://console.aws.amazon.com/rekognition/).

2. Dans le volet de gauche, choisissez Utiliser des étiquettes personnalisées. La page d'accueil des étiquettes personnalisées Amazon Rekognition s'affiche.
3. Dans le panneau de navigation de gauche, choisissez Projects. La vue Projets s'affiche.
4. Choisissez le projet auquel vous souhaitez ajouter un ensemble de données.
5. Dans le panneau de navigation de gauche, sous le nom du projet, choisissez ensemble de données.
6. Si le projet ne possède pas de jeu de données existant, la page Créer un jeu de données s'affiche. Procédez comme suit :
 - a. Sur la page Créer un jeu de données, entrez les informations relatives à la source de l'image. Pour plus d'informations, veuillez consulter [the section called "Création de jeux de données avec des images"](#).
 - b. Choisissez Créer un jeu de données pour créer le jeu de données.
7. Si le projet possède un ensemble de données existant (formation ou test), la page des détails du projet s'affiche. Procédez comme suit :
 - a. Sur la page des détails du projet, sélectionnez Actions.
 - b. Si vous souhaitez ajouter un jeu de données d'entraînement, choisissez Créer un jeu de données d'entraînement.
 - c. Si vous souhaitez ajouter un jeu de données de test, choisissez Créer un jeu de données de test.
 - d. Sur la page Créer un jeu de données, entrez les informations relatives à la source de l'image. Pour plus d'informations, veuillez consulter [the section called "Création de jeux de données avec des images"](#).
 - e. Choisissez Créer un jeu de données pour créer le jeu de données.
8. Ajoutez des images à votre jeu de données. Pour plus d'informations, veuillez consulter [Ajouter d'autres images \(console\)](#).
9. Ajoutez des étiquettes à votre jeu de données. Pour plus d'informations, veuillez consulter [Ajout de nouvelles étiquettes \(console\)](#).
10. Ajoutez des étiquettes à vos images. Si vous ajoutez des étiquettes au niveau de l'image, consultez [the section called "Attribution d'étiquettes au niveau de l'image à une image"](#). Si vous ajoutez des cadres de sélection, consultez [Étiquetage des objets à l'aide de cadres de délimitation](#). Pour plus d'informations, veuillez consulter [Utilisation des jeux de données](#).

Ajouter un jeu de données à un projet (SDK)

Vous pouvez ajouter un ensemble de données de train ou de test à un ensemble de données de test à un projet existant de la manière suivante :

- Créez un ensemble de données à l'aide d'un fichier manifeste. Pour plus d'informations, veuillez consulter [Création d'un jeu de données à l'aide d'un fichier manifeste \(SDK\) SageMaker Ground Truth](#).
- Créez un jeu de données vide et remplissez-le ensuite. L'exemple suivant montre comment créer un ensemble de données vide. Pour ajouter des entrées après avoir créé un jeu de données vide, consultez [Ajouter d'autres images à un jeu de données](#).

Pour ajouter un jeu de données à un projet (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS CLI et l'AWS SDK. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez les exemples suivants pour ajouter des lignes JSON à un ensemble de données.

CLI

Remplacez `project_arn` par le projet auquel vous souhaitez ajouter le jeu de données. Remplacez `dataset_type` par `TRAIN` pour créer un jeu de données d'entraînement ou `TEST` pour créer un ensemble de données de test.

```
aws rekognition create-dataset --project-arn project_arn \  
  --dataset-type dataset_type \  
  --profile custom-labels-access
```

Python

Utilisez le code suivant pour créer un ensemble de données. Fournissez les options de ligne de commande suivantes :

- `project_arn`— ARN du projet auquel vous souhaitez ajouter le jeu de données de test.
- `type`— le type de jeu de données que vous souhaitez créer (entraînement ou test)

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0
```

```
import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_empty_dataset(rek_client, project_arn, dataset_type):
    """
    Creates an empty Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
    dataset.
    :param dataset_type: The type of the dataset that you want to create (train
    or test).
    """

    try:
        #Create the dataset.
        logger.info("Creating empty %s dataset for project %s",
                    dataset_type, project_arn)

        dataset_type=dataset_type.upper()

        response = rek_client.create_dataset(
            ProjectArn=project_arn, DatasetType=dataset_type
        )

        dataset_arn=response['DatasetArn']

        logger.info("dataset ARN: %s", dataset_arn)

        finished=False
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

            status=dataset['DatasetDescription']['Status']

            if status == "CREATE_IN_PROGRESS":
```

```
        logger.info("Creating dataset: %s ", dataset_arn))
        time.sleep(5)
        continue

    if status == "CREATE_COMPLETE":
        logger.info("Dataset created: %s", dataset_arn)
        finished=True
        continue

    if status == "CREATE_FAILED":
        error_message = f"Dataset creation failed: {status} :
{dataset_arn}"
        logger.exception(error_message)
        raise Exception(error_message)

    error_message = f"Failed. Unexpected state for dataset creation:
{status} : {dataset_arn}"
    logger.exception(error_message)
    raise Exception(error_message)

    return dataset_arn

except ClientError as err:
    logger.exception("Couldn't create dataset: %s", err.response['Error']
['Message'])
    raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
the empty dataset."
    )

    parser.add_argument(
        "dataset_type", help="The type of the empty dataset that you want to
create (train or test).")
    )
```

```
def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Creating empty {args.dataset_type} dataset for project
{args.project_arn}")

        # Create the empty dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        dataset_arn=create_empty_dataset(rekognition_client,
            args.project_arn,
            args.dataset_type.lower())

        print(f"Finished creating empty dataset: {dataset_arn}")

    except ClientError as err:
        logger.exception("Problem creating empty dataset: %s", err)
        print(f"Problem creating empty dataset: {err}")
    except Exception as err:
        logger.exception("Problem creating empty dataset: %s", err)
        print(f"Problem creating empty dataset: {err}")

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

Java V2

Utilisez le code suivant pour créer un ensemble de données. Fournissez les options de ligne de commande suivantes :

- `project_arn`— ARN du projet auquel vous souhaitez ajouter le jeu de données de test.

- type— le type de jeu de données que vous souhaitez créer (entraînement ou test)

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

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.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateEmptyDataset {

    public static final Logger logger =
        Logger.getLogger(CreateEmptyDataset.class.getName());

    public static String createMyEmptyDataset(RekognitionClient rekClient,
        String projectArn, String datasetType)
        throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Creating empty {0} dataset for project :
{1}",
                new Object[] { datasetType.toString(), projectArn });

            DatasetType requestDatasetType = null;
```

```
switch (datasetType) {
    case "train":
        requestDatasetType = DatasetType.TRAIN;
        break;
    case "test":
        requestDatasetType = DatasetType.TEST;
        break;
    default:
        logger.log(Level.SEVERE, "Unrecognized dataset type: {0}",
datasetType);
        throw new Exception("Unrecognized dataset type: " +
datasetType);
}

        CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)
            .datasetType(requestDatasetType).build();

        CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

        boolean created = false;

        //Wait until updates finishes

        do {

            DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                .datasetArn(response.datasetArn()).build();
            DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

            DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

            DatasetStatus status = datasetDescription.status();

            logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());

            switch (status) {
```



```
        case CREATE_COMPLETE:
            logger.log(Level.INFO, "Dataset created");
            created = true;
            break;

        case CREATE_IN_PROGRESS:
            Thread.sleep(5000);
            break;

        case CREATE_FAILED:
            String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                + datasetDescription.statusMessage() + " " +
response.datasetArn();
            logger.log(Level.SEVERE, error);
            throw new Exception(error);

        default:
            String unexpectedError = "Unexpected creation state: " +
datasetDescription.statusAsString() + " "
                + datasetDescription.statusMessage() + " " +
response.datasetArn();
            logger.log(Level.SEVERE, unexpectedError);
            throw new Exception(unexpectedError);
    }

    } while (created == false);

    return response.datasetArn();

} catch (RekognitionException e) {
    logger.log(Level.SEVERE, "Could not create dataset: {0}",
e.getMessage());
    throw e;
}

}

public static void main(String args[]) {

    String datasetType = null;
    String datasetArn = null;
    String projectArn = null;
```

```
        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>\n\n" + "Where:\n"
            + "    project_arn - the ARN of the project that you want to add copy the data to.\n\n"
            + "    dataset_type - the type of the empty dataset that you want to create (train or test).\n\n";

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

        projectArn = args[0];
        datasetType = args[1];

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Create the dataset
            datasetArn = createMyEmptyDataset(rekClient, projectArn, datasetType);

            System.out.println(String.format("Created dataset: %s", datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}", rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }
    }
}
```

```
}  
  
}
```

3. Ajoutez des images au jeu de données. Pour plus d'informations, veuillez consulter [Ajouter d'autres images \(SDK\)](#).

Ajouter d'autres images à un jeu de données

Vous pouvez ajouter d'autres images à vos ensembles de données en utilisant la console Amazon Rekognition Custom Labels ou en appelant l'UpdateDatasetEntriesAPI.

Rubriques

- [Ajouter d'autres images \(console\)](#)
- [Ajouter d'autres images \(SDK\)](#)

Ajouter d'autres images (console)

Lorsque vous utilisez la console Amazon Rekognition Custom Labels, vous chargez des images depuis votre ordinateur local. Les images sont ajoutées à l'emplacement du compartiment Amazon S3 (console ou externe) où sont stockées les images utilisées pour créer le jeu de données.

Pour ajouter d'autres images à votre jeu de données (console)

1. Ouvrez la console Amazon Rekognition à l'[adresse https://console.aws.amazon.com/rekognition/](https://console.aws.amazon.com/rekognition/).
2. Dans le volet de gauche, choisissez Utiliser des étiquettes personnalisées. La page d'accueil des étiquettes personnalisées Amazon Rekognition s'affiche.
3. Dans le panneau de navigation de gauche, choisissez Projects. La vue Projets s'affiche.
4. Choisissez le projet que vous souhaitez utiliser.
5. Dans le panneau de navigation de gauche, sous le nom du projet, choisissez ensemble de données.
6. Choisissez Actions et sélectionnez le jeu de données auquel vous souhaitez ajouter des images.
7. Choisissez les images que vous souhaitez télécharger dans l'ensemble de données. Vous pouvez faire glisser les images ou choisir les images que vous souhaitez télécharger depuis votre ordinateur local. Vous pouvez télécharger jusqu'à 30 images à la fois.

8. Choisissez Charger des images.
9. Choisissez Save Changes (Enregistrer les modifications).
10. Étiquetez les images. Pour plus d'informations, veuillez consulter [Étiquetage des images](#).

Ajouter d'autres images (SDK)

UpdateDatasetEntries met à jour ou ajoute des lignes JSON à un fichier manifeste. Vous transmettez les lignes JSON sous la forme d'un objet de données codé en octets 64 dans le GroundTruth champ. Si vous utilisez un AWS SDK pour appeler UpdateDatasetEntries, le SDK code les données pour vous. Chaque ligne JSON contient des informations relatives à une seule image, telles que les étiquettes attribuées ou les informations relatives au cadre de délimitation. Par exemple :

```
{"source-ref": "s3://bucket/image", "BB": {"annotations": [{"left": 1849, "top": 1039, "width": 422, "height": 283, "class_id": 0}, {"left": 1849, "top": 1340, "width": 443, "height": 415, "class_id": 1}, {"left": 2637, "top": 1380, "width": 676, "height": 338, "class_id": 2}, {"left": 2634, "top": 1051, "width": 673, "height": 338, "class_id": 3}], "image_size": [{"width": 4000, "height": 2667, "depth": 3}], "BB-metadata": {"job-name": "labeling-job/BB", "class-map": {"0": "comparator", "1": "pot_resistor", "2": "ir_phototransistor", "3": "ir_led"}, "human-annotated": "yes", "objects": [{"confidence": 1}, {"confidence": 1}, {"confidence": 1}, {"confidence": 1}], "creation-date": "2021-06-22T10:11:18.006Z", "type": "groundtruth/object-detection"}}
```

Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).

Utilisez `source-ref` le champ comme clé pour identifier les images que vous souhaitez mettre à jour. Si l'ensemble de données ne contient pas de valeur `source-ref` champ correspondante, la ligne JSON est ajoutée en tant que nouvelle image.

Pour ajouter d'autres images à un jeu de données (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS CLI et l'AWS SDK. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez les exemples suivants pour ajouter des lignes JSON à un ensemble de données.

CLI

Remplacez la valeur de `GroundTruth` par les lignes JSON que vous souhaitez utiliser. Vous devez éviter tout caractère spécial dans la ligne JSON.

```
aws rekognition update-dataset-entries\
  --dataset-arn dataset_arn \
  --changes '{"GroundTruth" : "{\\"source-ref\\":\\"s3://your_bucket/your_image
\\" ,\\"BB\\":{\\"annotations\\":[{\\"left\\":1776,\\"top\\":1017,\\"width\\":458,\\"height
\\" :317,\\"class_id\\":0},{\\"left\\":1797,\\"top\\":1334,\\"width\\":418,\\"height
\\" :415,\\"class_id\\":1},{\\"left\\":2597,\\"top\\":1361,\\"width\\":655,\\"height
\\" :329,\\"class_id\\":2},{\\"left\\":2581,\\"top\\":1020,\\"width\\":689,\\"height
\\" :338,\\"class_id\\":3}],\\"image_size\\":[{\\"width\\":4000,\\"height\\":2667,
\\"depth\\":3}]],\\"BB-metadata\\":{\\"job-name\\":\\"labeling-job/BB\\",\\"class-map
\\" :{\\"0\\":\\"comparator\\",\\"1\\":\\"pot_resistor\\",\\"2\\":\\"ir_phototransistor\\",
\\"3\\":\\"ir_led\\"},\\"human-annotated\\":\\"yes\\",\\"objects\\":[{\\"confidence\\":1},
{\\"confidence\\":1},{\\"confidence\\":1}],\\"creation-date\\":
\\"2021-06-22T10:10:48.492Z\\",\\"type\\":\\"groundtruth/object-detection\\"}]"}' \
  --cli-binary-format raw-in-base64-out \
  --profile custom-labels-access
```

Python

Utilisez le code suivant. Fournissez les paramètres de ligne de commande suivants :

- `dataset_arn` — ARN du jeu de données que vous souhaitez mettre à jour.
- `updates_file` : fichier contenant les mises à jour de la ligne JSON.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to add entries to an Amazon Rekognition Custom Labels dataset.
"""

import argparse
import logging
import time
import json
```

```
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def update_dataset_entries(rek_client, dataset_arn, updates_file):
    """
    Adds dataset entries to an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that you want to update.
    :param updates_file: The manifest file of JSON Lines that contains the
    updates.
    """

    try:
        status=""
        status_message=""

        # Update dataset entries.
        logger.info("Updating dataset %s", dataset_arn)

        with open(updates_file) as f:
            manifest_file = f.read()

        changes=json.loads('{ "GroundTruth" : ' +
            json.dumps(manifest_file) +
            '}')

        rek_client.update_dataset_entries(
            Changes=changes, DatasetArn=dataset_arn
        )

        finished=False
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

            status=dataset['DatasetDescription']['Status']
            status_message=dataset['DatasetDescription']['StatusMessage']

            if status == "UPDATE_IN_PROGRESS":
```

```
        logger.info("Updating dataset: %s ", dataset_arn)
        time.sleep(5)
        continue

    if status == "UPDATE_COMPLETE":
        logger.info("Dataset updated: %s : %s : %s",
                    status, status_message, dataset_arn)
        finished=True
        continue

    if status == "UPDATE_FAILED":
        error_message = f"Dataset update failed: {status} :
{status_message} : {dataset_arn}"
        logger.exception(error_message)
        raise Exception (error_message)

        error_message = f"Failed. Unexpected state for dataset update:
{status} : {status_message} : {dataset_arn}"
        logger.exception(error_message)
        raise Exception(error_message)

    logger.info("Added entries to dataset")

    return status, status_message

except ClientError as err:
    logger.exception("Couldn't update dataset: %s", err.response['Error']
['Message'])
    raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to update."
    )

    parser.add_argument(
```

```
        "updates_file", help="The manifest file of JSON Lines that contains the
updates."
    )

def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        #get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Updating dataset {args.dataset_arn} with entries from
{args.updates_file}.")

        # Update the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        status, status_message=update_dataset_entries(rekognition_client,
            args.dataset_arn,
            args.updates_file)

        print(f"Finished updates dataset: {status} : {status_message}")

    except ClientError as err:
        logger.exception("Problem updating dataset: %s", err)
        print(f"Problem updating dataset: {err}")

    except Exception as err:
        logger.exception("Problem updating dataset: %s", err)
        print(f"Problem updating dataset: {err}")

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


Java V2

- `dataset_arn` — ARN du jeu de données que vous souhaitez mettre à jour.
- `update_file` — le fichier qui contient les mises à jour de la ligne JSON.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
 */
package com.example.rekognition;

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.DatasetChanges;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.UpdateDatasetEntriesRequest;
import
    software.amazon.awssdk.services.rekognition.model.UpdateDatasetEntriesResponse;

import java.io.FileInputStream;
import java.io.InputStream;
import java.util.logging.Level;
import java.util.logging.Logger;

public class UpdateDatasetEntries {

    public static final Logger logger =
        Logger.getLogger(UpdateDatasetEntries.class.getName());

    public static String updateMyDataset(RekognitionClient rekClient, String
datasetArn,
        String updateFile
        ) throws Exception, RekognitionException {
```

```
try {

    logger.log(Level.INFO, "Updating dataset {0}",
        new Object[] { datasetArn});

    InputStream sourceStream = new FileInputStream(updateFile);
    SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

    DatasetChanges datasetChanges = DatasetChanges.builder()
        .groundTruth(sourceBytes).build();

    UpdateDatasetEntriesRequest updateDatasetEntriesRequest =
UpdateDatasetEntriesRequest.builder()
        .changes(datasetChanges)
        .datasetArn(datasetArn)
        .build();

    UpdateDatasetEntriesResponse response =
rekClient.updateDatasetEntries(updateDatasetEntriesRequest);

    boolean updated = false;

    //Wait until update completes

    do {

        DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
            .datasetArn(datasetArn).build();
        DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

        DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

        DatasetStatus status = datasetDescription.status();

        logger.log(Level.INFO, " dataset ARN: {0} ", datasetArn);

        switch (status) {

            case UPDATE_COMPLETE:
```

```
        logger.log(Level.INFO, "Dataset updated");
        updated = true;
        break;

    case UPDATE_IN_PROGRESS:
        Thread.sleep(5000);
        break;

    case UPDATE_FAILED:
        String error = "Dataset update failed: " +
datasetDescription.statusAsString() + " "
            + datasetDescription.statusMessage() + " " +
datasetArn;

        logger.log(Level.SEVERE, error);
        throw new Exception(error);

    default:
        String unexpectedError = "Unexpected update state: " +
datasetDescription.statusAsString() + " "
            + datasetDescription.statusMessage() + " " +
datasetArn;

        logger.log(Level.SEVERE, unexpectedError);
        throw new Exception(unexpectedError);
    }

    } while (updated == false);

    return datasetArn;

} catch (RekognitionException e) {
    logger.log(Level.SEVERE, "Could not update dataset: {0}",
e.getMessage());
    throw e;
}

}

public static void main(String args[]) {

    String updatesFile = null;
    String datasetArn = null;
```

```
        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_arn>  
<updates_file>\n\n" + "Where:\n"  
            + "    dataset_arn - the ARN of the dataset that you want to  
update.\n\n"  
            + "    update_file - The file that includes in JSON Line updates.  
\n\n";  
  
        if (args.length != 2) {  
            System.out.println(USAGE);  
            System.exit(1);  
        }  
  
        datasetArn = args[0];  
        updatesFile = args[1];  
  
        try {  
  
            // Get the Rekognition client.  
            RekognitionClient rekClient = RekognitionClient.builder()  
                .credentialsProvider(ProfileCredentialsProvider.create("custom-  
labels-access"))  
                .region(Region.US_WEST_2)  
                .build();  
  
            // Update the dataset  
            datasetArn = updateMyDataset(rekClient, datasetArn, updatesFile);  
  
            System.out.println(String.format("Dataset updated: %s",  
datasetArn));  
  
            rekClient.close();  
  
        } catch (RekognitionException rekError) {  
            logger.log(Level.SEVERE, "Rekognition client error: {0}",  
rekError.getMessage());  
            System.exit(1);  
        } catch (Exception rekError) {  
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());  
            System.exit(1);  
        }  
  
    }  
}
```

```
}
```

Création d'un jeu de données à l'aide d'un jeu de données existant (SDK)

La procédure suivante vous montre comment créer un jeu de données à partir d'un jeu de données existant à l'aide de l'[CreateDataset](#) opération.

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS CLI et les AWS SDK. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez l'exemple de code suivant pour créer un ensemble de données en copiant un autre ensemble de données.

AWS CLI

Utilisez le code suivant pour créer l'ensemble de données. Remplacez les éléments suivants :

- `project_arn`— ARN du projet auquel vous souhaitez ajouter le jeu de données.
- `dataset_type`— avec le type de jeu de données (TRAIN ou TEST) que vous souhaitez créer dans le projet.
- `dataset_arn`— avec l'ARN du jeu de données que vous souhaitez copier.

```
aws rekognition create-dataset --project-arn project_arn \  
  --dataset-type dataset_type \  
  --dataset-source '{ "DatasetArn" : "dataset_arn" }' \  
  --profile custom-labels-access
```

Python

L'exemple suivant crée un ensemble de données à partir d'un jeu de données existant et affiche son ARN.

Pour exécuter le programme, saisissez les arguments de ligne de commande suivants :

- `project_arn`— ARN du projet que vous souhaitez utiliser.
- `dataset_type`— le type du jeu de données du projet que vous souhaitez créer (`trainoutest`).

- `dataset_arn`— ARN du jeu de données à partir duquel vous souhaitez créer le jeu de données.

```
# Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)

import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_dataset_from_existing_dataset(rek_client, project_arn, dataset_type,
dataset_arn):
    """
    Creates an Amazon Rekognition Custom Labels dataset using an existing
    dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
    dataset.
    :param dataset_type: The type of the dataset that you want to create (train
    or test).
    :param dataset_arn: The ARN of the existing dataset that you want to use.
    """

    try:
        # Create the dataset

        dataset_type=dataset_type.upper()

        logger.info(
            "Creating %s dataset for project %s from dataset %s.",
            dataset_type,project_arn, dataset_arn)

        dataset_source = json.loads(
```

```
        '{ "DatasetArn": "' + dataset_arn + "'"
    )

    response = rek_client.create_dataset(
        ProjectArn=project_arn, DatasetType=dataset_type,
DatasetSource=dataset_source
    )

    dataset_arn = response['DatasetArn']

    logger.info("New dataset ARN: %s", dataset_arn)

    finished = False
    while finished is False:

        dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

        status = dataset['DatasetDescription']['Status']

        if status == "CREATE_IN_PROGRESS":

            logger.info(("Creating dataset: %s ", dataset_arn))
            time.sleep(5)
            continue

        if status == "CREATE_COMPLETE":
            logger.info("Dataset created: %s", dataset_arn)
            finished = True
            continue

        if status == "CREATE_FAILED":
            error_message = f"Dataset creation failed: {status} :
{dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        error_message = f"Failed. Unexpected state for dataset creation:
{status} : {dataset_arn}"
        logger.exception(error_message)
        raise Exception(error_message)

    return dataset_arn

except ClientError as err:
```

```
        logger.exception(
            "Couldn't create dataset: %s",err.response['Error']['Message'] )
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
the dataset."
    )

    parser.add_argument(
        "dataset_type", help="The type of the dataset that you want to create
(train or test).")
    )

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to copy from."
    )

def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Creating {args.dataset_type} dataset for project
{args.project_arn}")

        # Create the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
```



```
rekognition_client = session.client("rekognition")

dataset_arn = create_dataset_from_existing_dataset(rekognition_client,
                                                  args.project_arn,
                                                  args.dataset_type,
                                                  args.dataset_arn)

print(f"Finished creating dataset: {dataset_arn}")

except ClientError as err:
    logger.exception("Problem creating dataset: %s", err)
    print(f"Problem creating dataset: {err}")
except Exception as err:
    logger.exception("Problem creating dataset: %s", err)
    print(f"Problem creating dataset: {err}")

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

Java V2

L'exemple suivant crée un ensemble de données à partir d'un jeu de données existant et affiche son ARN.

Pour exécuter le programme, saisissez les arguments de ligne de commande suivants :

- `project_arn`— ARN du projet que vous souhaitez utiliser.
- `dataset_type`— le type du jeu de données du projet que vous souhaitez créer (`trainoutest`).
- `dataset_arn`— ARN du jeu de données à partir duquel vous souhaitez créer le jeu de données.

```
/*
 Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 SPDX-License-Identifier: Apache-2.0
 */

package com.example.rekognition;

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.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetSource;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateDatasetExisting {

    public static final Logger logger =
        Logger.getLogger(CreateDatasetExisting.class.getName());

    public static String createMyDataset(RekognitionClient rekClient, String
        projectArn, String datasetType,
        String existingDatasetArn) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Creating {0} dataset for project : {1} from
                dataset {2} ",
                new Object[] { datasetType.toString(), projectArn,
                    existingDatasetArn });

            DatasetType requestDatasetType = null;

            switch (datasetType) {
                case "train":
                    requestDatasetType = DatasetType.TRAIN;
                    break;
                case "test":
                    requestDatasetType = DatasetType.TEST;
                    break;
                default:
                    logger.log(Level.SEVERE, "Unrecognized dataset type: {0}",
                        datasetType);
            }
        }
    }
}
```

```
        throw new Exception("Unrecognized dataset type: " +
datasetType);
    }

    DatasetSource datasetSource =
DatasetSource.builder().datasetArn(existingDatasetArn).build();

    CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)

.datasetType(requestDatasetType).datasetSource(datasetSource).build();

    CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

    boolean created = false;

    //Wait until create finishes

    do {

        DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
            .datasetArn(response.datasetArn()).build();
        DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

        DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

        DatasetStatus status = datasetDescription.status();

        logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());

        switch (status) {

            case CREATE_COMPLETE:
                logger.log(Level.INFO, "Dataset created");
                created = true;
                break;

            case CREATE_IN_PROGRESS:
```

```
        Thread.sleep(5000);
        break;

        case CREATE_FAILED:
            String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                + datasetDescription.statusMessage() + " " +
response.datasetArn();
            logger.log(Level.SEVERE, error);
            throw new Exception(error);

        default:
            String unexpectedError = "Unexpected creation state: " +
datasetDescription.statusAsString() + " "
                + datasetDescription.statusMessage() + " " +
response.datasetArn();
            logger.log(Level.SEVERE, unexpectedError);
            throw new Exception(unexpectedError);
    }

    } while (created == false);

    return response.datasetArn();

} catch (RekognitionException e) {
    logger.log(Level.SEVERE, "Could not create dataset: {0}",
e.getMessage());
    throw e;
}

}

public static void main(String[] args) {

    String datasetType = null;
    String datasetArn = null;
    String projectArn = null;
    String datasetSourceArn = null;

    final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>
<dataset_arn>\n\n" + "Where:\n"
        + "    project_arn - the ARN of the project that you want to add
copy the datast to.\n\n"
```

```
        + "    dataset_type - the type of the dataset that you want to
create (train or test).\n\n"
        + "    dataset_arn - the ARN of the dataset that you want to copy
from.\n\n";

    if (args.length != 3) {
        System.out.println(USAGE);
        System.exit(1);
    }

    projectArn = args[0];
    datasetType = args[1];
    datasetSourceArn = args[2];

    try {

        // Get the Rekognition client
        RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

        // Create the dataset
        datasetArn = createMyDataset(rekClient, projectArn, datasetType,
datasetSourceArn);

        System.out.println(String.format("Created dataset: %s",
datasetArn));

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        System.exit(1);
    } catch (Exception rekError) {
        logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
        System.exit(1);
    }

}
```

```
}
```

Décrire un ensemble de données (SDK)

Vous pouvez utiliser l'`DescribeDatasetAPI` pour obtenir des informations sur un ensemble de données.

Pour décrire un jeu de données (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS CLI et les AWS SDK. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez l'exemple de code suivant pour décrire un ensemble de données.

AWS CLI

Remplacez la valeur `dataset-arn` par l'ARN du jeu de données que vous souhaitez décrire.

```
aws rekognition describe-dataset --dataset-arn dataset_arn \  
  --profile custom-labels-access
```

Python

Utilisez le code suivant. Fournissez les paramètres de ligne de commande suivants :

- `dataset_arn` — ARN du jeu de données que vous souhaitez décrire.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Shows how to describe an Amazon Rekognition Custom Labels dataset.  
"""  
  
import argparse  
import logging  
import boto3  
  
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)

def describe_dataset(rek_client, dataset_arn):
    """
    Describes an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that you want to describe.
    """

    try:
        # Describe the dataset
        logger.info("Describing dataset %s", dataset_arn)

        dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

        description = dataset['DatasetDescription']

        print(f"Created: {str(description['CreationTimestamp'])}")
        print(f"Updated: {str(description['LastUpdatedTimestamp'])}")
        print(f"Status: {description['Status']}")
        print(f"Status message: {description['StatusMessage']}")
        print(f"Status code: {description['StatusCode']}")
        print("Stats:")
        print(
            f"\tLabeled entries: {description['DatasetStats']
            ['LabeledEntries']}")
        print(
            f"\tTotal entries: {description['DatasetStats']['TotalEntries']}")
        print(f"\tTotal labels: {description['DatasetStats']['TotalLabels']}")

    except ClientError as err:
        logger.exception("Couldn't describe dataset: %s",
                        err.response['Error']['Message'])
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """
```

```
parser.add_argument(
    "dataset_arn", help="The ARN of the dataset that you want to describe."
)

def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Describing dataset {args.dataset_arn}")

        # Describe the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        describe_dataset(rekognition_client, args.dataset_arn)

        print(f"Finished describing dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message=f"Problem describing dataset: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem describing dataset: {err}"
        logger.exception(error_message)
        print(error_message)

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

Java V2

- `dataset_arn` — ARN du jeu de données que vous souhaitez décrire.


```
/*
    Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
    SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

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.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStats;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class DescribeDataset {

    public static final Logger logger =
        Logger.getLogger(DescribeDataset.class.getName());

    public static void describeMyDataset(RekognitionClient rekClient, String
datasetArn) {

        try {

            DescribeDatasetRequest describeDatasetRequest =
                DescribeDatasetRequest.builder().datasetArn(datasetArn)
                    .build();

            DescribeDatasetResponse describeDatasetResponse =
                rekClient.describeDataset(describeDatasetRequest);

            DatasetDescription datasetDescription =
                describeDatasetResponse.datasetDescription();
            DatasetStats datasetStats = datasetDescription.datasetStats();

            System.out.println("ARN: " + datasetArn);
        }
    }
}
```

```
        System.out.println("Created: " +
datasetDescription.creationTimestamp().toString());
        System.out.println("Updated: " +
datasetDescription.lastUpdatedTimestamp().toString());
        System.out.println("Status: " +
datasetDescription.statusAsString());
        System.out.println("Message: " +
datasetDescription.statusMessage());
        System.out.println("Total Labels: " +
datasetStats.totalLabels().toString());
        System.out.println("Total entries: " +
datasetStats.totalEntries().toString());
        System.out.println("Entries with labels: " +
datasetStats.labeledEntries().toString());
        System.out.println("Entries with at least 1 error: " +
datasetStats.errorEntries().toString());

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        throw rekError;
    }

}

public static void main(String[] args) {

    final String USAGE = "\n" + "Usage: " + "<dataset_arn>\n\n" + "Where:\n"
        + "    dataset_arn - The ARN of the dataset that you want to
describe.\n\n";

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

    String datasetArn = args[0];

    try {

        // Get the Rekognition client.
        RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
```

```
        .region(Region.US_WEST_2)
        .build();

        // Describe the dataset.
        describeMyDataset(rekClient, datasetArn);

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        System.exit(1);
    }
}
}
```

Répertorier les entrées du jeu de données (SDK)

Vous pouvez utiliser l'`ListDatasetEntriesAPI` pour répertorier les lignes JSON pour chaque image d'un ensemble de données. Pour plus d'informations, veuillez consulter [Création d'un fichier manifeste](#).

Pour répertorier les entrées du jeu de données (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS CLI et les AWS SDK. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez l'exemple de code suivant pour répertorier les entrées d'un ensemble de données.

AWS CLI

Remplacez la valeur `dataset-arn` par l'ARN du jeu de données que vous souhaitez répertorier.

```
aws rekognition list-dataset-entries --dataset-arn dataset_arn \  
--profile custom-labels-access
```

Pour répertorier uniquement les lignes JSON contenant des erreurs, spécifiez `has-errors`.

```
aws rekognition list-dataset-entries --dataset-arn dataset_arn \  
  --has-errors \  
  --profile custom-labels-access
```

Python

Utilisez le code suivant. Fournissez les paramètres de ligne de commande suivants :

- `dataset_arn` — ARN du jeu de données que vous souhaitez répertorier.
- `show_errors_only` — Spécifiez `true` si vous souhaitez voir uniquement les erreurs. `false` sinon.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Shows how to list the entries in an Amazon Rekognition Custom Labels dataset.  
"""  
  
import argparse  
import logging  
import boto3  
  
from botocore.exceptions import ClientError  
  
logger = logging.getLogger(__name__)  
  
def list_dataset_entries(rek_client, dataset_arn, show_errors):  
    """  
    Lists the entries in an Amazon Rekognition Custom Labels dataset.  
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.  
    :param dataset_arn: The ARN of the dataet that you want to use.  
    """  
  
    try:  
        # List the entries.  
        logger.info("Listing dataset entries for the dataset %s.", dataset_arn)
```

```
finished = False
count = 0
next_token = ""
show_errors_only = False

if show_errors.lower() == "true":
    show_errors_only = True

while finished is False:

    response = rek_client.list_dataset_entries(
        DatasetArn=dataset_arn,
        HasErrors=show_errors_only,
        MaxResults=100,
        NextToken=next_token)

    count += len(response['DatasetEntries'])

    for entry in response['DatasetEntries']:
        print(entry)

    if 'NextToken' not in response:
        finished = True
        logger.info("No more entries. Total:%s", count)
    else:
        next_token = next_token = response['NextToken']
        logger.info("Getting more entries. Total so far :%s", count)

except ClientError as err:
    logger.exception(
        "Couldn't list dataset: %s",
        err.response['Error']['Message'])
    raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to list."
```

```
)

parser.add_argument(
    "show_errors_only", help="true if you want to see errors only. false
otherwise."
)

def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Listing entries for dataset {args.dataset_arn}")

        # List the dataset entries.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        list_dataset_entries(rekognition_client,
                             args.dataset_arn,
                             args.show_errors_only)

        print(f"Finished listing entries for dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message = f"Problem listing dataset: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem listing dataset: {err}"
        logger.exception(error_message)
        print(error_message)

if __name__ == "__main__":
```

```
main()
```

Java V2

Utilisez le code suivant. Fournissez les paramètres de ligne de commande suivants :

- `dataset_arn` — ARN du jeu de données que vous souhaitez répertorier.
- `show_errors_only` — Spécifiez `true` si vous souhaitez voir uniquement les erreurs. `false` sinon.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
 */

package com.example.rekognition;

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.ListDatasetEntriesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.paginators.ListDatasetEntriesIterable;

import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ListDatasetEntries {

    public static final Logger logger =
        Logger.getLogger(ListDatasetEntries.class.getName());

    public static void listMyDatasetEntries(RekognitionClient rekClient, String
        datasetArn, boolean showErrorsOnly)
        throws Exception, RekognitionException {

        try {
```

```
        logger.log(Level.INFO, "Listing dataset {0}", new Object[]
{ datasetArn });

        ListDatasetEntriesRequest listDatasetEntriesRequest =
ListDatasetEntriesRequest.builder()

.hasErrors(showErrorsOnly).datasetArn(datasetArn).maxResults(1).build();

        ListDatasetEntriesIterable datasetEntriesList = rekClient
                .listDatasetEntriesPaginator(listDatasetEntriesRequest);

        datasetEntriesList.stream().flatMap(r ->
r.datasetEntries().stream())
                .forEach(datasetEntry ->
System.out.println(datasetEntry.toString()));

    } catch (RekognitionException e) {
        logger.log(Level.SEVERE, "Could not update dataset: {0}",
e.getMessage());
        throw e;
    }

}

public static void main(String args[]) {

    boolean showErrorsOnly = false;
    String datasetArn = null;

    final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_arn>
<updates_file>\n\n" + "Where:\n"
        + "    dataset_arn - the ARN of the dataset that you want to
update.\n\n"
        + "    show_errors_only - true to show only errors. false
otherwise.\n\n";

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

    datasetArn = args[0];
    if (args[1].toLowerCase().equals("true")) {
```



```
        showErrorsOnly = true;
    }

    try {

        // Get the Rekognition client.
        RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

        // list the dataset entries.

        listMyDatasetEntries(rekClient, datasetArn, showErrorsOnly);

        System.out.println(String.format("Finished listing entries for :
%s", datasetArn));

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        System.exit(1);
    } catch (Exception rekError) {
        logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
        System.exit(1);
    }

}

}
```

Distribution d'un jeu de données d'entraînement (SDK)

Amazon Rekognition Custom Labels nécessite un jeu de données d'entraînement et un autre de test pour entraîner votre modèle.

Si vous utilisez l'API, vous pouvez l'utiliser pour distribuer 20 % du jeu de données d'apprentissage dans un jeu de données de test vide. [DistributeDatasetEntries](#) La distribution du jeu de données d'apprentissage peut s'avérer utile si vous ne disposez que d'un seul fichier manifeste. Utilisez le

fichier manifeste unique pour créer votre ensemble de données d'entraînement. Créez ensuite un jeu de données de test vide et `DistributeDatasetEntries` utilisez-le pour remplir le jeu de données de test.

Note

Si vous utilisez la console Amazon Rekognition Custom Labels et que vous commencez avec un seul projet de jeu de données, Amazon Rekognition Custom Labels divise (distribue) le jeu de données d'apprentissage, pendant l'entraînement, pour créer un jeu de données de test. 20 % des entrées du jeu de données d'apprentissage sont déplacées vers l'ensemble de données de test.

Pour distribuer un jeu de données d'entraînement (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS CLI et les AWS SDK. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Créez un projet. Pour plus d'informations, veuillez consulter [Création d'un projet Amazon Rekognition Custom Labels \(SDK\)](#).
3. Créez votre jeu de données d'entraînement. Pour savoir comment les jeux de données, veuillez consulter [Création de jeux de données d'entraînement et de test](#).
4. Créez un ensemble de données de test vide.
5. Utilisez l'exemple de code suivant pour distribuer 20 % des entrées du jeu de données d'entraînement dans l'ensemble de données de test. Vous pouvez obtenir les noms de ressources Amazon (ARN) pour les ensembles de données d'un projet en appelant [DescribeProjects](#). Pour obtenir un exemple de code, veuillez consulter [Décrire un projet \(SDK\)](#).

AWS CLI

Modifiez la valeur de `training_dataset_arn` et `test_dataset_arn` avec l'ARNs des jeux de données que vous souhaitez utiliser.

```
aws rekognition distribute-dataset-entries --datasets [{"Arn":  
  "training_dataset_arn"}, {"Arn": "test_dataset_arn"}] \  
  --profile custom-labels-access
```

Python

Utilisez le code suivant. Fournissez les paramètres de ligne de commande suivants :

- `training_dataset_arn` — ARN du jeu de données d'apprentissage à partir duquel vous distribuez les entrées.
- `test_dataset_arn` — ARN du jeu de données de test auquel vous distribuez des entrées.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def check_dataset_status(rek_client, dataset_arn):
    """
    Checks the current status of a dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The dataset that you want to check.
    :return: The dataset status and status message.
    """
    finished = False
    status = ""
    status_message = ""

    while finished is False:

        dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

        status = dataset['DatasetDescription']['Status']
        status_message = dataset['DatasetDescription']['StatusMessage']

        if status == "UPDATE_IN_PROGRESS":
```

```
        logger.info("Distributing dataset: %s ", dataset_arn)
        time.sleep(5)
        continue

    if status == "UPDATE_COMPLETE":
        logger.info(
            "Dataset distribution complete: %s : %s : %s",
            status, status_message, dataset_arn)
        finished = True
        continue

    if status == "UPDATE_FAILED":
        logger.exception(
            "Dataset distribution failed: %s : %s : %s",
            status, status_message, dataset_arn)
        finished = True
        break

    logger.exception(
        "Failed. Unexpected state for dataset distribution: %s : %s : %s",
        status, status_message, dataset_arn)
    finished = True
    status_message = "An unexpected error occurred while distributing the
dataset"
    break

    return status, status_message

def distribute_dataset_entries(rek_client, training_dataset_arn,
test_dataset_arn):
    """
    Distributes 20% of the supplied training dataset into the supplied test
    dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param training_dataset_arn: The ARN of the training dataset that you
    distribute entries from.
    :param test_dataset_arn: The ARN of the test dataset that you distribute
    entries to.
    """

    try:
        # List dataset labels.
```

```
        logger.info("Distributing training dataset entries (%s) into test
dataset (%s).",
                    training_dataset_arn,test_dataset_arn)

        datasets = json.loads(
            '[{"Arn" : "' + str(training_dataset_arn) + '"}, {"Arn" : "' +
str(test_dataset_arn) + '"}]')

        rek_client.distribute_dataset_entries(
            Datasets=datasets
        )

        training_dataset_status, training_dataset_status_message =
check_dataset_status(
            rek_client, training_dataset_arn)
        test_dataset_status, test_dataset_status_message = check_dataset_status(
            rek_client, test_dataset_arn)

        if training_dataset_status == 'UPDATE_COMPLETE' and test_dataset_status
== "UPDATE_COMPLETE":
            print("Distribution complete")
        else:
            print("Distribution failed:")
            print(
                f"\ttraining dataset: {training_dataset_status} :
{training_dataset_status_message}")
            print(
                f"\ttest dataset: {test_dataset_status} :
{test_dataset_status_message}")

    except ClientError as err:
        logger.exception(
            "Couldn't distribute dataset: %s",err.response['Error']['Message'] )
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
```

```
        "training_dataset_arn", help="The ARN of the training dataset that you
want to distribute from."
    )

    parser.add_argument(
        "test_dataset_arn", help="The ARN of the test dataset that you want to
distribute to."
    )

def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Distributing training dataset entries
({args.training_dataset_arn}) "\
            f"into test dataset ({args.test_dataset_arn}).")

        # Distribute the datasets.

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        distribute_dataset_entries(rekognition_client,
                                  args.training_dataset_arn,
                                  args.test_dataset_arn)

        print("Finished distributing datasets.")

    except ClientError as err:
        logger.exception("Problem distributing datasets: %s", err)
        print(f"Problem listing dataset labels: {err}")
    except Exception as err:
        logger.exception("Problem distributing datasets: %s", err)
        print(f"Problem distributing datasets: {err}")
```

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

Java V2

Utilisez le code suivant. Fournissez les paramètres de ligne de commande suivants :

- `training_dataset_arn` — ARN du jeu de données d'apprentissage à partir duquel vous distribuez les entrées.
- `test_dataset_arn` — ARN du jeu de données de test auquel vous distribuez des entrées.

```
/*  
    Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
    SPDX-License-Identifier: Apache-2.0  
*/  
package com.example.rekognition;  
  
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.DatasetDescription;  
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;  
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;  
import  
    software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;  
import software.amazon.awssdk.services.rekognition.model.DistributeDataset;  
import  
    software.amazon.awssdk.services.rekognition.model.DistributeDatasetEntriesRequest;  
import software.amazon.awssdk.services.rekognition.model.RekognitionException;  
  
import java.util.ArrayList;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class DistributeDatasetEntries {  
  
    public static final Logger logger =  
        Logger.getLogger(DistributeDatasetEntries.class.getName());
```

```
public static DatasetStatus checkDatasetStatus(RekognitionClient rekClient,
String datasetArn)
    throws Exception, RekognitionException {

    boolean distributed = false;
    DatasetStatus status = null;

    // Wait until distribution completes

    do {

        DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder().datasetArn(datasetArn)
            .build();
        DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

        DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

        status = datasetDescription.status();

        logger.log(Level.INFO, " dataset ARN: {0} ", datasetArn);

        switch (status) {

            case UPDATE_COMPLETE:
                logger.log(Level.INFO, "Dataset updated");
                distributed = true;
                break;

            case UPDATE_IN_PROGRESS:
                Thread.sleep(5000);
                break;

            case UPDATE_FAILED:
                String error = "Dataset distribution failed: " +
datasetDescription.statusAsString() + " "
                    + datasetDescription.statusMessage() + " " + datasetArn;
                logger.log(Level.SEVERE, error);
                break;

            default:
```



```
        String unexpectedError = "Unexpected distribution state: " +
datasetDescription.statusAsString() + " "
            + datasetDescription.statusMessage() + " " + datasetArn;
        logger.log(Level.SEVERE, unexpectedError);

    }

} while (distributed == false);

return status;

}

public static void distributeMyDatasetEntries(RekognitionClient rekClient,
String trainingDatasetArn,
    String testDatasetArn) throws Exception, RekognitionException {

    try {

        logger.log(Level.INFO, "Distributing {0} dataset to {1} ",
            new Object[] { trainingDatasetArn, testDatasetArn });

        DistributeDataset distributeTrainingDataset =
DistributeDataset.builder().arn(trainingDatasetArn).build();

        DistributeDataset distributeTestDataset =
DistributeDataset.builder().arn(testDatasetArn).build();

        ArrayList<DistributeDataset> datasets = new ArrayList();

        datasets.add(distributeTrainingDataset);
        datasets.add(distributeTestDataset);

        DistributeDatasetEntriesRequest distributeDatasetEntriesRequest =
DistributeDatasetEntriesRequest.builder()
            .datasets(datasets).build();

        rekClient.distributeDatasetEntries(distributeDatasetEntriesRequest);

        DatasetStatus trainingStatus = checkDatasetStatus(rekClient,
trainingDatasetArn);
        DatasetStatus testStatus = checkDatasetStatus(rekClient,
testDatasetArn);
```

```
        if (trainingStatus == DatasetStatus.UPDATE_COMPLETE && testStatus ==
DatasetStatus.UPDATE_COMPLETE) {
            logger.log(Level.INFO, "Successfully distributed dataset: {0}",
trainingDatasetArn);

            } else {

                throw new Exception("Failed to distribute dataset: " +
trainingDatasetArn);
            }

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not distribute dataset: {0}",
e.getMessage());
            throw e;
        }
    }

    public static void main(String[] args) {

        String trainingDatasetArn = null;
        String testDatasetArn = null;

        final String USAGE = "\n" + "Usage: " + "<training_dataset_arn>
<test_dataset_arn>\n\n" + "Where:\n"
            + "    training_dataset_arn - the ARN of the dataset that you
want to distribute from.\n\n"
            + "    test_dataset_arn - the ARN of the dataset that you want to
distribute to.\n\n";

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

        trainingDatasetArn = args[0];
        testDatasetArn = args[1];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
```

```
        .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
        .region(Region.US_WEST_2)
        .build();

        // Distribute the dataset
        distributeMyDatasetEntries(rekClient, trainingDatasetArn,
testDatasetArn);

        System.out.println("Datasets distributed.");

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        System.exit(1);
    } catch (Exception rekError) {
        logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
        System.exit(1);
    }
}
}
```

Supprimer un ensemble de données.

Vous pouvez supprimer les ensembles de données d'entraînement et de test d'un projet.

Rubriques

- [Suppression d'un ensemble de données \(console\)](#)
- [Suppression d'un ensemble de données Amazon Rekognition Custom Labels personnalisés Labels \(SDK\)](#)

Suppression d'un ensemble de données (console)

Procédez comme suit pour supprimer un ensemble de données. Ensuite, s'il reste un ensemble de données (train ou test) au projet, la page des détails du projet s'affiche. S'il ne reste aucun jeu de données dans le projet, la page Créer un jeu de données s'affiche.

Si vous supprimez le jeu de données d'entraînement, vous devez en créer un nouveau pour le projet avant de pouvoir entraîner un modèle. Pour plus d'informations, veuillez consulter [Création de jeux de données d'entraînement et de test avec des images](#).

Si vous supprimez le jeu de données de test, vous pouvez entraîner un modèle sans créer de nouveau jeu de données de test. Pendant l'entraînement, le jeu de données d'entraînement est divisé pour créer un nouvel ensemble de données de test pour le projet. La division du jeu de données d'entraînement réduit le nombre d'images disponibles pour l'entraînement. Pour préserver la qualité, nous vous recommandons de créer un nouvel ensemble de données de test avant d'entraîner un modèle. Pour plus d'informations, veuillez consulter [Ajouter un jeu de données à un projet](#).

Pour supprimer un ensemble de données.

1. Ouvrez la console Amazon Rekognition à l'[adresse https://console.aws.amazon.com/rekognition/](https://console.aws.amazon.com/rekognition/).
2. Dans le volet de gauche, choisissez Utiliser des étiquettes personnalisées. La page d'accueil des étiquettes personnalisées Amazon Rekognition s'affiche.
3. Dans le panneau de navigation de gauche, choisissez Projets. La vue Projets s'affiche.
4. Choisissez le projet qui contient le jeu de données que vous souhaitez supprimer.
5. Dans le panneau de navigation de gauche, sous le nom du projet, choisissez ensemble de données.
6. Choisissez Actions
7. Pour supprimer le jeu de données d'entraînement, choisissez Supprimer le jeu de données d'entraînement.
8. Pour supprimer le jeu de données de test, choisissez Supprimer l'ensemble de données de test.
9. Dans la boîte de dialogue Supprimer le train ou le jeu de données de test, saisissez delete pour confirmer que vous souhaitez supprimer le jeu de données.
10. Choisissez Supprimer le train ou l'ensemble de données de test pour supprimer l'ensemble de données.

Suppression d'un ensemble de données Amazon Rekognition Custom Labels personnalisés Labels (SDK)

Vous supprimez un jeu de données Amazon Rekognition Custom Labels en appelant [DeleteDataset](#) en spécifiant l'Amazon Resource Name (ARN) du jeu de données que vous

souhaitez supprimer. Pour obtenir les ARN des ensembles de données de formation et de test d'un projet, appelez [DescribeProjects](#). La réponse inclut un tableau d'[ProjectDescription](#) objets. Les ARN (`DatasetArn`) et les types de jeux de données (`DatasetType`) figurent dans la `Datasets` liste.

Si vous supprimez le jeu de données d'entraînement, vous devez en créer un nouveau pour le projet avant de pouvoir entraîner un modèle. Si vous supprimez le jeu de données de test, vous devez en créer un nouveau avant de pouvoir entraîner le modèle. Pour plus d'informations, veuillez consulter [Ajouter un jeu de données à un projet \(SDK\)](#).

Pour supprimer un jeu de données (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS CLI et les AWS SDK. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez le code suivant pour supprimer un ensemble de données.

AWS CLI

Modifiez la valeur de `dataset-arn` avec l'ARN du jeu de données que vous souhaitez supprimer.

```
aws rekognition delete-dataset --dataset-arn dataset-arn \  
--profile custom-labels-access
```

Python

Utilisez le code suivant. Fournissez les paramètres de ligne de commande suivants :

- `dataset_arn` — ARN du jeu de données que vous souhaitez supprimer.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Shows how to delete an Amazon Rekognition Custom Labels dataset.  
"""  
  
import argparse  
import logging  
import time  
import boto3
```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def delete_dataset(rek_client, dataset_arn):
    """
    Deletes an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that you want to delete.
    """

    try:
        # Delete the dataset,
        logger.info("Deleting dataset: %s", dataset_arn)

        rek_client.delete_dataset(DatasetArn=dataset_arn)

        deleted = False

        logger.info("waiting for dataset deletion %s", dataset_arn)

        # Dataset might not be deleted yet, so wait.
        while deleted is False:
            try:
                rek_client.describe_dataset(DatasetArn=dataset_arn)
                time.sleep(5)
            except ClientError as err:
                if err.response['Error']['Code'] == 'ResourceNotFoundException':
                    logger.info("dataset deleted: %s", dataset_arn)
                    deleted = True
                else:
                    raise

        logger.info("dataset deleted: %s", dataset_arn)

        return True

    except ClientError as err:
        logger.exception("Couldn't delete dataset - %s: %s",
                        dataset_arn, err.response['Error']['Message'])
        raise
```

```
def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to delete."
    )

def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Deleting dataset: {args.dataset_arn}")

        # Delete the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        delete_dataset(rekognition_client,
                       args.dataset_arn)

        print(f"Finished deleting dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message = f"Problem deleting dataset: {err}"
        logger.exception(error_message)
        print(error_message)

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

Java V2

Utilisez le code suivant. Fournissez les paramètres de ligne de commande suivants :

- `dataset_arn` — ARN du jeu de données que vous souhaitez supprimer.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import java.util.logging.Level;
import java.util.logging.Logger;

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.DeleteDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteDataset {

    public static final Logger logger =
        Logger.getLogger(DeleteDataset.class.getName());

    public static void deleteMyDataset(RekognitionClient rekClient, String
datasetArn) throws InterruptedException {

        try {

            logger.log(Level.INFO, "Deleting dataset: {0}", datasetArn);

            // Delete the dataset

            DeleteDatasetRequest deleteDatasetRequest =
DeleteDatasetRequest.builder().datasetArn(datasetArn).build();

            DeleteDatasetResponse response =
rekClient.deleteDataset(deleteDatasetRequest);
```



```
        // Wait until deletion finishes

        DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder().datasetArn(datasetArn)
            .build();

        Boolean deleted = false;

        do {

            try {

                rekClient.describeDataset(describeDatasetRequest);
                Thread.sleep(5000);
            } catch (RekognitionException e) {
                String errorCode = e.awsErrorDetails().errorCode();
                if (errorCode.equals("ResourceNotFoundException")) {
                    logger.log(Level.INFO, "Dataset deleted: {0}",
datasetArn);

                    deleted = true;
                } else {
                    logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());

                    throw e;
                }
            }

            } while (Boolean.FALSE.equals(deleted));

            logger.log(Level.INFO, "Dataset deleted: {0} ", datasetArn);

        } catch (

            RekognitionException e) {
                logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
                throw e;
            }
        }

        public static void main(String args[]) {
```

```
final String USAGE = "\n" + "Usage: " + "<dataset_arn>\n\n" + "Where:\n"
    + "  dataset_arn - The ARN of the dataset that you want to
delete.\n\n";

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

String datasetArn = args[0];

try {

    // Get the Rekognition client.
    RekognitionClient rekClient = RekognitionClient.builder()
        .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
        .region(Region.US_WEST_2)
        .build();

    // Delete the dataset
    deleteMyDataset(rekClient, datasetArn);

    System.out.println(String.format("Dataset deleted: %s",
datasetArn));

    rekClient.close();

} catch (RekognitionException rekError) {
    logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
    System.exit(1);
}

catch (InterruptedException intError) {
    logger.log(Level.SEVERE, "Exception while sleeping: {0}",
intError.getMessage());
    System.exit(1);
}

}
```

}

Gestion d'un modèle Amazon Rekognition Custom Labels

Un modèle d'étiquettes personnalisées Amazon Rekognition est un modèle mathématique qui prédit la présence d'objets, de scènes et de concepts dans les nouvelles images. Pour ce faire, il trouve des modèles dans les images utilisées pour entraîner le modèle. Cette section explique comment entraîner un modèle, évaluer ses performances et apporter des améliorations. Il vous montre également comment rendre un modèle disponible pour utilisation et comment supprimer un modèle lorsque vous n'en avez plus besoin.

Rubriques

- [Suppression d'un modèle Amazon Rekognition Custom Labels](#)
- [Marquer un modèle](#)
- [Décrire un modèle \(SDK\)](#)
- [Copie d'un modèle Amazon Rekognition Custom Labels \(SDK\)](#)

Suppression d'un modèle Amazon Rekognition Custom Labels

Vous pouvez supprimer un modèle à l'aide de la console Amazon Rekognition Custom Labels ou de l'[DeleteProjectVersion](#) API. Vous ne pouvez pas supprimer un modèle s'il est en cours d'exécution ou s'il est en cours d'entraînement. Pour arrêter un modèle en cours d'exécution, utilisez l'[StopProjectVersion](#) API. Pour plus d'informations, veuillez consulter [Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition \(kit SDK\)](#). Si un modèle est en cours d'entraînement, attendez qu'il soit terminé avant de le supprimer.

Un modèle supprimé ne peut pas être rétabli.

Rubriques

- [Suppression d'un modèle Amazon Rekognition Custom Labels \(console\)](#)
- [Suppression d'un modèle Amazon Rekognition Custom Labels \(SDK\)](#)

Suppression d'un modèle Amazon Rekognition Custom Labels (console)

La procédure suivante indique comment supprimer un modèle de la page de détails d'un projet. Vous pouvez également supprimer un modèle de la page détaillée d'un modèle.

Pour supprimer un modèle (console)

1. Ouvrez la console Amazon Rekognition à l'[adresse https://console.aws.amazon.com/rekognition/](https://console.aws.amazon.com/rekognition/).
2. Choisissez Utiliser des étiquettes personnalisées.
3. Sélectionnez Get started (Démarrer).
4. Dans le panneau de navigation de gauche, choisissez Projects (Projets).
5. Choisissez le projet qui contient le modèle que vous voulez supprimer. La page de Détails du projet s'ouvre.
6. Dans la section Modèles (Modèles), sélectionnez les modèles que vous voulez supprimer.

Note

Si le modèle ne peut pas être sélectionné, cela signifie qu'il est en cours d'exécution ou en cours d'entraînement et ne peut pas être supprimé. Vérifiez le champ Status et réessayez après avoir arrêté le modèle en cours d'exécution, ou attendez la fin de l'entraînement.

7. Choisissez Supprimer le modèle et la boîte de dialogue Supprimer le modèle s'affiche.
8. Saisissez supprimer pour confirmer la suppression.
9. Choisissez Delete (Supprimer) pour supprimer le modèle. La suppression du modèle peut prendre un certain temps.

Note

Si vous fermez la boîte de dialogue pendant la suppression du modèle, les modèles sont toujours supprimés.

Suppression d'un modèle Amazon Rekognition Custom Labels (SDK)

Pour supprimer un modèle d'étiquettes personnalisées Amazon Rekognition, appelez [DeleteProjectVersion](#) et fournissez l'Amazon Resource Name (ARN) du modèle que vous souhaitez supprimer. Vous pouvez obtenir l'ARN du modèle dans la section Utiliser votre modèle de la page de détails du modèle dans la console Amazon Rekognition Custom Labels. Vous pouvez également appeler [DescribeProjectVersions](#) et fournir les informations suivantes.

- L'ARN du projet (`ProjectArn`) auquel le modèle est associé.
- Le nom de version (`VersionNames`) du modèle.

L'ARN du modèle est le `ProjectVersionArn` champ de l'[ProjectVersionDescription](#) objet, à partir de la `DescribeProjectVersions` réponse.

Vous ne pouvez supprimer un modèle en cours d'exécution ou en cours d'entraînement. Pour déterminer si le modèle est en cours d'exécution ou s'entraîne, appelez [DescribeProjectVersions](#) et vérifiez le `Status` champ de l'[ProjectVersionDescription](#) objet du modèle. Pour arrêter un modèle en cours d'exécution, utilisez l'[StopProjectVersion](#) API. Pour plus d'informations, veuillez consulter [Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition \(kit SDK\)](#). Vous devez attendre la fin de l'entraînement d'un modèle avant de pouvoir le supprimer.

Pour supprimer un modèle (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS SDK AWS CLI et. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez le code suivant pour supprimer un modèle.

AWS CLI

Remplacez la valeur `project-version-arn` de par le nom du projet que vous souhaitez supprimer.

```
aws rekognition delete-project-version --project-version-arn model_arn \  
--profile custom-labels-access
```

Python

Fournissez les paramètres de ligne de commande suivants

- `project_arn`— l'ARN du projet qui contient le modèle que vous souhaitez supprimer.
- `model_arn`— l'ARN de la version du modèle que vous souhaitez supprimer.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""
```

Purpose

Shows how to delete an existing Amazon Rekognition Custom Labels model.

```
"""

import argparse
import logging
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def find_forward_slash(input_string, n):
    """
    Returns the location of '/' after n number of occurrences.
    :param input_string: The string you want to search
    : n: the occurrence that you want to find.
    """
    position = input_string.find('/')
    while position >= 0 and n > 1:
        position = input_string.find('/', position + 1)
        n -= 1
    return position

def delete_model(rek_client, project_arn, model_arn):
    """
    Deletes an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param model_arn: The ARN of the model version that you want to delete.
    """

    try:
        # Delete the model
        logger.info("Deleting dataset: {%s}", model_arn)

        rek_client.delete_project_version(ProjectVersionArn=model_arn)

        # Get the model version name
        start = find_forward_slash(model_arn, 3) + 1
        end = find_forward_slash(model_arn, 4)
        version_name = model_arn[start:end]
```

```
        deleted = False

        # model might not be deleted yet, so wait deletion finishes.
        while deleted is False:
            describe_response =
rek_client.describe_project_versions(ProjectArn=project_arn,
VersionNames=[version_name])
            if len(describe_response['ProjectVersionDescriptions']) == 0:
                deleted = True
            else:
                logger.info("Waiting for model deletion %s", model_arn)
                time.sleep(5)

        logger.info("model deleted: %s", model_arn)

        return True

    except ClientError as err:
        logger.exception("Couldn't delete model - %s: %s",
                        model_arn, err.response['Error']['Message'])
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains the model that
you want to delete."
    )

    parser.add_argument(
        "model_arn", help="The ARN of the model version that you want to
delete."
    )

def confirm_model_deletion(model_arn):
    """
```

```
Confirms deletion of the model. Returns True if delete entered.
:param model_arn: The ARN of the model that you want to delete.
"""
print(f"Are you sure you wany to delete model {model_arn} ?\n", model_arn)

start = input("Enter delete to delete your model: ")
if start == "delete":
    return True
else:
    return False

def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        if confirm_model_deletion(args.model_arn) is True:
            print(f"Deleting model: {args.model_arn}")

            # Delete the model.
            session = boto3.Session(profile_name='custom-labels-access')
            rekognition_client = session.client("rekognition")

            delete_model(rekognition_client,
                        args.project_arn,
                        args.model_arn)

            print(f"Finished deleting model: {args.model_arn}")
        else:
            print(f"Not deleting model {args.model_arn}")

    except ClientError as err:
        print(f"Problem deleting model: {err}")

if __name__ == "__main__":
```



```
main()
```

Java V2

- `project_arn`— l'ARN du projet qui contient le modèle que vous souhaitez supprimer.
- `model_arn`— l'ARN de la version du modèle que vous souhaitez supprimer.

```
//Copyright 2021 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)

import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.services.rekognition.RekognitionClient;

import
    software.amazon.awssdk.services.rekognition.model.DeleteProjectVersionRequest;
import
    software.amazon.awssdk.services.rekognition.model.DeleteProjectVersionResponse;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteModel {

    public static final Logger logger =
        Logger.getLogger(DeleteModel.class.getName());

    public static int findForwardSlash(String modelArn, int n) {

        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
        return start;
    }
}
```

```
}

    public static void deleteMyModel(RekognitionClient rekClient, String
projectArn, String modelArn)
        throws InterruptedException {

    try {

        logger.log(Level.INFO, "Deleting model: {0}", projectArn);

        // Delete the model

        DeleteProjectVersionRequest deleteProjectVersionRequest =
DeleteProjectVersionRequest.builder()
            .projectVersionArn(modelArn).build();

        DeleteProjectVersionResponse response =
            rekClient.deleteProjectVersion(deleteProjectVersionRequest);

        logger.log(Level.INFO, "Status: {0}", response.status());

        // Get the model version

        int start = findForwardSlash(modelArn, 3) + 1;
        int end = findForwardSlash(modelArn, 4);

        String versionName = modelArn.substring(start, end);

        Boolean deleted = false;

        DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
            .projectArn(projectArn).versionNames(versionName).build();

        // Wait until model is deleted.

        do {

            DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

            .describeProjectVersions(describeProjectVersionsRequest);
```

```
        if
(describeProjectVersionsResponse.projectVersionDescriptions().size()==0) {
            logger.log(Level.INFO, "Waiting for model deletion: {0}",
modelArn);
            Thread.sleep(5000);
        } else {
            deleted = true;
            logger.log(Level.INFO, "Model deleted: {0}", modelArn);
        }

    } while (Boolean.FALSE.equals(deleted));

    logger.log(Level.INFO, "Model deleted: {0}", modelArn);

} catch (

    RekognitionException e) {
        logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
        throw e;
    }

}

public static void main(String args[]) {

    final String USAGE = "\n" + "Usage: " + "<project_arn> <model_arn>\n\n"
+ "Where:\n"
        + "    project_arn - The ARN of the project that contains the
model that you want to delete.\n\n"
        + "    model_version - The ARN of the model that you want to
delete.\n\n";

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

    String projectArn = args[0];
    String modelVersion = args[1];

    try {

        RekognitionClient rekClient = RekognitionClient.builder().build();
```

```
        // Delete the model
        deleteMyModel(rekClient, projectArn, modelVersion);

        System.out.println(String.format("model deleted: %s",
modelVersion));

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        System.exit(1);
    }

    catch (InterruptedException intError) {
        logger.log(Level.SEVERE, "Exception while sleeping: {0}",
intError.getMessage());
        System.exit(1);
    }

}

}
```

Marquer un modèle

Vous pouvez identifier, organiser, rechercher et filtrer vos modèles Amazon Rekognition à l'aide de balises. Chaque balise est une étiquette composée d'une clé définie par l'utilisateur et d'une valeur. Par exemple, pour vous aider à déterminer la facturation de vos modèles, étiquetez vos modèles à l'aide d'une `CostCenter` clé et ajoutez le numéro de centre de coûts approprié comme valeur. Pour de plus amples informations, veuillez consulter [Balisage des ressources AWS](#).

Utilisez des balises pour :

- Suivez la facturation d'un modèle à l'aide de balises de répartition des coûts. Pour plus d'informations, consultez [Utilisation des balises de répartition des coûts](#).
- Contrôlez l'accès à un modèle à l'aide de la Identity and Access Management (IAM). Pour de plus amples informations, veuillez consulter [Contrôle de l'accès aux AWS ressources à l'aide des balises de ressources](#).

- Automatisez la gestion des modèles. Par exemple, vous pouvez exécuter des scripts de démarrage ou d'arrêt automatisés qui désactivent les modèles de développement en dehors des heures ouvrables afin de réduire les coûts. Pour plus d'informations, veuillez consulter [Exécution d'un modèle Étiquettes personnalisées Amazon Rekognition entraîné](#).

Vous pouvez étiqueter des modèles à l'aide de la console Amazon Rekognition ou à l'aide des kits deAWS développement logiciel (SDK).

Rubriques

- [Marquage des modèles \(console\)](#)
- [Affichage des balises de modèle](#)
- [Marquage des modèles \(SDK\)](#)

Marquage des modèles (console)

Vous pouvez utiliser la console Rekognition pour ajouter des balises aux modèles, afficher les balises associées à un modèle et supprimer des balises.

Ajout ou suppression de balises

Cette procédure explique comment ajouter des balises à un modèle existant ou en supprimer. Vous pouvez ajouter des balises à un nouveau modèle lorsqu'il est entraîné. Pour plus d'informations, veuillez consulter [Entraînement d'un modèle Amazon Rekognition Custom Labels Custom Labels](#).

Pour ajouter des balises à un modèle existant ou en supprimer à l'aide de la console

1. Ouvrez la console Amazon Rekognition à l'[adresse https://console.aws.amazon.com/rekognition/](https://console.aws.amazon.com/rekognition/).
2. Sélectionnez Get started (Démarrer).
3. Dans le panneau de navigation, choisissez Projects (Projets).
4. Sur la page Ressources des projets, choisissez le projet qui contient le modèle que vous souhaitez baliser.
5. Dans le panneau de navigation, sous le projet que vous avez précédemment choisi, choisissez Models (Modèles).
6. Dans la section Modèles (Modèles), choisissez le modèle auquel vous souhaitez ajouter une balise.

7. Sur la page de détails du modèle, choisissez l'onglet Tags (Balises).
8. Dans la section Tags (Balises) choisissez Manage tags (Gérer les balises).
9. Sur la page Gérer les balises, choisissez Ajouter une nouvelle étiquette.
10. Entrez une clé et une valeur.
 - a. Pour Key (Clé), entrez un nom pour la clé.
 - b. Pour Valeur, entrez une valeur.
11. Pour ajouter d'autres balises, répétez les étapes 9 et 10.
12. (Facultatif) Pour supprimer une balise, choisissez Supprimer en regard de la balise à supprimer. Si vous supprimez une étiquette précédemment enregistrée, elle est supprimée lorsque vous enregistrez vos modifications.
13. Choisissez Enregistrer les modifications pour enregistrer vos modifications.

Affichage des balises de modèle

Vous pouvez utiliser la console Amazon Rekognition pour afficher les balises attachées à un modèle.

Pour afficher les balises associées à tous les modèles d'un projet, vous devez utiliser le kit SDK AWS. Pour plus d'informations, veuillez consulter [Établissement de la liste des balises](#).

Pour afficher les balises attachées à un modèle

1. Ouvrez la console Amazon Rekognition à l'[adresse https://console.aws.amazon.com/rekognition/](https://console.aws.amazon.com/rekognition/).
2. Sélectionnez Get started (Démarrer).
3. Dans le panneau de navigation, choisissez Projects (Projets).
4. Sur la page Ressources des projets, choisissez le projet qui contient le modèle dont vous souhaitez afficher la balise.
5. Dans le panneau de navigation, sous le projet que vous avez précédemment choisi, choisissez Models (Modèles).
6. Dans la section Modèles (Modèles), choisissez le modèle dont vous souhaitez afficher l'étiquette.
7. Sur la page de détails du modèle, choisissez l'onglet Tags (Balises). Les balises sont affichées dans la section Étiquettes.

Marquage des modèles (SDK)

Vous pouvez utiliser leAWS SDK pour :

- Ajouter des balises à un nouveau modèle
- Ajouter des balises à un modèle existant
- Établissement de la liste des balises attachées à un modèle
- Supprimer les balises d'un modèle

Les balises desAWS CLI exemples suivants sont au format suivant.

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

Vous pouvez utiliser ce format.

```
--tags key1=value1,key2=value2
```

Si vous n'avez pas installé leAWS CLI, consultez [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).

Ajout d'balises à un nouveau modèle

Vous pouvez ajouter des balises à un modèle lorsque vous le créez à l'aide de cette [CreateProjectVersion](#)opération. Spécifiez une ou plusieurs balises dans le paramètre d'entrée duTags tableau.

```
aws rekognition create-project-version --project-arn project_arn \  
  --version-name version_name \  
  --output-config '{ "S3Location": { "Bucket": "output_bucket", "Prefix": "output  
folder" } }' \  
  --tags '{"key1":"value1","key2":"value2"}' \  
  --profile custom-labels-access
```

Pour plus d'informations sur la création et l'entraînement d'un modèle, consultez [Entraînement d'un modèle \(SDK\)](#).

Ajout de balises à un modèle existant

Pour ajouter une ou plusieurs identifications à un modèle existant, utilisez l'[TagResource](#)opération d'. Spécifiez l'Amazon Resource Name (ARN) (ResourceArn) du modèle et les balises (Tags) que vous souhaitez ajouter. L'exemple suivant montre comment ajouter deux balises.

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

Vous pouvez obtenir l'ARN d'un modèle en appelant [CreateProjectVersion](#).

Établissement de la liste des balises

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

```
aws rekognition list-tags-for-resource --resource-arn resource-arn \  
--profile custom-labels-access
```

La sortie affiche la liste des balises attachées au modèle.

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

Pour voir quels modèles d'un projet ont une étiquette spécifique, appelez `DescribeProjectVersions` pour obtenir la liste des modèles. Appelez ensuite `ListTagsForResource` chaque modèle dans le formulaire de réponse `DescribeProjectVersions`. Inspectez le formulaire de réponse `ListTagsForResource` pour voir si l'étiquette requise est présente.

L'exemple Python 3 suivant vous montre comment rechercher dans tous vos projets une clé et une valeur de balise spécifiques. La sortie inclut l'ARN du projet et l'ARN du modèle dans lequel une clé correspondante est trouvée.

Pour rechercher la valeur d'une balise

1. Enregistrez le code suivant dans un fichier nommé `find_tag.py`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0
```



```
"""
Purpose
Shows how to find a tag value that's associated with models within
your Amazon Rekognition Custom Labels projects.
"""

import logging
import argparse
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def find_tag_in_projects(rekognition_client, key, value):
    """
    Finds Amazon Rekognition Custom Label models tagged with the supplied key and
    key value.
    :param rekognition_client: An Amazon Rekognition boto3 client.
    :param key: The tag key to find.
    :param value: The value of the tag that you want to find.
    return: A list of matching model versions (and model projects) that were found.
    """
    try:

        found_tags = []
        found = False

        projects = rekognition_client.describe_projects()
        # Iterate through each project and models within a project.
        for project in projects["ProjectDescriptions"]:
            logger.info("Searching project: %s ...", project["ProjectArn"])

            models = rekognition_client.describe_project_versions(
                ProjectArn=(project["ProjectArn"])
            )

            for model in models["ProjectVersionDescriptions"]:
                logger.info("Searching model %s", model["ProjectVersionArn"])

                tags = rekognition_client.list_tags_for_resource(
                    ResourceArn=model["ProjectVersionArn"]
                )
```

```
        logger.info(
            "\tSearching model: %s for tag: %s value: %s.",
            model["ProjectVersionArn"],
            key,
            value,
        )
        # Check if tag exists.

        if key in tags["Tags"]:
            if tags["Tags"][key] == value:
                found = True
                logger.info(
                    "\t\tMATCH: Project: %s: model version %s",
                    project["ProjectArn"],
                    model["ProjectVersionArn"],
                )
                found_tags.append(
                    {
                        "Project": project["ProjectArn"],
                        "ModelVersion": model["ProjectVersionArn"],
                    }
                )

            if found is False:
                logger.info("No match for Tag %s with value %s.", key, value)
            return found_tags
    except ClientError as err:
        logger.info("Problem finding tags: %s. ", format(err))
        raise

def main():
    """
    Entry point for example.
    """
    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    # Set up command line arguments.
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)

    parser.add_argument("tag", help="The tag that you want to find.")
    parser.add_argument("value", help="The tag value that you want to find.")
```

```
args = parser.parse_args()
key = args.tag
value = args.value

print(f"Searching your models for tag: {key} with value: {value}.")

session = boto3.Session(profile_name='custom-labels-access')
rekognition_client = session.client("rekognition")

# Get tagged models for all projects.
tagged_models = find_tag_in_projects(rekognition_client, key, value)

print("Matched models\n-----")
if len(tagged_models) > 0:
    for model in tagged_models:
        print(
            "Project: {project}\nModel version: {version}\n".format(
                project=model["Project"], version=model["ModelVersion"]
            )
        )
else:
    print("No matches found.")

print("Done.")

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

2. À partir de l'invite de commande, entrez la commande suivante : Remplacez la *clé* et la *valeur* par le nom de la clé et la valeur de clé que vous souhaitez rechercher.

```
python find_tag.py key value
```

Suppression de balises d'un modèle

Pour supprimer une ou plusieurs identifications d'un modèle, utilisez l'[UntagResource](#) opération d'. 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"]' \  
--profile custom-labels-access
```

Vous pouvez spécifier `tag-keys` dans ce format.

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

Décrire un modèle (SDK)

Vous pouvez utiliser `DescribeProjectVersionsAPI` pour obtenir des informations sur une version d'un modèle. Si vous ne le spécifiez pas `VersionName`, `DescribeProjectVersions` renvoie les descriptions de toutes les versions du modèle du projet.

Pour décrire un modèle (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS SDK AWS CLI et. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez l'exemple de code suivant pour décrire une version d'un modèle.

AWS CLI

Remplacez la valeur `project-arn` de par l'ARN du projet que vous souhaitez décrire.
Remplacez la valeur `version-name` de par la version du modèle que vous souhaitez décrire.

```
aws rekognition describe-project-versions --project-arn project_arn \  
--version-names version_name \  
--profile custom-labels-access
```

Python

Utilisez le code suivant : Fournissez les paramètres de ligne de commande suivants :

- `project_arn` — L'ARN du modèle que vous voulez décrire.
- `model_version` — version du modèle que vous souhaitez décrire.

Par exemple : `python describe_model.py project_arn model_version`

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

```
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to describe an Amazon Rekognition Custom Labels model.
"""

import argparse
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def describe_model(rek_client, project_arn, version_name):
    """
    Describes an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that contains the model.
    :param version_name: The version name of the model that you want to
    describe.
    """

    try:
        # Describe the model
        logger.info("Describing model: %s for project %s",
                    version_name, project_arn)

        describe_response =
rek_client.describe_project_versions(ProjectArn=project_arn,

VersionNames=[version_name])
        for model in describe_response['ProjectVersionDescriptions']:
            print(f"Created: {str(model['CreationTimestamp'])} ")
            print(f"ARN: {str(model['ProjectVersionArn'])} ")
            if 'BillableTrainingTimeInSeconds' in model:
                print(
                    f"Billing training time (minutes):
{str(model['BillableTrainingTimeInSeconds']/60)} ")
            print("Evaluation results: ")
            if 'EvaluationResult' in model:
                evaluation_results = model['EvaluationResult']
                print(f"\tF1 score: {str(evaluation_results['F1Score'])}")
```

```
        print(
            f"\tSummary location: s3://{evaluation_results['Summary']
['S3Object']['Bucket']}/{evaluation_results['Summary']['S3Object']['Name']}")

        if 'ManifestSummary' in model:
            print(
                f"Manifest summary location: s3://{model['ManifestSummary']
['S3Object']['Bucket']}/{model['ManifestSummary']['S3Object']['Name']}")
            if 'OutputConfig' in model:
                print(
                    f"Training output location: s3://{model['OutputConfig']
['S3Bucket']}/{model['OutputConfig']['S3KeyPrefix']}")
                if 'MinInferenceUnits' in model:
                    print(
                        f"Minimum inference units:
{str(model['MinInferenceUnits'])}")
                    if 'MaxInferenceUnits' in model:
                        print(
                            f"Maximum Inference units:
{str(model['MaxInferenceUnits'])}")

                print("Status: " + model['Status'])
                print("Message: " + model['StatusMessage'])

    except ClientError as err:
        logger.exception(
            "Couldn't describe model: %s", err.response['Error']['Message'])
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which the model resides."
    )
    parser.add_argument(
        "version_name", help="The version of the model that you want to
describe."
    )
```

```
def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Describing model: {args.version_name} for project
{args.project_arn}.")

        # Describe the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        describe_model(rekognition_client, args.project_arn,
                        args.version_name)

        print(
            f"Finished describing model: {args.version_name} for project
{args.project_arn}.")

    except ClientError as err:
        error_message = f"Problem describing model: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem describing model: {err}"
        logger.exception(error_message)
        print(error_message)

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

Java V2

Utilisez le code suivant : Fournissez les paramètres de ligne de commande suivants :

- `project_arn` — L'ARN du modèle que vous voulez décrire.
- `model_version` — version du modèle que vous souhaitez décrire.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

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.DescribeProjectVersionsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.EvaluationResult;
import software.amazon.awssdk.services.rekognition.model.GroundTruthManifest;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
    software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class DescribeModel {

    public static final Logger logger =
        Logger.getLogger(DescribeModel.class.getName());

    public static void describeMyModel(RekognitionClient rekClient, String
        projectArn, String versionName) {

        try {
```



```
// If a single version name is supplied, build request argument

DescribeProjectVersionsRequest describeProjectVersionsRequest =
null;

    if (versionName == null) {
        describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder().projectArn(projectArn)
        .build();
    } else {
        describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder().projectArn(projectArn)
        .versionNames(versionName).build();
    }

DescribeProjectVersionsResponse describeProjectVersionsResponse =
rekClient
        .describeProjectVersions(describeProjectVersionsRequest);

    for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
        .projectVersionDescriptions()) {

        System.out.println("ARN: " +
projectVersionDescription.projectVersionArn());
        System.out.println("Status: " +
projectVersionDescription.statusAsString());
        System.out.println("Message: " +
projectVersionDescription.statusMessage());

        if (projectVersionDescription.billableTrainingTimeInSeconds() !=
null) {
            System.out.println(
                "Billable minutes: " +
(projectVersionDescription.billableTrainingTimeInSeconds() / 60));
        }

        if (projectVersionDescription.evaluationResult() != null) {
            EvaluationResult evaluationResult =
projectVersionDescription.evaluationResult();

            System.out.println("F1 Score: " +
evaluationResult.f1Score());
        }
    }
}
```

```
        System.out.println("Summary location: s3://" +
evaluationResult.summary().s3object().bucket() + "/"
        + evaluationResult.summary().s3object().name());
    }

    if (projectVersionDescription.manifestSummary() != null) {
        GroundTruthManifest manifestSummary =
projectVersionDescription.manifestSummary();
        System.out.println("Manifest summary location: s3://" +
manifestSummary.s3object().bucket() + "/"
        + manifestSummary.s3object().name());

    }

    if (projectVersionDescription.outputConfig() != null) {
        OutputConfig outputConfig =
projectVersionDescription.outputConfig();
        System.out.println(
            "Training output: s3://" + outputConfig.s3Bucket() +
"/" + outputConfig.s3KeyPrefix());
    }

    if (projectVersionDescription.minInferenceUnits() != null) {
        System.out.println("Min inference units: " +
projectVersionDescription.minInferenceUnits());
    }

    System.out.println();

}

} catch (RekognitionException rekError) {
    logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
    throw rekError;
}

}

public static void main(String args[]) {

    String projectArn = null;
    String versionName = null;
```

```
    final String USAGE = "\n" + "Usage: " + "<project_arn> <version_name>\n\n" + "Where:\n"
        + "    project_arn - The ARN of the project that contains the models you want to describe.\n\n"
        + "    version_name - (optional) The version name of the model that you want to describe. \n\n"
        + "                                If you don't specify a value, all model versions are described.\n\n";

    if (args.length > 2 || args.length == 0) {
        System.out.println(USAGE);
        System.exit(1);
    }

    projectArn = args[0];

    if (args.length == 2) {
        versionName = args[1];
    }

    try {

        // Get the Rekognition client.
        RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-labels-access"))
            .region(Region.US_WEST_2)
            .build();

        // Describe the model
        describeMyModel(rekClient, projectArn, versionName);

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}", rekError.getMessage());
        System.exit(1);
    }

}
```

Copie d'un modèle Amazon Rekognition Custom Labels (SDK)

Vous pouvez utiliser cette [CopyProjectVersion](#) opération pour copier une version de modèle Amazon Rekognition Custom Labels d'un projet Amazon Rekognition Custom Labels source vers un projet de destination. Le projet de destination peut se trouver dans un AWS compte différent ou dans le même AWS compte. Un scénario typique consiste à copier un modèle testé d'un AWS compte de développement vers un AWS compte de production.

Vous pouvez également entraîner le modèle dans le compte de destination à l'aide du jeu de données source. L'utilisation de [CopyProjectVersion](#) l'opération présente les avantages suivants.

- Le comportement du modèle est cohérent. L'entraînement des modèles n'est pas déterministe et il n'est pas garanti que deux modèles entraînés avec le même ensemble de données produisent les mêmes prédictions. La copie du modèle [CopyProjectVersion](#) permet de s'assurer que le comportement du modèle copié est cohérent avec le modèle source et que vous n'aurez pas besoin de retester le modèle.
- La formation des modèles n'est pas obligatoire. Cela vous permet d'économiser de l'argent car chaque formation réussie d'un modèle vous est facturée.

Pour copier un modèle vers un autre AWS compte, vous devez disposer d'un projet Amazon Rekognition Custom Labels dans le AWS compte de destination. Pour plus d'informations sur la création d'un projet, reportez-vous à [Crée un projet](#). Assurez-vous de créer le projet dans le AWS compte de destination.

Une [stratégie de projet](#) est une stratégie basée sur les ressources qui définit les autorisations de copie pour la version du modèle que vous souhaitez copier. Vous devrez utiliser une [politique de projet](#) lorsque le projet de destination se trouve dans un AWS compte différent de celui du projet source.

Il n'est pas nécessaire d'utiliser une [politique de projet](#) lorsque vous copiez des versions de modèles dans le même compte. Toutefois, vous pouvez choisir d'utiliser une [politique de projet](#) pour les projets intercomptes si vous souhaitez mieux contrôler ces ressources.

Vous associez la politique du projet au projet source en appelant l'[PutProjectPolicy](#) opération.

Vous ne pouvez pas l'utiliser [CopyProjectVersion](#) pour copier un modèle vers un projet situé dans une autre AWS région. De plus, vous ne pouvez pas copier un modèle à l'aide de la console

Amazon Rekognition Custom Labels. Dans ces cas, vous pouvez entraîner le modèle dans le projet de destination à l'aide des jeux de données utilisés pour entraîner le modèle source. Pour plus d'informations, veuillez consulter [Entraînement d'un modèle Amazon Rekognition Custom Labels Custom Labels](#).

Pour copier un modèle d'un projet source vers un projet de destination, procédez comme suit :

Pour copier un modèle

1. [Créez un document de politique de projet](#).
2. [Joignez la politique du projet au projet source](#).
3. [Copiez le modèle avec l'CopyProjectVersionopération](#).

Pour supprimer une politique de projet d'un projet, appelez [DeleteProjectPolicy](#). Pour obtenir la liste des politiques de projet associées à un projet, appelez [ListProjectPolicies](#).

Rubriques

- [Création d'un document de politique de projet](#)
- [Joindre une politique de projet \(SDK\)](#)
- [Copier un modèle \(SDK\)](#)
- [Répertorier les politiques du projet \(SDK\)](#)
- [Supprimer une politique de projet \(SDK\)](#)

Création d'un document de politique de projet

Rekognition Custom Labels utilise une politique basée sur les ressources, appelée politique de projet, pour gérer les autorisations de copie pour une version modèle. Une politique de projet est un document au format JSON.

Une politique de projet autorise ou refuse l'autorisation [principale](#) 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 sur un autreAWS compte. Cela est également vrai si le projet de destination se trouve dans le mêmeAWS compte que le projet source et que vous souhaitez restreindre l'accès à des versions de modèles spécifiques. Par exemple, il se peut que vous souhaitiez refuser les autorisations de copie pour un rôle IAM spécifique au sein d'unAWS compte.

L'exemple suivant permet au principal `arn:aws:iam::111111111111:role/Admin` de copier la version du modèle `arn:aws:rekognition:us-east-1:123456789012:project/my_project/version/test_1/1627045542080`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111111111111:role/Admin"
      },
      "Action": "rekognition:CopyProjectVersion",
      "Resource": "arn:aws:rekognition:us-east-1:111111111111:project/my_project/
version/test_1/1627045542080"
    }
  ]
}
```

Note

`Action`, `ResourcePrincipal`, et `Effect` sont des champs obligatoires dans un document de politique de projet.

Le seul pris en charge `action` est `rekognition:CopyProjectVersion`.

`NotActionNotResource`, et `NotPrincipal` sont des champs interdits qui ne doivent pas figurer dans le document de politique du projet.

Si vous ne spécifiez pas de politique de projet, un responsable du même AWS compte que le projet source peut toujours copier un modèle, à condition que le principal dispose d'une politique basée sur l'identité `AmazonRekognitionCustomLabelsFullAccess`, par exemple, qui autorise les appels `CopyProjectVersion`.

La procédure suivante crée un fichier de document de politique de projet que vous pouvez utiliser avec l'exemple Python dans [Joindre une politique de projet \(SDK\)](#). Si vous utilisez la `put-project-policy` AWS CLI commande, vous fournissez la politique du projet sous forme de chaîne JSON.

Pour créer un document de politique de projet

1. Dans un éditeur de texte, créez le document suivant. Modifiez les valeurs suivantes :

- Effet — Spécifiez si `ALLOW` vous souhaitez accorder l'autorisation de copie. Spécifiez `DENY` si vous souhaitez refuser l'autorisation de copie.
- Principal : au principal pour lequel vous souhaitez autoriser ou refuser l'accès aux versions du modèle que vous spécifiez `Resource`. Par exemple, vous pouvez spécifier le [principal du compte AWS](#) pour un autre `AWS` compte. Nous ne limitons pas les principes que vous pouvez utiliser. Pour de plus amples informations, veuillez consulter [Spécification d'un principal](#).
- Ressource : L'Amazon Resource Name (ARN) de la version du modèle pour laquelle vous souhaitez spécifier des autorisations de copie. Si vous souhaitez accorder des autorisations à toutes les versions de modèles du projet source, utilisez le format suivant `arn:aws:rekognition:region:account:project/source project/version/*`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "ALLOW or DENY",
      "Principal": {
        "AWS": "principal"
      },
      "Action": "rekognition:CopyProjectVersion",
      "Resource": "Model version ARN"
    }
  ]
}
```

2. Enregistrez la politique de projet sur votre ordinateur.
3. Joignez la politique du projet au projet source en suivant les instructions figurant sur [Joindre une politique de projet \(SDK\)](#).

Joindre une politique de projet (SDK)

Vous associez une politique de projet à un projet Amazon Rekognition Custom Labels en appelant l'[PutProjectpolicy](#) opération.

Associez plusieurs politiques de projet à un projet en `PutProjectPolicy` appelant chaque politique de projet que vous souhaitez ajouter. Vous pouvez associer jusqu'à cinq politiques de projet à un

projet. Si vous devez ajouter d'autres stratégies de projet, vous pouvez demander une augmentation de [limite](#).

Lorsque vous associez pour la première fois une politique de projet unique à un projet, ne spécifiez pas d'ID de révision dans le paramètre `PolicyRevisionId` d'entrée. Le formulaire de réponse `PutProjectPolicy` est un identifiant de révision pour la politique de projet qu'Amazon Rekognition Custom Labels crée pour vous. Vous pouvez utiliser l'ID de révision pour mettre à jour ou supprimer la dernière révision d'une politique de projet. Amazon Rekognition Custom Labels conserve uniquement la dernière révision de la politique de projet. Si vous essayez de mettre à jour ou de supprimer une version précédente d'une politique de projet, vous obtenez un `InvalidPolicyRevisionIdException` message d'erreur.

Pour mettre à jour une politique de projet existante, spécifiez l'ID de révision de la politique de projet dans le paramètre `PolicyRevisionId` d'entrée. Vous pouvez obtenir les ID de révision des politiques de projet d'un projet en appelant [ListProjectPolicies](#).

Après avoir associé une politique de projet à un projet source, vous pouvez copier le modèle du projet source vers le projet de destination. Pour plus d'informations, veuillez consulter [Copier un modèle \(SDK\)](#).

Pour supprimer une politique de projet d'un projet, appelez [DeleteProjectPolicy](#). Pour obtenir la liste des politiques de projet associées à un projet, appelez [ListProjectPolicies](#).

Pour associer une politique de projet à un projet (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS SDK AWS CLI et. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. [Créez un document de politique de projet](#).
3. Utilisez le code suivant pour joindre la politique du projet au projet, dans le AWS compte de confiance, qui contient la version du modèle que vous souhaitez copier. Pour obtenir l'ARN du projet, appelez [DescribeProjects](#). Pour obtenir la version du modèle ARN appelez [DescribeProjectVersions](#).

AWS CLI

Modifiez les valeurs suivantes :

- `project-arn` à l'ARN du projet source dans le AWS compte de confiance qui contient la version du modèle que vous souhaitez copier.

- `policy-name` nom de la politique que vous choisissez.
- `principal` Au principal auquel vous souhaitez autoriser ou refuser l'accès aux versions du modèle que vous spécifiez `Model version ARN`.
- `project-version-arn` à l'ARN de la version du modèle que vous voulez copier.

Si vous souhaitez mettre à jour une politique de projet existante, spécifiez le `policy-revision-id` paramètre et fournissez l'ID de révision de la politique de projet souhaitée.

```
aws rekognition put-project-policy \
  --project-arn project-arn \
  --policy-name policy-name \
  --policy-document '{ "Version":"2012-10-17", "Statement":
  [{ "Effect":"ALLOW or DENY", "Principal":{"AWS":"principal" },
  "Action":"rekognition:CopyProjectVersion", "Resource":"project-version-arn" ]}]' \
  --profile custom-labels-access
```

Python

Utilisez le code suivant : Fournissez les paramètres de ligne de commande suivants :

- `project_arn`— L'ARN du projet source auquel vous souhaitez associer la politique de projet.
- `policy_name`— Un nom de politique que vous choisissez.
- `project_policy`— Le fichier qui contient le document de politique du projet.
- `policy_revision_id`— (Facultatif). Si vous souhaitez mettre à jour une révision existante d'une politique de projet, spécifiez l'ID de révision de la politique de projet.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
```

Shows how to attach a project policy to an Amazon Rekognition Custom Labels project.

```
"""

import boto3
import argparse
import logging
import json
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def put_project_policy(rek_client, project_arn, policy_name,
policy_document_file, policy_revision_id=None):
    """
    Attaches a project policy to an Amazon Rekognition Custom Labels project.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param policy_name: A name for the project policy.
    :param project_arn: The Amazon Resource Name (ARN) of the source project
    that you want to attach the project policy to.
    :param policy_document_file: The JSON project policy document to
    attach to the source project.
    :param policy_revision_id: (Optional) The revision of an existing policy to
    update.
    Pass None to attach new policy.
    :return The revision ID for the project policy.
    """

    try:

        policy_document_json = ""
        response = None

        with open(policy_document_file, 'r') as policy_document:
            policy_document_json = json.dumps(json.load(policy_document))

        logger.info(
            "Attaching %s project_policy to project %s.",
            policy_name, project_arn)

        if policy_revision_id is None:
            response = rek_client.put_project_policy(ProjectArn=project_arn,
                                                    PolicyName=policy_name,
```

```
PolicyDocument=policy_document_json)

    else:
        response = rek_client.put_project_policy(ProjectArn=project_arn,
                                                PolicyName=policy_name,

PolicyDocument=policy_document_json,

PolicyRevisionId=policy_revision_id)

        new_revision_id = response['PolicyRevisionId']

        logger.info(
            "Finished creating project policy %s. Revision ID: %s",
            policy_name, new_revision_id)

        return new_revision_id

except ClientError as err:
    logger.exception(
        "Couldn't attach %s project policy to project %s: %s }",
        policy_name, project_arn, err.response['Error']['Message'] )
    raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The Amazon Resource Name (ARN) of the project "
        "that you want to attach the project policy to."
    )
    parser.add_argument(
        "policy_name", help="A name for the project policy."
    )

    parser.add_argument(
        "project_policy", help="The file containing the project policy JSON"
    )
```

```
parser.add_argument(
    "--policy_revision_id", help="The revision of an existing policy to
update. "
    "If you don't supply a value, a new project policy is created.",
    required=False
)

def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)

        args = parser.parse_args()

        print(f"Attaching policy to {args.project_arn}")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        # Attach a new policy or update an existing policy.

        response = put_project_policy(rekognition_client,
                                     args.project_arn,
                                     args.policy_name,
                                     args.project_policy,
                                     args.policy_revision_id)

        print(
            f"project policy {args.policy_name} attached to project
{args.project_arn}")
        print(f"Revision ID: {response}")

    except ClientError as err:
        print("Problem attaching project policy: %s", err)
```

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

Java V2

Utilisez le code suivant : Fournissez les paramètres de ligne de commande suivants :

- `project_arn`— L'ARN du projet source auquel vous souhaitez associer la politique de projet.
- `project_policy_name`— Un nom de politique que vous choisissez.
- `project_policy_document`— Le fichier qui contient le document de politique du projet.
- `project_policy_revision_id`— (Facultatif). Si vous souhaitez mettre à jour une révision existante d'une politique de projet, spécifiez l'ID de révision de la politique de projet.

```
/*  
    Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
    SPDX-License-Identifier: Apache-2.0  
*/  
  
package com.example.rekognition;  
  
import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
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.PutProjectPolicyRequest;  
import software.amazon.awssdk.services.rekognition.model.RekognitionException;  
  
public class PutProjectPolicy {  
  
    public static final Logger logger =  
        Logger.getLogger(PutProjectPolicy.class.getName());
```

```
public static void putMyProjectPolicy(RekognitionClient rekClient, String
projectArn, String projectPolicyName,
    String projectPolicyFileName, String projectPolicyRevisionId)
throws IOException {

    try {

        Path filePath = Path.of(projectPolicyFileName);

        String policyDocument = Files.readString(filePath);

        String[] logArguments = new String[] { projectPolicyFileName,
projectPolicyName };

        PutProjectPolicyRequest putProjectPolicyRequest = null;

        logger.log(Level.INFO, "Attaching Project policy: {0} to project:
{1}", logArguments);

        // Attach the project policy.

        if (projectPolicyRevisionId == null) {
            putProjectPolicyRequest =
PutProjectPolicyRequest.builder().projectArn(projectArn)
.policyName(projectPolicyName).policyDocument(policyDocument).build();
        } else {
            putProjectPolicyRequest =
PutProjectPolicyRequest.builder().projectArn(projectArn)
.policyName(projectPolicyName).policyRevisionId(projectPolicyRevisionId)
                .policyDocument(policyDocument)

                .build();
        }

        rekClient.putProjectPolicy(putProjectPolicyRequest);

        logger.log(Level.INFO, "Attached Project policy: {0} to project:
{1}", logArguments);
```

```
    } catch (
        RekognitionException e) {
        logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
        throw e;
    }
}

public static void main(String args[]) {

    final String USAGE = "\n" + "Usage: "
        + "<project_arn> <project_policy_name> <policy_document>
<project_policy_revision_id>\n\n" + "Where:\n"
        + "    project_arn - The ARN of the project that you want to
attach the project policy to.\n\n"
        + "    project_policy_name - A name for the project policy.\n\n"
        + "    project_policy_document - The file name of the project
policy.\n\n"
        + "    project_policy_revision_id - (Optional) The revision ID of
the project policy that you want to update.\n\n";

    if (args.length < 3 || args.length > 4) {
        System.out.println(USAGE);
        System.exit(1);
    }

    String projectArn = args[0];
    String projectPolicyName = args[1];
    String projectPolicyDocument = args[2];
    String projectPolicyRevisionId = null;

    if (args.length == 4) {
        projectPolicyRevisionId = args[3];
    }

    try {

        RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();
```

```
        // Attach the project policy.
        putMyProjectPolicy(rekClient, projectArn, projectPolicyName,
projectPolicyDocument,
            projectPolicyRevisionId);

        System.out.println(
            String.format("project policy %s: attached to project: %s",
projectPolicyName, projectArn));

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        System.exit(1);
    }

    catch (IOException intError) {
        logger.log(Level.SEVERE, "Exception while reading policy document:
{0}", intError.getMessage());
        System.exit(1);
    }

}

}
```

4. Copiez la version du modèle en suivant les instructions disponibles sur [Copier un modèle \(SDK\)](#).

Copier un modèle (SDK)

Vous pouvez utiliser l'`CopyProjectVersionAPI` pour copier une version de modèle d'un projet source vers un projet de destination. Le projet de destination peut figurer sur un AWS compte différent mais doit appartenir à la même AWS région. Si le projet de destination se trouve dans un autre AWS compte (ou si vous souhaitez accorder des autorisations spécifiques pour une version de modèle copiée dans un AWS compte), vous devez joindre une politique de projet au projet source. Pour plus d'informations, veuillez consulter [Création d'un document de politique de projet](#). L'`CopyProjectVersionAPI` doit avoir accès à votre compartiment Amazon S3.

Le modèle copié inclut les résultats d'apprentissage du modèle source, mais n'inclut pas les jeux de données source.

LeAWS compte source n'est pas propriétaire du modèle copié dans un compte de destination, sauf si vous définissez les autorisations appropriées.

Pour copier un modèle (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez lesAWS SDKAWS CLI et. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Joignez une politique de projet au projet source en suivant les instructions figurant sur [Joindre une politique de projet \(SDK\)](#).
3. Si vous copiez le modèle vers un autreAWS compte, assurez-vous que vous avez un projet dans leAWS compte de destination.
4. Utilisez le code suivant pour copier la version du modèle vers un projet de destination.

AWS CLI

Modifiez les valeurs suivantes :

- `source-project-arn` à l'ARN du projet source qui contient la version du modèle que vous souhaitez copier.
- `source-project-version-arn` à l'ARN de la version du modèle que vous souhaitez copier.
- `destination-project-arn` à l'ARN du projet de destination dans lequel vous souhaitez copier le modèle.
- `version-name` à un nom de version pour le modèle dans le projet de destination.
- `bucket` dans le compartiment S3 dans lequel vous souhaitez copier les résultats d'entraînement du modèle source.
- `folder` dans le dossier dans `bucket` lequel vous souhaitez copier les résultats d'entraînement du modèle source.
- (Facultatif) `kms-key-id` à l'ID de clé AWS Key Management Service pour le modèle.
- (Facultatif) `key` vers la clé de tag de votre choix.
- (Facultatif) `value` à la valeur de balise de votre choix.

```
aws rekognition copy-project-version \
```

```
--source-project-arn source-project-arn \  
--source-project-version-arn source-project-version-arn \  
--destination-project-arn destination-project-arn \  
--version-name version-name \  
--output-config '{"S3Bucket": "bucket", "S3KeyPrefix": "folder"}' \  
--kms-key-id arn:myKey \  
--tags '{"key": "key"}' \  
--profile custom-labels-access
```

Python

Utilisez le code suivant : Fournissez les paramètres de ligne de commande suivants :

- `source_project_arn`— l'ARN du projet source dans leAWS compte source qui contient la version du modèle que vous souhaitez copier.
- `source_project_version-arn`— l'ARN de la version du modèle dans leAWS compte source que vous souhaitez copier.
- `destination_project_arn`— l'ARN du projet de destination dans lequel vous souhaitez copier le modèle.
- `destination_version_name`— le nom de version du modèle dans le projet de destination.
- `training_results`— l'emplacement S3 vers lequel vous souhaitez copier les résultats d'entraînement pour la version du modèle source.
- (Facultatif)`kms_key_id` à l'ID de clé AWS Key Management Service pour le modèle.
- (Facultatif)`tag_name` vers la clé de tag de votre choix.
- (Facultatif)`tag_value` à la valeur de balise de votre choix.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
import argparse  
import logging  
import time  
import boto3  
from botocore.exceptions import ClientError  
  
logger = logging.getLogger(__name__)
```

```
def copy_model(
    rekognition_client, source_project_arn, source_project_version_arn,
    destination_project_arn, training_results, destination_version_name):
    """
    Copies a version of a Amazon Rekognition Custom Labels model.

    :param rekognition_client: A Boto3 Amazon Rekognition Custom Labels client.
    :param source_project_arn: The ARN of the source project that contains the
    model that you want to copy.
    :param source_project_version_arn: The ARN of the model version that you
    want
    to copy.
    :param destination_project_arn: The ARN of the project that you want to copy
    the model
    to.
    :param training_results: The Amazon S3 location where training results for
    the model
    should be stored.
    return: The model status and version.
    """
    try:
        logger.info("Copying model...%s from %s to %s ",
source_project_version_arn,
                    source_project_arn,
                    destination_project_arn)

        output_bucket, output_folder = training_results.replace(
            "s3://", "").split("/", 1)
        output_config = {"S3Bucket": output_bucket,
                        "S3KeyPrefix": output_folder}

        response = rekognition_client.copy_project_version(
            DestinationProjectArn=destination_project_arn,
            OutputConfig=output_config,
            SourceProjectArn=source_project_arn,
            SourceProjectVersionArn=source_project_version_arn,
            VersionName=destination_version_name
        )

        destination_model_arn = response["ProjectVersionArn"]

        logger.info("Destination model ARN: %s", destination_model_arn)
```

```
# Wait until training completes.
finished = False
status = "UNKNOWN"
while finished is False:
    model_description =
rekognition_client.describe_project_versions(ProjectArn=destination_project_arn,
                                             VersionNames=[destination_version_name])
    status = model_description["ProjectVersionDescriptions"][0]
["Status"]

    if status == "COPYING_IN_PROGRESS":
        logger.info("Model copying in progress...")
        time.sleep(60)
        continue

    if status == "COPYING_COMPLETED":
        logger.info("Model was successfully copied.")

    if status == "COPYING_FAILED":
        logger.info(
            "Model copy failed: %s ",
            model_description["ProjectVersionDescriptions"][0]
["StatusMessage"])

        finished = True
except ClientError:
    logger.exception("Couldn't copy model.")
    raise
else:
    return destination_model_arn, status

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "source_project_arn",
        help="The ARN of the project that contains the model that you want to
copy."
    )
```

```
parser.add_argument(
    "source_project_version_arn",
    help="The ARN of the model version that you want to copy."
)

parser.add_argument(
    "destination_project_arn",
    help="The ARN of the project which receives the copied model."
)

parser.add_argument(
    "destination_version_name",
    help="The version name for the model in the destination project."
)

parser.add_argument(
    "training_results",
    help="The S3 location in the destination account that receives the
training results for the copied model."
)

def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Copying model version {args.source_project_version_arn} to project
{args.destination_project_arn}")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        # Copy the model.

        model_arn, status = copy_model(rekognition_client,
```

```
        args.source_project_arn,  
        args.source_project_version_arn,  
        args.destination_project_arn,  
        args.training_results,  
        args.destination_version_name,  
    )  
  
    print(f"Finished copying model: {model_arn}")  
    print(f"Status: {status}")  
  
    except ClientError as err:  
        print(f"Problem copying model: {err}")  
  
if __name__ == "__main__":  
    main()
```

Java V2

Utilisez le code suivant : Fournissez les paramètres de ligne de commande suivants :

- `source_project_arn`— l'ARN du projet source dans leAWS compte source qui contient la version du modèle que vous souhaitez copier.
- `source_project_version_arn`— l'ARN de la version du modèle dans leAWS compte source que vous souhaitez copier.
- `destination_project_arn`— l'ARN du projet de destination dans lequel vous souhaitez copier le modèle.
- `destination_version_name`— le nom de version du modèle dans le projet de destination.
- `output_bucket`: le compartiment S3 dans lequel vous souhaitez copier les résultats d'entraînement pour la version du modèle source.
- `output_folder`: le dossier du S3 dans lequel vous souhaitez copier les résultats d'entraînement pour la version du modèle source.

```
/*  
    Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
    SPDX-License-Identifier: Apache-2.0  
*/
```

```
package com.example.rekognition;
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.CopyProjectVersionRequest;
import
    software.amazon.awssdk.services.rekognition.model.CopyProjectVersionResponse;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
    software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;

import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CopyModel {

    public static final Logger logger =
        Logger.getLogger(CopyModel.class.getName());

    public static ProjectVersionDescription copyMyModel(RekognitionClient
        rekClient,
        String sourceProjectArn,
        String sourceProjectVersionArn,
        String destinationProjectArn,
        String versionName,
        String outputBucket,
        String outputFolder) throws InterruptedException {

        try {

            OutputConfig outputConfig =
                OutputConfig.builder().s3Bucket(outputBucket).s3KeyPrefix(outputFolder).build();

            String[] logArguments = new String[] { versionName,
                sourceProjectArn, destinationProjectArn };

```

```
        logger.log(Level.INFO, "Copying model {0} for from project {1} to
project {2}", logArguments);

        CopyProjectVersionRequest copyProjectVersionRequest =
CopyProjectVersionRequest.builder()
            .sourceProjectArn(sourceProjectArn)
            .sourceProjectVersionArn(sourceProjectVersionArn)
            .versionName(versionName)
            .destinationProjectArn(destinationProjectArn)
            .outputConfig(outputConfig)
            .build();

        CopyProjectVersionResponse response =
rekClient.copyProjectVersion(copyProjectVersionRequest);

        logger.log(Level.INFO, "Destination model ARN: {0}",
response.projectVersionArn());
        logger.log(Level.INFO, "Copying model...");

        // wait until copying completes.

        boolean finished = false;

        ProjectVersionDescription copiedModel = null;

        while (Boolean.FALSE.equals(finished)) {
            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                .versionNames(versionName)
                .projectArn(destinationProjectArn)
                .build();

            DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient
                .describeProjectVersions(describeProjectVersionsRequest);

            for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                .projectVersionDescriptions()) {

                copiedModel = projectVersionDescription;

                switch (projectVersionDescription.status()) {
```



```
        case COPYING_IN_PROGRESS:
            logger.log(Level.INFO, "Copying model...");
            Thread.sleep(5000);
            continue;

        case COPYING_COMPLETED:
            finished = true;
            logger.log(Level.INFO, "Copying completed");
            break;

        case COPYING_FAILED:
            finished = true;
            logger.log(Level.INFO, "Copying failed...");
            break;

        default:
            finished = true;
            logger.log(Level.INFO, "Unexpected copy status %s",
                projectVersionDescription.statusAsString());
            break;
    }

}

}

}

        logger.log(Level.INFO, "Finished copying model {0} for from project
{1} to project {2}", logArguments);

        return copiedModel;

    } catch (RekognitionException e) {
        logger.log(Level.SEVERE, "Could not train model: {0}",
e.getMessage());
        throw e;
    }

}

public static void main(String args[]) {

    String sourceProjectArn = null;
```

```
String sourceProjectVersionArn = null;
String destinationProjectArn = null;
String versionName = null;
String bucket = null;
String location = null;

final String USAGE = "\n" + "Usage: "
    + "<source_project_arn> <source_project_version_arn>
<destination_project_arn> <version_name> <output_bucket> <output_folder>\n\n"
    + "Where:\n"
    + "  source_project_arn - The ARN of the project that contains
the model that you want to copy. \n\n"
    + "  source_project_version_arn - The ARN of the project that
contains the model that you want to copy. \n\n"
    + "  destination_project_arn - The ARN of the destination
project that you want to copy the model to. \n\n"
    + "  version_name - A version name for the copied model.\n\n"
    + "  output_bucket - The S3 bucket in which to place the
training output. \n\n"
    + "  output_folder - The folder within the bucket that the
training output is stored in. \n\n";

if (args.length != 6) {
    System.out.println(USAGE);
    System.exit(1);
}

sourceProjectArn = args[0];
sourceProjectVersionArn = args[1];
destinationProjectArn = args[2];
versionName = args[3];
bucket = args[4];
location = args[5];

try {

    // Get the Rekognition client.
    RekognitionClient rekClient = RekognitionClient.builder()
        .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
        .region(Region.US_WEST_2)
        .build();

    // Copy the model.
```

```
        ProjectVersionDescription copiedModel = copyMyModel(rekClient,
            sourceProjectArn,
            sourceProjectVersionArn,
            destinationProjectArn,
            versionName,
            bucket,
            location);

        System.out.println(String.format("Model copied: %s Status: %s",
            copiedModel.projectVersionArn(),
            copiedModel.statusMessage()));

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
            rekError.getMessage());
        System.exit(1);
    } catch (InterruptedException intError) {
        logger.log(Level.SEVERE, "Exception while sleeping: {0}",
            intError.getMessage());
        System.exit(1);
    }
}
}
```

Répertorier les politiques du projet (SDK)

Vous pouvez utiliser cette [ListProjectPolicies](#) opération pour répertorier les politiques de projet associées à un projet Amazon Rekognition Custom Labels.

Pour répertorier les politiques de projet attachées à un secret (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS SDK AWS CLI et. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez le code suivant pour répertorier les politiques du projet.

AWS CLI

Choisissez `project-arn` le nom de ressource Amazon du projet pour lequel vous souhaitez répertorier les politiques de projet attachées.

```
aws rekognition list-project-policies \  
  --project-arn project-arn \  
  --profile custom-labels-access
```

Python

Utilisez le code suivant : Fournissez les paramètres de ligne de commande suivants :

- `project_arn` : nom de ressource Amazon du projet pour lequel vous souhaitez répertorier les politiques de projet attachées.

Par exemple : `python list_project_policies.py project_arn`

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
  
"""  
Purpose  
Amazon Rekognition Custom Labels model example used in the service  
documentation:  
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-  
sdk.html  
Shows how to list the project policies in an Amazon Rekognition Custom Labels  
project.  
"""  
  
import argparse  
import logging  
import boto3  
from botocore.exceptions import ClientError  
  
logger = logging.getLogger(__name__)  
  
def display_project_policy(project_policy):
```

```
"""
Displays information about a Custom Labels project policy.
:param project_policy: The project policy (ProjectPolicy)
that you want to display information about.
"""

print(f"Policy name: {(project_policy['PolicyName'])}")
print(f"Project Arn: {project_policy['ProjectArn']}")
print(f"Document: {(project_policy['PolicyDocument'])}")
print(f"Revision ID: {(project_policy['PolicyRevisionId'])}")
print()

def list_project_policies(rek_client, project_arn):
    """
    Describes an Amazon Rekognition Custom Labels project, or all projects.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The Amazon Resource Name of the project you want to use.
    """

    try:

        max_results = 5
        pagination_token = ''
        finished = False

        logger.info("Listing project policies in: %s.", project_arn)
        print('Projects\n-----')
        while not finished:

            response = rek_client.list_project_policies(
                ProjectArn=project_arn, MaxResults=max_results,
                NextToken=pagination_token)

            for project in response['ProjectPolicies']:
                display_project_policy(project)

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

        logger.info("Finished listing project policies.")
```

```
except ClientError as err:
    logger.exception(
        "Couldn't list policies for - %s: %s",
        project_arn,err.response['Error']['Message'])
    raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The Amazon Resource Name of the project for which
you want to list project policies."
    )

def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Listing project policies in: {args.project_arn}")

        # List the project policies.

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        list_project_policies(rekognition_client,
                              args.project_arn)

    except ClientError as err:
        print(f"Problem list project_policies: {err}")
```

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

Java V2

Utilisez le code suivant : Fournissez les paramètres de ligne de commande suivants :

- `project_arn` — L'ARN du projet contenant les politiques de projet que vous souhaitez répertorier.

```
/*  
    Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
    SPDX-License-Identifier: Apache-2.0  
*/  
  
package com.example.rekognition;  
  
import java.util.logging.Level;  
import java.util.logging.Logger;  
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.ListProjectPoliciesRequest;  
import  
    software.amazon.awssdk.services.rekognition.model.ListProjectPoliciesResponse;  
import software.amazon.awssdk.services.rekognition.model.ProjectPolicy;  
import software.amazon.awssdk.services.rekognition.model.RekognitionException;  
  
public class ListProjectPolicies {  
  
    public static final Logger logger =  
        Logger.getLogger(ListProjectPolicies.class.getName());  
  
    public static void listMyProjectPolicies(RekognitionClient rekClient, String  
projectArn) {  
  
        try {  
  
            logger.log(Level.INFO, "Listing project policies for project: {0}",  
projectArn);
```

```
// List the project policies.

Boolean finished = false;
String nextToken = null;

while (Boolean.FALSE.equals(finished)) {

    ListProjectPoliciesRequest listProjectPoliciesRequest =
ListProjectPoliciesRequest.builder()
    .maxResults(5)
    .projectArn(projectArn)
    .nextToken(nextToken)
    .build();

    ListProjectPoliciesResponse response =
rekClient.listProjectPolicies(listProjectPoliciesRequest);

    for (ProjectPolicy projectPolicy : response.projectPolicies()) {

        System.out.println(String.format("Name: %s",
projectPolicy.policyName()));
        System.out.println(String.format("Revision ID: %s\n",
projectPolicy.policyRevisionId()));

    }

    nextToken = response.nextToken();

    if (nextToken == null) {
        finished = true;
    }

}

logger.log(Level.INFO, "Finished listing project policies for
project: {0}", projectArn);

} catch (

    RekognitionException e) {
    logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
```



```
        throw e;
    }
}

public static void main(String args[]) {

    final String USAGE = "\n" + "Usage: " + "<project_arn> \n\n" + "Where:
\n"
        + "    project_arn - The ARN of the project with the project
policies that you want to list.\n\n";
    ;

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

    String projectArn = args[0];

    try {

        RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

        // List the project policies.
        listMyProjectPolicies(rekClient, projectArn);

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        System.exit(1);
    }

}
}
```

Supprimer une politique de projet (SDK)

Vous pouvez utiliser cette [DeleteProjectPolicy](#) opération pour supprimer une révision d'une politique de projet existante dans un projet Amazon Rekognition Custom Labels. Si vous souhaitez supprimer toutes les révisions d'une politique de projet qui sont attachées à un projet, utilisez [ListProjectPolicies](#) pour obtenir les identifiants de révision de chaque politique de projet attachée au projet. Appelez ensuite `DeletePolicy` pour chaque nom de politique.

Pour supprimer une révision d'une politique de projet (SDK)

1. Si vous ne l'avez pas déjà fait, installez et configurez les AWS SDK AWS CLI et. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Utilisez le code suivant pour supprimer une politique de projet.

`DeletePolicy` prend `ProjectARN`, `PolicyName` et `PolicyRevisionId`.

`ProjectARN` et `PolicyName` sont obligatoires pour cette API. `PolicyRevisionId` est facultatif, mais peut être inclus à des fins de mises à jour atomiques.

AWS CLI

Modifiez les valeurs suivantes :

- `policy-name` au nom de la politique de projet que vous souhaitez supprimer.
- `policy-revision-id` à l'ID de révision de la politique de projet que vous souhaitez supprimer.
- `project-arn` au nom de ressource Amazon du projet qui contient la révision de la politique du projet que vous souhaitez supprimer.

```
aws rekognition delete-project-policy \  
  --policy-name policy-name \  
  --policy-revision-id policy-revision-id \  
  --project-arn project-arn \  
  --profile custom-labels-access
```

Python

Utilisez le code suivant : Fournissez les paramètres de ligne de commande suivants :

- `policy-name`— Choisissez le nom de la stratégie du projet que vous souhaitez supprimer.
- `project-arn`— Le nom de la ressource Amazon du projet qui contient la politique du projet que vous souhaitez supprimer.
- `policy-revision-id`— L'ID de révision de la politique de projet que vous souhaitez supprimer.

Par exemple `:python delete_project_policy.py policy_name project_arn policy_revision_id`

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to delete a revision of a project policy.
"""

import argparse
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def delete_project_policy(rekognition_client, policy_name, project_arn,
    policy_revision_id=None):
    """
    Deletes a project policy.

    :param rekognition_client: A Boto3 Amazon Rekognition client.
    :param policy_name: The name of the project policy that you want to delete.
    :param policy_revision_id: The revision ID for the project policy that you
    want to delete.
```

```
    :param project_arn: The Amazon Resource Name of the project that contains
the project policy
that you want to delete.
"""
    try:
        logger.info("Deleting project policy: %s", policy_name)

        if policy_revision_id is None:
            rekognition_client.delete_project_policy(
                PolicyName=policy_name,
                ProjectArn=project_arn)

        else:
            rekognition_client.delete_project_policy(
                PolicyName=policy_name,
                PolicyRevisionId=policy_revision_id,
                ProjectArn=project_arn)

        logger.info("Deleted project policy: %s", policy_name)
    except ClientError:
        logger.exception("Couldn't delete project policy.")
        raise

def confirm_project_policy_deletion(policy_name):
    """
    Confirms deletion of the project policy. Returns True if delete entered.
    :param model_arn: The ARN of the model that you want to delete.
    """
    print(
        f"Are you sure you want to delete project policy {policy_name} ?\n",
        policy_name)

    delete = input("Enter delete to delete your project policy: ")
    if delete == "delete":
        return True
    else:
        return False

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """
```

```
"""

parser.add_argument(
    "policy_name", help="The ARN of the project that contains the project
policy that you want to delete."
)

parser.add_argument(
    "project_arn", help="The ARN of the project project policy you want to
delete."
)

parser.add_argument(
    "--policy_revision_id", help="(Optional) The revision ID of the project
policy that you want to delete.",
    required=False
)

def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        if confirm_project_policy_deletion(args.policy_name) is True:
            print(f"Deleting project_policy: {args.policy_name}")

            session = boto3.Session(profile_name='custom-labels-access')
            rekognition_client = session.client("rekognition")

            # Delete the project policy.

            delete_project_policy(rekognition_client,
                                  args.policy_name,
                                  args.project_arn,
                                  args.policy_revision_id)
```

```
        print(f"Finished deleting project policy: {args.policy_name}")
    else:
        print(f"Not deleting project policy {args.policy_name}")
except ClientError as err:
    print(f"Couldn't delete project policy in {args.policy_name}: {err}")

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

Java V2

Utilisez le code suivant : Fournissez les paramètres de ligne de commande suivants :

- `policy-name`— Choisissez le nom de la stratégie du projet que vous souhaitez supprimer.
- `project-arn`— Le nom de la ressource Amazon du projet qui contient la politique du projet que vous souhaitez supprimer.
- `policy-revision-id`— L'ID de révision de la politique de projet que vous souhaitez supprimer.

```
/*
 Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.logging.Level;
import java.util.logging.Logger;

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.DeleteProjectPolicyRequest;

import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteProjectPolicy {
```

```
public static final Logger logger =
Logger.getLogger(DeleteProjectPolicy.class.getName());

public static void deleteMyProjectPolicy(RekognitionClient rekClient, String
projectArn,
    String projectPolicyName,
    String projectPolicyRevisionId)
    throws InterruptedException {

    try {
        String[] logArguments = new String[] { projectPolicyName,
projectPolicyRevisionId };

        logger.log(Level.INFO, "Deleting: Project policy: {0} revision:
{1}", logArguments);

        // Delete the project policy.

        DeleteProjectPolicyRequest deleteProjectPolicyRequest =
DeleteProjectPolicyRequest.builder()
            .policyName(projectPolicyName)
            .policyRevisionId(projectPolicyRevisionId)
            .projectArn(projectArn).build();

        rekClient.deleteProjectPolicy(deleteProjectPolicyRequest);

        logger.log(Level.INFO, "Deleted: Project policy: {0} revision: {1}",
logArguments);

    } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<project_arn>
<project_policy_name> <project_policy_revision_id>\n\n"
```

```
        + "Where:\n"
        + "    project_arn - The ARN of the project that has the project
policy that you want to delete.\n\n"
        + "    project_policy_name - The name of the project policy that
you want to delete.\n\n"
        + "    project_policy_revision_id - The revision of the project
policy that you want to delete.\n\n";

    if (args.length != 3) {
        System.out.println(USAGE);
        System.exit(1);
    }

    String projectArn = args[0];
    String projectPolicyName = args[1];
    String projectPolicyRevisionId = args[2];

    try {

        RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

        // Delete the project policy.
        deleteMyProjectPolicy(rekClient, projectArn, projectPolicyName,
projectPolicyRevisionId);

        System.out.println(String.format("project policy deleted: %s
revision: %s", projectPolicyName,
            projectPolicyRevisionId));

        rekClient.close();

    } catch (RekognitionException rekError) {
        logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
        System.exit(1);
    }

    catch (InterruptedException intError) {
        logger.log(Level.SEVERE, "Exception while sleeping: {0}",
intError.getMessage());
    }
}
```



```
        System.exit(1);  
    }  
}  
}
```

Exemples

Cette section contient des informations sur des exemples que vous pouvez utiliser avec les étiquettes personnalisées Amazon Rekognition.

Exemple	Description
Solution de feedback sur les modèles	Montre comment améliorer un modèle à l'aide de la vérification humaine pour créer un nouvel ensemble de données d'entraînement.
Démonstration des étiquettes personnalisées Amazon Rekognition	Démonstration d'une interface utilisateur qui affiche les résultats d'un appel à <code>DetectCustomLabels</code> .
Analyse vidéo	Montre comment vous pouvez utiliser <code>DetectCustomLabels</code> avec des images extraites d'une vidéo.
Analyse d'images à l'aide d'un AWS Lambda fonction	Montre comment vous pouvez utiliser <code>DetectCustomLabels</code> avec une fonction Lambda.
Création d'un fichier manifeste à partir d'un fichier CSV	Montre comment utiliser un fichier CSV pour créer un fichier manifeste adapté à la recherche objets, scènes et concepts associé à une image complète (classification).

Solution de feedback sur les modèles

La solution Model Feedback vous permet de donner votre avis sur les prévisions de votre modèle et d'apporter des améliorations en utilisant la vérification humaine. Selon le cas d'utilisation, vous pouvez réussir avec un jeu de données d'entraînement ne contenant que quelques images. Un ensemble d'apprentissage annoté plus important peut être nécessaire pour créer un modèle plus précis. À l'aide de la solution Model Feedback, vous pouvez créer un ensemble de données plus volumineux grâce à l'assistance relative aux modèles.

Pour installer et configurer la solution Model Feedback, voir [Solution de feedback sur les modèles](#).

Le flux de travail pour l'amélioration continue du modèle est le suivant :

1. Entraînez la première version de votre modèle (éventuellement avec un petit jeu de données d'entraînement).
2. Fournissez un jeu de données non annoté pour la solution Model Feedback.
3. La solution Model Feedback utilise le modèle actuel. Il lance des tâches de vérification humaine pour annoter un nouvel ensemble de données.
4. Sur la base des commentaires humains, la solution Model Feedback génère un fichier manifeste que vous utilisez pour créer un nouveau modèle.

Démonstration des étiquettes personnalisées Amazon Rekognition

La démonstration d'étiquettes personnalisées Amazon Rekognition présente une interface utilisateur qui analyse les images provenant de votre ordinateur local à l'aide du [DetectCustomLabelsAPI](#).

L'application affiche des informations sur les modèles d'étiquettes personnalisées Amazon Rekognition dans votre AWS compte. Après avoir sélectionné un modèle en cours d'exécution, vous pouvez analyser une image depuis votre ordinateur local. Si nécessaire, vous pouvez démarrer un modèle. Vous pouvez également arrêter un modèle en cours d'exécution. L'application montre une intégration avec d'autres services AWS tels qu'Amazon Cognito, Amazon S3 et Amazon CloudFront.

Pour plus d'informations, voir [Démo d'étiquettes personnalisées Amazon Rekognition](#).

Analyse vidéo

L'exemple suivant montre comment vous pouvez utiliser `DetectCustomLabels` avec des images extraites d'une vidéo. Le code a été testé avec des fichiers vidéo dans `movetmp4` format.

En utilisant `DetectCustomLabels` avec images capturées

1. Si ce n'est pas déjà fait, installez et configurez AWS CLI et le AWS SDK. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
2. Assurez-vous d'avoir `rekognition:DetectCustomLabels` et `AmazonS3ReadOnlyAccess` autorisations. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).

3. Utilisez l'exemple de code suivant. Modifiez la valeur de `videoFile` au nom d'un fichier vidéo. Modifiez la valeur de `projectVersionArn` à l'Amazon Resource Name (ARN) de votre modèle d'étiquettes personnalisées Amazon Rekognition.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to analyze a local video with an Amazon Rekognition Custom Labels model.
"""
import argparse
import logging
import json
import math
import cv2
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def analyze_video(rek_client, project_version_arn, video_file):
    """
    Analyzes a local video file with an Amazon Rekognition Custom Labels model.
    Creates a results JSON file based on the name of the supplied video file.
    :param rek_client: A Boto3 Amazon Rekognition client.
    :param project_version_arn: The ARN of the Custom Labels model that you want to
    use.
    :param video_file: The video file that you want to analyze.
    """

    custom_labels = []
    cap = cv2.VideoCapture(video_file)
    frame_rate = cap.get(5) # Frame rate.
    while cap.isOpened():
        frame_id = cap.get(1) # Current frame number.
        print(f"Processing frame id: {frame_id}")
        ret, frame = cap.read()
        if ret is not True:
            break
        if frame_id % math.floor(frame_rate) == 0:
```

```
    has_frame, image_bytes = cv2.imencode(".jpg", frame)

    if has_frame:
        response = rek_client.detect_custom_labels(
            Image={
                'Bytes': image_bytes.tobytes(),
            },
            ProjectVersionArn=project_version_arn
        )

        for elabel in response["CustomLabels"]:
            elabel["Timestamp"] = (frame_id/frame_rate)*1000
            custom_labels.append(elabel)

print(custom_labels)

with open(video_file + ".json", "w", encoding="utf-8") as f:
    f.write(json.dumps(custom_labels))

cap.release()

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_version_arn", help="The ARN of the model that you want to use."
    )

    parser.add_argument(
        "video_file", help="The local path to the video that you want to analyze."
    )

def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:
        # Get command line arguments.
```

```
parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
add_arguments(parser)
args = parser.parse_args()

session = boto3.Session(profile_name='custom-labels-access')
rekognition_client = session.client("rekognition")

analyze_video(rekognition_client,
              args.project_version_arn, args.video_file)

except ClientError as err:
    print(f"Couldn't analyze video: {err}")

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

Analyse d'images à l'aide d'unAWS Lambda fonction

AWS Lambda est un service informatique qui vous permet d'exécuter un code sans demander la mise en service ou la gestion des serveurs. Par exemple, vous pouvez analyser des images envoyées depuis une application mobile sans avoir à créer un serveur pour héberger le code de l'application. Les instructions suivantes montrent comment créer une fonction Lambda en Python qui appelle [DetectCustomLabels](#). La fonction analyse une image fournie et renvoie la liste des étiquettes présentes dans l'image. Les instructions incluent un exemple de code Python montrant comment appeler la fonction Lambda avec une image contenue dans un compartiment Amazon S3 ou une image fournie depuis un ordinateur local.

Rubriques

- [Étape 1 : Création d'unAWS Lambda fonction \(console\)](#)
- [Étape 2 : \(Facultatif\) Création d'une couche \(console\)](#)
- [Étape 3 : Ajouter du code Python \(console\)](#)
- [Étape 4 : essayez votre fonction Lambda](#)

Étape 1 : Création d'unAWS Lambda fonction (console)

Au cours de cette étape, vous créez un videAWS fonction et un rôle d'exécution IAM qui permet à votre fonction d'appeler `DetectCustomLabel` opération. Il permet également d'accéder au

compartiment Amazon S3 qui stocke les images à des fins d'analyse. Vous devez également spécifier des variables d'environnement pour les éléments suivants :

- Le modèle d'étiquettes personnalisées Amazon Rekognition que vous souhaitez que votre fonction Lambda utilise.
- La limite de confiance que vous souhaitez que le modèle utilise.

Vous ajouterez ensuite le code source et éventuellement une couche à la fonction Lambda.

Pour créer un AWS Lambda fonction (console)

1. Connectez-vous à la AWS Management Console et ouvrez la console AWS Lambda à l'adresse <https://console.aws.amazon.com/lambda/>.
2. Sélectionnez Créer une fonction. Pour plus d'informations, voir [Création d'une fonction Lambda avec la console](#).
3. Choisissez les options suivantes.
 - Choisissez Créer à partir de zéro.
 - Entrez une valeur pour Nom de la fonction.
 - Pour Temps d'exécution choisir Python 3.10.
4. Choisissez Créer une fonction pour créer le AWS Lambda fonction.
5. Sur la page des fonctions, choisissez Configuration onglet.
6. Sur le Variables d'environnement volet, choisissez Modifier.
7. Ajoutez les variables d'environnement suivantes. Pour chaque variable, choisissez Ajouter une variable d'environnement puis entrez la clé et la valeur de la variable.

Clé	Valeur
MODÈLE_ARN	Le nom de ressource Amazon (ARN) du modèle que vous souhaitez que votre fonction Lambda utilise. Vous pouvez obtenir l'ARN du modèle à partir du Modèle d'utilisation onglet de la page de détails du modèle dans la console Amazon Rekognition Custom Labels.

Clé	Valeur
CONFIANCE	La valeur minimale (0—100) de la confiance du modèle dans la prédiction d'une étiquette. La fonction Lambda ne renvoie pas d'étiquettes dont les valeurs de confiance sont inférieures à cette valeur.

8. Choisissez **Enregistrer** pour enregistrer les variables d'environnement.
9. Sur le **Autorisations** volet, En dessous **Nom** du rôle, choisissez le rôle d'exécution pour ouvrir le rôle dans la console IAM.
10. Dans le **Autorisations** onglet, choisissez **Ajouter des autorisations** puis **Créer une politique en ligne**.
11. Choisissez **JASON** remplacez la politique existante par la politique suivante.

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

12. Choisissez **Suivant**.
13. Dans **Détails** de la politique, entrez un nom pour la politique, tel que **DetectCustomLabels**-accès.
14. Choisissez **Create Policy** (Créer une politique).
15. Si vous stockez des images à des fins d'analyse dans un compartiment Amazon S3, répétez les étapes 10 à 14.
 - a. Pour l'étape 11, appliquez la politique suivante. Remplacer *chemin du bucket ou du dossier* avec le chemin du compartiment et du dossier Amazon S3 vers les images que vous souhaitez analyser.

```
{
  "Version": "2012-10-17",
  "Statement": [
```



```
{
  "Sid": "S3Access",
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::bucket/folder path/*"
}
```

- b. Pour l'étape 13, choisissez un autre nom de politique, tel que Accès au compartiment S3.

Étape 2 : (Facultatif) Création d'une couche (console)

Pour exécuter cet exemple, vous n'avez pas besoin de suivre cette étape.

Le `detectCustomLabels` l'opération est incluse dans l'environnement Lambda Python par défaut dans le cadre de `AWSSDK pour Python (Boto3)`. Si d'autres parties de votre fonction Lambda doivent être récentes AWS mises à jour de service qui ne figurent pas dans l'environnement Lambda Python par défaut, procédez comme suit pour ajouter la dernière version du SDK Boto3 en tant que couche à votre fonction.

Tout d'abord, vous créez une archive de fichiers `.zip` contenant le SDK Boto3. Vous créez ensuite une couche et ajoutez l'archive de fichiers `.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 commandes et entrez les commandes suivantes.

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

2. Notez le nom du fichier zip (`boto3-layer.zip`). Vous en aurez besoin à l'étape 6 de cette procédure.
3. Ouvrez la console AWS Lambda à l'adresse <https://console.aws.amazon.com/lambda/>.
4. Choisissez Layers dans le volet de navigation.
5. Sélectionnez Créer un calque.
6. Saisissez un Name (Nom) et une Description pour la règle.
7. Choisissez Charger un fichier .zip et choisissez 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 des environnements d'exécution compatibles, choisissez Python 3.9.
10. Choisissez Créez 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 dans [Étape 1 : Création d'un AWS Lambda fonction \(console\)](#).
14. Cliquez sur l'onglet Code.
15. Dans la Couches section, choisissez Ajouter une couche.
16. Choisissez Couches personnalisées.
17. Dans Couches personnalisées, choisissez le nom de la couche que vous avez saisi à l'étape 6.
18. Dans Version choisissez la version de la couche, qui doit être 1.
19. Choisissez Add (Ajouter).

Étape 3 : Ajouter du code Python (console)

Au cours de cette étape, vous allez ajouter du code Python à votre fonction Lambda à l'aide de l'éditeur de code de la console Lambda. Le code analyse une image fournie avec `DetectCustomLabel` et renvoie la liste des étiquettes présentes dans l'image. L'image fournie peut se trouver dans un compartiment Amazon S3 ou être fournie sous forme d'octets d'image codés en octet 64.

Pour ajouter du code Python (console)

1. Si vous n'êtes pas dans la console Lambda, procédez comme suit :
 - a. Ouvrez la console AWS Lambda à l'adresse <https://console.aws.amazon.com/lambda/>.
 - b. Ouvrez la fonction Lambda que vous avez créée dans [Étape 1 : Création d'un AWS Lambda fonction \(console\)](#).
2. Cliquez sur l'onglet Code.
3. Dans Source du code, remplacez le code dans `lambda_function.py` avec les éléments suivants :

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0
```

```
"""
Purpose
An AWS lambda function that analyzes images with an the Amazon Rekognition
Custom Labels model.
"""
import json
import base64
from os import environ
import logging
import boto3

from botocore.exceptions import ClientError

# Set up logging.
logger = logging.getLogger(__name__)

# Get the model ARN and confidence.
model_arn = environ['MODEL_ARN']
min_confidence = int(environ.get('CONFIDENCE', 50))

# Get the boto3 client.
rek_client = boto3.client('rekognition')

def lambda_handler(event, context):
    """
    Lambda handler function
    param: event: The event object for the Lambda function.
    param: context: The context object for the lambda function.
    return: The labels found in the image passed in the event
    object.
    """

    try:

        # Determine image source.
        if 'image' in event:
            # Decode the image
            image_bytes = event['image'].encode('utf-8')
            img_b64decoded = base64.b64decode(image_bytes)
            image = {'Bytes': img_b64decoded}
```

```
elif 'S3object' in event:
    image = {'S3object':
             {'Bucket': event['S3object']['Bucket'],
              'Name': event['S3object']['Name']}
            }

else:
    raise ValueError(
        'Invalid source. Only image base 64 encoded image bytes or S3object
are supported.')

# Analyze the image.
response = rek_client.detect_custom_labels(Image=image,
                                           MinConfidence=min_confidence,
                                           ProjectVersionArn=model_arn)

# Get the custom labels
labels = response['CustomLabels']

lambda_response = {
    "statusCode": 200,
    "body": json.dumps(labels)
}

except ClientError as err:
    error_message = f"Couldn't analyze image. " + \
        err.response['Error']['Message']

    lambda_response = {
        'statusCode': 400,
        'body': {
            "Error": 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
```

4. Choisissez `Déployer` pour déployer votre fonction Lambda.

Étape 4 : essayez votre fonction Lambda

Au cours de cette étape, vous utilisez du code Python sur votre ordinateur pour transmettre une image locale, ou une image d'un bucket Amazon S3, à votre fonction Lambda. Les images transmises depuis un ordinateur local doivent être inférieures à 6291456 octets. Si vos images sont plus grandes, chargez-les dans un compartiment Amazon S3 et appelez le script avec le chemin Amazon S3 vers l'image. Pour plus d'informations sur le chargement de fichiers image vers un compartiment Amazon S3, consultez [Chargement d'objets](#).

Assurez-vous d'exécuter le code dans la même AWS Région dans laquelle vous avez créé la fonction Lambda. Vous pouvez consulter la AWS Région de votre fonction Lambda dans la barre de navigation de la page de détails de la fonction dans [Console Lambda](#).

Si la fonction AWS Lambda renvoie une erreur de délai d'expiration, prolonge le délai d'expiration pour la fonction Lambda. Pour plus d'informations, voir [Configuration du délai d'expiration de la fonction \(console\)](#).

Pour plus d'informations sur l'appel d'une fonction Lambda à partir de votre code, voir [Invoquant AWS Lambda Fonctions](#).

Pour essayer votre fonction Lambda

1. Assurez-vous d'avoir `lambda:InvokeFunction` autorisation. Vous pouvez utiliser la politique suivante.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "InvokeLambda",
```

```
        "Effect": "Allow",
        "Action": "lambda:InvokeFunction",
        "Resource": "ARN for lambda function"
    }
]
}
```

Vous pouvez obtenir l'ARN de votre fonction Lambda à partir de la vue d'ensemble des fonctions dans [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. 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.

2. Installation et configuration AWSSDK pour Python. Pour plus d'informations, veuillez consulter [Étape 4 : Configurer AWS CLI et les kits AWS SDK](#).
3. [Démarrez le modèle](#) que vous avez spécifié à l'étape 7 de [Étape 1 : Création d'un AWS Lambda fonction \(console\)](#).
4. Enregistrez le code suivant dans un fichier nommé `client.py`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Test code for running the Amazon Rekognition Custom Labels Lambda
```

```
function example code.
"""

import argparse
import logging
import base64
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def analyze_image(function_name, image):
    """Analyzes an image with an AWS Lambda function.
    :param image: The image that you want to analyze.
    :return The status and classification result for
    the image analysis.
    """

    lambda_client = boto3.client('lambda')

    lambda_payload = {}

    if image.startswith('s3://'):
        logger.info("Analyzing image from S3 bucket: %s", image)
        bucket, key = image.replace("s3://", "").split("/", 1)
        s3_object = {
            'Bucket': bucket,
            'Name': key
        }
        lambda_payload = {"S3Object": s3_object}

    # Call the lambda function with the image.
    else:
        with open(image, 'rb') as image_file:
            logger.info("Analyzing local image image: %s ", image)
            image_bytes = image_file.read()
            data = base64.b64encode(image_bytes).decode("utf8")

            lambda_payload = {"image": data}
```

```
response = lambda_client.invoke(FunctionName=function_name,
                                Payload=json.dumps(lambda_payload))

return json.loads(response['Payload'].read().decode())

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "function", help="The name of the AWS Lambda function that you want " \
        "to use to analyze the image.")
    parser.add_argument(
        "image", help="The local image that you want to analyze.")

def main():
    """
    Entrypoint for script.
    """
    try:
        logging.basicConfig(level=logging.INFO,
                            format="%(levelname)s: %(message)s")

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Get analysis results.
        result = analyze_image(args.function, args.image)
        status = result['statusCode']

        if status == 200:
            labels = result['body']
            labels = json.loads(labels)
            print(f"There are {len(labels)} labels in the image.")
            for custom_label in labels:
                confidence = int(round(custom_label['Confidence'], 0))
                print(
                    f"Label: {custom_label['Name']}: Confidence: {confidence}%")
```



```
    else:
        print(f"Error: {result['statusCode']}")
        print(f"Message: {result['body']}")

    except ClientError as error:
        logging.error(error)
        print(error)

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

5. Exécutez le code. Pour l'argument de ligne de commande, indiquez le nom de la fonction Lambda et l'image que vous souhaitez analyser. Vous pouvez fournir un chemin vers une image locale ou le chemin S3 vers une image stockée dans un compartiment Amazon S3. Par exemple :

```
python client.py function_name s3://bucket/path/image.jpg
```

Si l'image se trouve dans un compartiment Amazon S3, assurez-vous qu'il s'agit du même compartiment que celui que vous avez spécifié à l'étape 15 de [Étape 1 : Création d'unAWS Lambdafonction \(console\)](#).

En cas de succès, la sortie est une liste des étiquettes trouvées dans l'image. Si aucune étiquette n'est renvoyée, envisagez de réduire la valeur de confiance que vous avez définie à l'étape 7 de [Étape 1 : Création d'unAWS Lambdafonction \(console\)](#).

6. Si vous avez terminé d'utiliser la fonction Lambda et que le modèle n'est pas utilisé par d'autres applications, [arrêter le modèle](#). N'oubliez pas de [démarrer le modèle](#) la prochaine fois que vous voudrez utiliser la fonction Lambda.

Sécurité

Vous pouvez sécuriser la gestion de vos projets, de vos modèles et de `DetectCustomLabel` opération que vos clients utilisent pour détecter les étiquettes personnalisées.

Pour plus d'informations sur la sécurisation d'Amazon Rekognition, consultez [Sécurité d'Amazon Rekognition](#).

Sécurisation des projets d'étiquettes personnalisées Amazon Rekognition

Vous pouvez sécuriser vos projets Amazon Rekognition Custom Labels en spécifiant les autorisations au niveau des ressources qui sont spécifiées dans les politiques basées sur l'identité. Pour plus d'informations, consultez [Stratégies basées sur l'identité et Stratégies basées sur une ressource](#).

Les ressources Amazon Rekognition Custom Labels que vous pouvez sécuriser sont les suivantes :

Ressource	Format du nom de ressource Amazon
Projet	<code>arn:aws:rekognition :*:*:project/<i>nom_projet</i> / date/heure</code>
Modèle	<code>arn:aws:rekognition :*:*:project/<i>nom_projet</i> / version/<i>nom</i>/date/heure</code>

L'exemple de politique suivant montre comment accorder une autorisation d'identité à :

- Décrivez tous les projets.
- Créez, démarrez, arrêtez et utilisez un modèle spécifique pour l'inférence.
- Créez un projet. Créez et décrivez un modèle spécifique.
- Refuser la création d'un projet spécifique.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "AllResources",
    "Effect": "Allow",
    "Action": "rekognition:DescribeProjects",
    "Resource": "*"
  },
  {
    "Sid": "SpecificProjectVersion",
    "Effect": "Allow",
    "Action": [
      "rekognition:StopProjectVersion",
      "rekognition:StartProjectVersion",
      "rekognition:DetectCustomLabels",
      "rekognition:CreateProjectVersion"
    ],
    "Resource": "arn:aws:rekognition:*:*:project/MyProject/version/MyVersion/*"
  },
  {
    "Sid": "SpecificProject",
    "Effect": "Allow",
    "Action": [
      "rekognition:CreateProject",
      "rekognition:DescribeProjectVersions",
      "rekognition:CreateProjectVersion"
    ],
    "Resource": "arn:aws:rekognition:*:*:project/MyProject/*"
  },
  {
    "Sid": "ExplicitDenyCreateProject",
    "Effect": "Deny",
    "Action": [
      "rekognition:CreateProject"
    ],
    "Resource": ["arn:aws:rekognition:*:*:project/SampleProject/*"]
  }
]
```

Sécurisation DetectCustomLabels

L'identité utilisée pour détecter les étiquettes personnalisées peut être différente de celle qui gère les modèles d'étiquettes personnalisées Amazon Rekognition.

Vous pouvez sécuriser l'accès d'une identité à `DetectCustomLabels` en appliquant une politique à l'identité. L'exemple suivant restreint l'accès à `DetectCustomLabels` uniquement et à un modèle spécifique. L'identité n'a accès à aucune autre opération Amazon Rekognition.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "rekognition:DetectCustomLabels"
      ],
      "Resource": "arn:aws:rekognition:*:*:project/MyProject/version/MyVersion/*"
    }
  ]
}
```

Politiques gérées par AWS

Nous fournissons le `AmazonRekognitionCustomLabelsFullAccess` AWS politique gérée que vous pouvez utiliser pour contrôler l'accès aux étiquettes personnalisées Amazon Rekognition. Pour plus d'informations, voir [Politique gérée par AWS : AmazonRekognitionCustomLabelsFullAccess](#).

Amazon Rekognition

Les sections suivantes fournissent des directives et des quotas lors de l'utilisation des étiquettes personnalisées Amazon Rekognition.

Régions prises en charge

Pour voir la liste des AWS régions dans lesquelles Amazon Rekognition Custom Labels est disponible, consultez [Régions et points de terminaison AWS](#) dans la Référence générale d'Amazon Web Services.

Quotas

Voici une liste de limites d'Amazon Rekognition Custom Labels. Pour plus d'informations sur les limites que vous pouvez modifier, consultez [Limites de service AWS](#). Pour modifier une limite, consultez [Création d'une demande](#).

Entraînement

- Les formats de fichier pris en charge sont les formats d'image PNG et JPEG.
- Le nombre maximum de jeux de données d'entraînement dans une version d'un modèle est de 1.
- La taille maximale du fichier manifeste du jeu de données est de 1 Go.
- Le nombre minimum d'étiquettes uniques par jeu de données Objects, Scenes, and Concepts (classification) est de 2.
- Le nombre minimum d'étiquettes uniques par jeu de données de localisation d'objets (détection) est de 1.
- Le nombre maximum d'étiquettes uniques par manifeste est de 250.
- Nombre minimal d'images par étiquette
- Le nombre maximum d'images par jeu de données de localisation d'objets (détection) est de 250 000.

La limite pour les AWS régions Asie-Pacifique (Mumbai) et Europe (Londres) est de 28 000 images.

- Le nombre maximum d'images par jeu de données Objects, Scenes, and Concepts (classification) est de 500 000. La valeur par défaut est 250 000. Pour demander une augmentation, voir [Create Case](#).

La limite pour les AWS régions Asie-Pacifique (Mumbai) et Europe (Londres) est de 28 000 images. Vous ne pouvez pas demander d'augmentation de limite

- Le nombre maximum d'étiquettes par image est de 50.
- Le nombre minimum de cadres de délimitation dans une image est de 0.
- Le nombre maximum de cadres de délimitation dans une image est de 50.
- La dimension d'image minimale du fichier image dans un compartiment Amazon S3 est de 64 pixels x 64 pixels.
- La dimension d'image maximale du fichier image dans un compartiment Amazon S3 est de 4 096 pixels x 4 096 pixels.
- La taille maximale d'image d'un compartiment Amazon S3 est de 15 Mo.
- Le rapport hauteur/largeur maximal de l'image est de 20:1.

Test

- Le nombre maximum de jeux de données de test dans une version d'un modèle est de 1.
- La taille maximale du fichier manifeste du jeu de données est de 1 Go.
- Le nombre minimum d'étiquettes uniques par jeu de données Objects, Scenes, and Concepts (classification) est de 2.
- Le nombre minimum d'étiquettes uniques par jeu de données de localisation d'objets (détection) est de 1.
- Le nombre maximum d'étiquettes uniques par jeu de données est de 250.
- Nombre minimal d'images par étiquette
- Nombre maximal d'images par étiquette
- Le nombre maximum d'images par jeu de données de localisation d'objets (détection) est de 250 000.

La limite pour les AWS régions Asie-Pacifique (Mumbai) et Europe (Londres) est de 7 000 images.

- Le nombre maximum d'images par jeu de données Objects, Scenes, and Concepts (classification) est de 500 000. La valeur par défaut est 250 000. Pour demander une augmentation, voir [Create Case](#).

La limite pour les AWS régions Asie-Pacifique (Mumbai) et Europe (Londres) est de 7 000 images. Vous ne pouvez pas demander d'augmentation de limite

- Le nombre minimum d'étiquettes par image et par manifeste est de 0.
- Le nombre maximum d'étiquettes par image et par manifeste est de 50.
- Le nombre minimum de cadres de délimitation dans une image par manifeste est de 0.
- Le nombre maximum de cadres de délimitation dans une image par manifeste est de 50.
- La dimension d'image minimale d'un fichier image dans un compartiment Amazon S3 est de 64 pixels x 64 pixels.
- La dimension d'image maximale d'un fichier image dans un compartiment Amazon S3 est de 4 096 pixels x 4 096 pixels.
- La taille maximale d'image d'un compartiment Amazon S3 est de 15 Mo.
- Les formats de fichier pris en charge sont les formats d'image PNG et JPEG.
- Le rapport hauteur/largeur maximal de l'image est de 20:1.

Détection

- La taille maximale des images transmises sous forme d'octets bruts est de 4 Mo.
- La taille maximale d'image d'un compartiment Amazon S3 est de 15 Mo.
- La dimension d'image minimale d'un fichier image d'entrée (stocké dans un compartiment Amazon S3 ou fourni sous forme d'octets d'image) est de 64 pixels x 64 pixels.
- La dimension d'image maximale d'un fichier image d'entrée (stocké dans un Amazon S3 ou fourni sous forme d'octets d'image) est de 4 096 pixels x 4 096 pixels.
- Les formats de fichier pris en charge sont les formats d'image PNG et JPEG.
- Le rapport hauteur/largeur maximal de l'image est de 20:1.

Copie de modèles

- Le nombre maximum de politiques de projet que vous pouvez [associer](#) à un projet est de 5.
- Le nombre maximal de tâches de copie simultanées dans une destination est de 5.

Référence Amazon Rekognition Custom Labels

L'API Amazon Rekognition Custom Labels est documentée dans le contenu de référence de l'API Amazon Rekognition. Voici une liste des opérations de l'API Amazon Rekognition Custom Labels avec des liens vers la rubrique de référence appropriée de l'API Amazon Rekognition. De plus, les liens de référence d'API contenus dans ce document renvoient à la rubrique de référence de l'API Amazon Rekognition Developer Guide appropriée. Pour plus d'informations sur l'utilisation de l'API, veuillez consulter

Cette section donne une vue d'ensemble du flux de travail pour former et utiliser un modèle d'étiquettes personnalisées Amazon Rekognition avec la console et leAWS SDK.

Note

Amazon Rekognition Custom Labels gère désormais les ensembles de données au sein d'un projet. Vous pouvez créer des ensembles de données pour vos projets à l'aide de la console

et duAWS SDK. Si vous avez déjà utilisé les étiquettes personnalisées Amazon Rekognition,

vos anciens ensembles de données devront peut-être être associés à un nouveau projet.

Pour plus d'informations, voir [Étape 6 \(Facultatif\) : Associer des jeux de données précédents à de nouveaux projets](#)

Rubriques

- [Choisissez votre type de modèle](#)
- [Création d'un modèle](#)
- [Améliorez votre modèle](#)
- [Démarrez votre modèle.](#)
- [Analyser une image](#)
- [Arrêter votre modèle](#)

Choisissez votre type de modèle

Choisissez votre type de modèle

Vous décidez d'abord du type de modèle que vous souhaitez former, en fonction de vos objectifs commerciaux. Par exemple, vous pouvez entraîner un mannequin à trouver votre logo sur les

réseaux sociaux, à identifier vos produits dans les rayons des magasins ou à classer les pièces de machines sur une chaîne de montage.

Les étiquettes personnalisées Amazon Rekognition peuvent entraîner les types de modèles suivants :

- [Trouvez des objets, des scènes et des concepts](#)
-

- [Trouver l'emplacement des objets](#)
-

- [Recherchez l'emplacement des marques](#)
-

Pour vous aider à choisir le type de modèle à entraîner, Amazon Rekognition Custom Labels propose des exemples de projets que vous pouvez utiliser. Pour plus d'informations, veuillez consulter [Démarrez avec les étiquettes personnalisées Amazon Rekognition](#).

Trouvez des objets, des scènes et des concepts

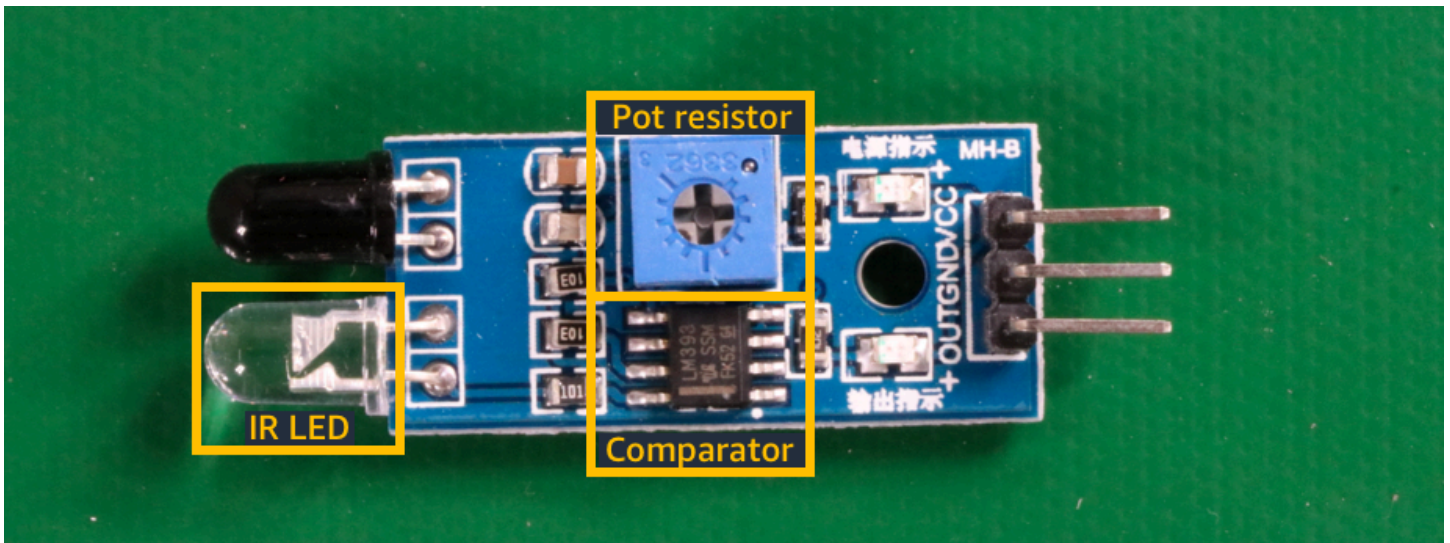
Le modèle prédit les classifications des objets, des scènes et des concepts associés à une image complète. Par exemple, vous pouvez entraîner un modèle qui détermine si une image contient une attraction touristique ou non. Pour un exemple de projet, voir [Image classification](#).



Vous pouvez également entraîner un modèle qui classe les images en plusieurs catégories. Par exemple, l'image précédente peut comporter des catégories telles que la couleur du ciel, le reflet ou le lac. Pour un exemple de projet, voir [Classification d'images à étiquettes multiples](#).

Trouver l'emplacement des objets

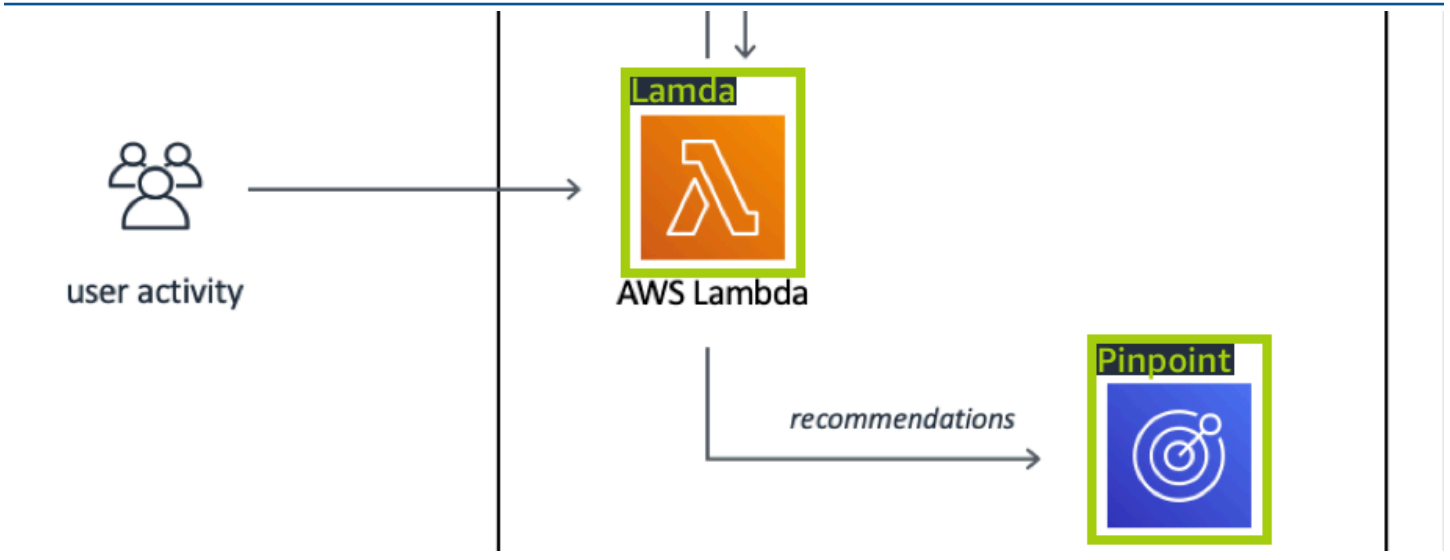
Le modèle prédit la position d'un objet sur une image. La prédiction inclut des informations sur le cadre de délimitation concernant l'emplacement de l'objet et une étiquette qui identifie l'objet dans le cadre de délimitation. Par exemple, l'image suivante montre des boîtes de délimitation situées autour de différentes parties d'une carte de circuit, telles qu'un comparateur ou une résistance en pot.



L'Localisation d'objet exemple de projet montre comment Amazon Rekognition Custom Labels utilise des zones de délimitation étiquetées pour former un modèle qui trouve l'emplacement des objets.

Recherchez l'emplacement des marques

Les étiquettes personnalisées Amazon Rekognition peuvent entraîner un modèle qui trouve l'emplacement des marques, telles que des logos, sur une image. La prédiction inclut des informations relatives au cadre de délimitation concernant l'emplacement de la marque et une étiquette identifiant l'objet à l'intérieur du cadre de délimitation. Pour un exemple de projet, voir Détection de marque.



Création d'un modèle

Pour créer un modèle, vous devez créer un projet, créer des ensembles de données d'entraînement et de test et entraîner le modèle.

Créer un projet

Un projet Amazon Rekognition Custom Labels est un regroupement des ressources nécessaires à la création et à la gestion d'un modèle. Un projet gère les éléments suivants :

- Ensembles de données : images et étiquettes d'images utilisées pour entraîner un modèle. Un projet possède un jeu de données d'entraînement et un jeu de données de test.
- Modèles : les logiciels que vous formez pour trouver les concepts, les scènes et les objets propres à votre entreprise. Vous pouvez avoir plusieurs versions d'un modèle dans un projet.

Nous vous recommandons d'utiliser un projet pour un seul cas d'utilisation, par exemple pour rechercher des pièces de circuit imprimé sur une carte de circuit imprimé.

Vous pouvez créer un projet à l'aide de la console Amazon Rekognition Custom Labels et de l'[CreateProjectAPI](#). Pour plus d'informations, veuillez consulter [Crée un projet..](#)

Création de jeux de données de formation et de test

Un jeu de données est un ensemble d'images et d'étiquettes qui décrivent ces images. Dans votre projet, vous créez un jeu de données d'entraînement et un autre de test qu'Amazon Rekognition Custom Labels utilise afin d'entraîner et de tester votre modèle.

Une étiquette identifie un objet, une scène, un concept ou un cadre de délimitation autour d'un objet dans une image. Les étiquettes sont attribuées à une image entière (au niveau de l'image) ou à un cadre de délimitation qui entoure un objet sur une image.

Important

La façon dont vous étiquetez les images dans vos ensembles de données détermine le type de modèle créé par Amazon Rekognition Custom Labels. Par exemple, pour entraîner un

modèle qui trouve des objets, des scènes et des concepts, vous attribuez des étiquettes de

~~niveau d'image aux images de vos ensembles de données d'entraînement et de test. Pour~~
plus d'informations, veuillez consulter [Utilisation des jeux de données.](#)

Les images doivent être au format PNG et JPEG, et vous devez suivre les recommandations relatives aux images d'entrée. Pour plus d'informations, veuillez consulter [Préparation des images](#).

Création d'ensembles de données d'entraînement et de test (console)

Vous pouvez démarrer un projet avec un seul jeu de données ou avec des ensembles de données d'entraînement et de test distincts. Si vous commencez avec un seul jeu de données, Amazon Rekognition Custom Labels divise votre jeu de données pendant l'entraînement afin de créer un jeu de données d'apprentissage (80 %) et un jeu de données de test (20 %) pour votre projet.

Commencez avec un seul ensemble de données si vous souhaitez qu'Amazon Rekognition Custom Labels décide quelles images seront utilisées pour la formation et les tests. Pour un contrôle total de la formation, des tests et du réglage des performances, nous vous recommandons de démarrer votre projet avec des ensembles de données d'entraînement et de test distincts.

Pour créer les jeux de données pour un projet, vous devez importer les images de l'une des manières suivantes :

- Importez des images depuis votre ordinateur local.
- Importer des images à partir d'un compartiment S3. Les étiquettes personnalisées Amazon Rekognition peuvent étiqueter les images en utilisant les noms des dossiers qui contiennent les images.
- Importez un fichier manifeste Amazon SageMaker Ground Truth.
- Copier un jeu de données Amazon Rekognition Custom Labels existant.

Pour plus d'informations, veuillez consulter [Création de jeux de données d'entraînement et de test avec des images](#).

Selon l'endroit d'où vous importez vos images, celles-ci peuvent ne pas être étiquetées. Par exemple, les images importées depuis un ordinateur local ne sont pas étiquetées. Les images importées depuis un fichier manifeste Amazon SageMaker Ground Truth sont étiquetées. Vous pouvez utiliser la console Amazon Rekognition Custom Labels pour ajouter, modifier et attribuer des étiquettes. Pour plus d'informations, veuillez consulter [Étiquetage des images](#).

Pour créer vos ensembles de données d'entraînement et de test à l'aide de la console, consultez [Création de jeux de données d'entraînement et de test avec des images](#). Pour consulter un didacticiel qui inclut la création de jeux de données d'entraînement et de test, consultez [Tutoriel : Classification d'images](#).

Création d'ensembles de données de formation et de test (SDK)

Pour créer vos ensembles de données d'entraînement et de test, vous utilisez l'`CreateDatasetAPI`. Vous pouvez créer un jeu de données en utilisant un fichier manifeste au format Amazon Sagemaker ou en copiant un jeu de données Amazon Rekognition Custom Labels existant. Pour plus d'informations, voir [Création de jeux de données d'entraînement et de test \(kit SDK\)](#). Si nécessaire, vous pouvez créer votre propre fichier manifeste. Pour plus d'informations, veuillez consulter [the section called "Création d'un fichier manifeste"](#).

Entraînez votre modèle

Entraînez votre modèle à l'aide du jeu de données d'entraînement. Une nouvelle version d'un modèle est créée chaque fois qu'il est entraîné. Pendant la formation, les étiquettes personnalisées Amazon Rekognition testent les performances de votre modèle entraîné. Vous pouvez utiliser les résultats pour évaluer et améliorer votre modèle. La formation prend un certain temps. Seule une formation de modèle réussie vous est facturée. Pour plus d'informations, veuillez consulter [Entraînement d'un modèle Amazon Rekognition Custom Labels Custom Labels](#). Si la formation du modèle échoue, Amazon Rekognition Custom Labels fournit des informations de débogage que vous pouvez utiliser. Pour plus d'informations, veuillez consulter [Déboguer un modèle d'entraînement ayant échoué](#).

Entraînez votre modèle (console)

Pour entraîner votre modèle à l'aide de la console, voir [Entraînement d'un modèle \(console\)](#).

Entraînement d'un modèle (SDK)

Vous entraînez un modèle d'étiquettes personnalisées Amazon Rekognition en appelant `CreateProjectVersion`. Pour plus d'informations, veuillez consulter [Entraînement d'un modèle \(SDK\)](#).

Améliorez votre modèle

Pendant les tests, Amazon Rekognition Custom Labels crée des mesures d'évaluation que vous pouvez utiliser pour améliorer votre modèle entraîné.

Évaluer votre modèle

Évaluez les performances de votre modèle à l'aide des mesures de performance créées lors des tests. Les indicateurs de performance, tels que F1, précision et rappel, vous permettent de comprendre les performances de votre modèle entraîné et de décider si vous êtes prêt à l'utiliser en production. Pour plus d'informations, veuillez consulter [Métriques pour évaluer votre modèle](#).

Évaluer un modèle (console)

Pour consulter les métriques de performances, consultez [Accès aux mesures d'évaluation \(console\)](#).

Évaluer un modèle (SDK)

Pour obtenir des mesures de performance, vous appelez `DescribeProjectVersions` pour obtenir les résultats des tests. Pour plus d'informations, veuillez consulter [Accès aux métriques d'évaluation \(SDK\) d'Amazon Rekognition Custom Labels](#). Les résultats des tests incluent des mesures non disponibles dans la console, telles qu'une matrice de confusion pour les résultats de classification.

Les résultats des tests sont renvoyés dans les formats suivants :

- Score F1 : valeur unique représentant les performances globales de précision et de rappel du modèle. Pour plus d'informations, veuillez consulter [F1](#).
 - Emplacement du fichier de synthèse : le résumé des tests inclut des mesures d'évaluation agrégées pour l'ensemble de données de test et des mesures pour chaque étiquette individuelle. `DescribeProjectVersions` renvoie le compartiment S3 et l'emplacement du dossier du fichier récapitulatif. Pour plus d'informations, veuillez consulter [Fichier récapitulatif](#).
 - Emplacement de l'instantané du manifeste d'évaluation : l'instantané contient des détails sur les résultats des tests, notamment les cotes de confiance et les résultats des tests de classification binaire, tels que les faux positifs. `DescribeProjectVersions` renvoie le compartiment S3 et l'emplacement du dossier des fichiers de capture instantanée. Pour plus d'informations, veuillez consulter [Aperçu du manifeste d'évaluation](#).
-

Améliorez votre modèle

Si des améliorations sont nécessaires, vous pouvez ajouter d'autres images d'apprentissage ou améliorer l'étiquetage des jeux de données. Pour plus d'informations, veuillez consulter [Amélioration d'un modèle Amazon Rekognition Custom Labels](#). Vous pouvez également donner votre avis sur les prévisions faites par votre modèle et les utiliser pour apporter des améliorations à votre modèle. Pour plus d'informations, veuillez consulter [Solution de feedback sur les modèles](#).

Améliorez votre modèle (console)

Pour ajouter des images à un jeu de données, consultez [Ajouter d'autres images à un jeu de données](#). Pour ajouter ou modifier des étiquettes, voir [the section called "Étiquetage des images"](#).

Pour réentraîner votre modèle, voir [Entraînement d'un modèle \(console\)](#).

Améliorez votre modèle (SDK)

Pour ajouter des images à un jeu de données ou modifier l'étiquetage d'une image, utilisez l'`UpdateDatasetEntriesAPI`. `UpdateDatasetEntries` met à jour ou ajoute des lignes JSON à un fichier manifeste. Chaque ligne JSON contient des informations relatives à une seule image, telles que les étiquettes attribuées ou les informations relatives au cadre de délimitation. Pour plus d'informations, veuillez consulter [Ajouter d'autres images \(SDK\)](#). Pour afficher les entrées d'un jeu de données, utilisez l'`ListDatasetEntriesAPI`.

Pour réentraîner votre modèle, voir [Entraînement d'un modèle \(SDK\)](#).

Démarrez votre modèle.

Avant de pouvoir utiliser votre modèle, vous devez le démarrer à l'aide de la console Amazon Rekognition Custom Labels ou de l'`StartProjectVersionAPI`. La durée de fonctionnement de votre modèle vous est facturée. Pour plus d'informations, veuillez consulter [Exécution d'un modèle entraîné](#).

Démarrez votre modèle (console).

Pour démarrer votre modèle à l'aide de la console, consultez [Démarrage d'un modèle Étiquettes personnalisées Amazon Rekognition \(Console\)](#).

Démarrez votre modèle.

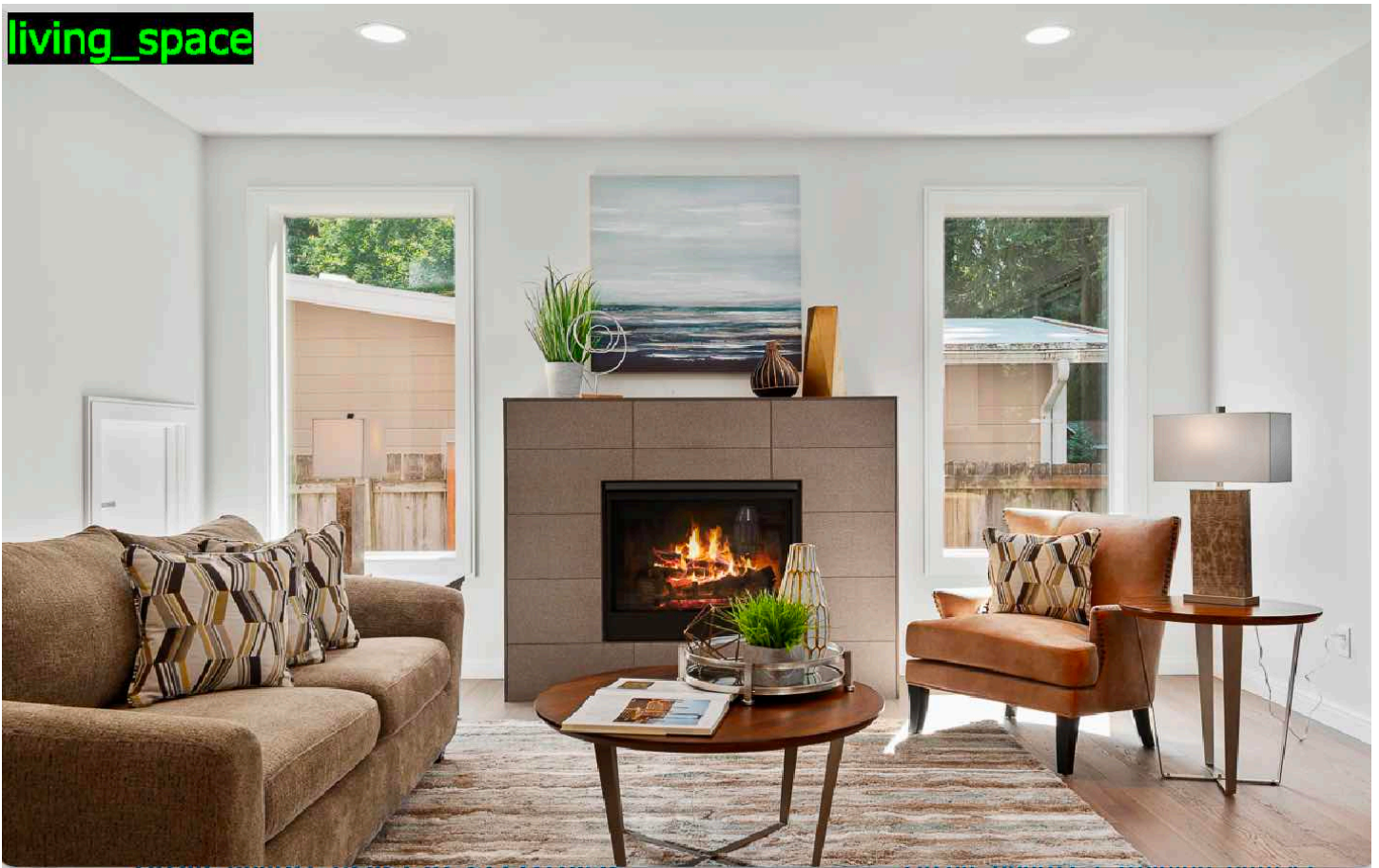
Vous commencez à appeler votre modèle `StartProjectVersion`. Pour plus d'informations, veuillez consulter [Démarrage d'un modèle Étiquettes personnalisées Amazon Rekognition \(kit SDK\)](#).

Analyser une image

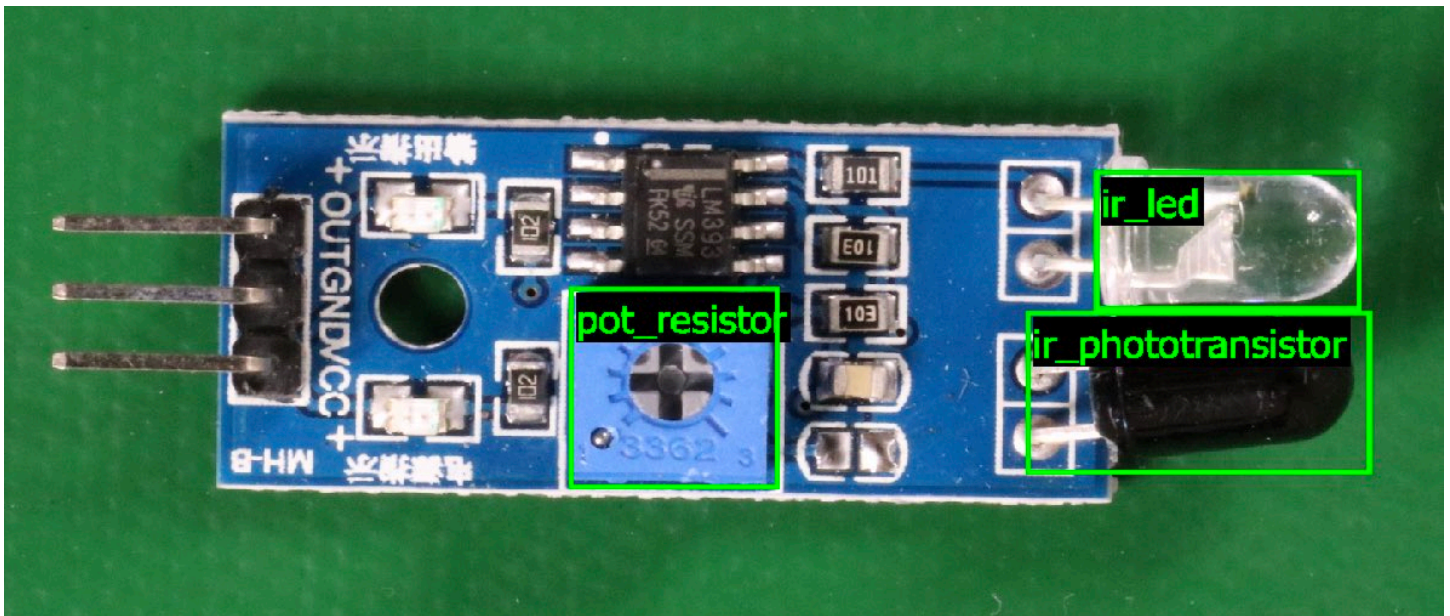
Pour analyser une image avec votre modèle, vous utilisez l'`DetectCustomLabelsAPI`. Vous pouvez spécifier une image locale ou une image stockée dans un compartiment S3. L'opération nécessite également l'Amazon Resource Name (ARN) du modèle que vous souhaitez utiliser.

Si votre modèle trouve des objets, des scènes et des concepts, la réponse inclut une liste des étiquettes au niveau de l'image présentes dans l'image. Par exemple, l'image suivante affiche les étiquettes au niveau de l'image trouvées à l'aide du projet d'exemple Rooms.

living_space



Si le modèle trouve l'emplacement des objets, la réponse inclut la liste des zones de délimitation étiquetées présentes dans l'image. Un cadre de délimitation représente l'emplacement d'un objet sur une image. Vous pouvez utiliser les informations du cadre de délimitation pour dessiner un cadre de délimitation autour d'un objet. Par exemple, l'image suivante montre des boîtes de délimitation autour de pièces de circuits imprimés trouvées à l'aide du projet d'exemple de cartes de circuits imprimés.



Pour plus d'informations, veuillez consulter [Analyse d'une image avec un modèle entraîné](#).

Arrêter votre modèle

Vous êtes facturé pour la durée pendant laquelle votre modèle fonctionne. Si vous n'utilisez plus votre modèle, arrêtez-le à l'aide de la console Amazon Rekognition Custom Labels ou de l'`StopProjectVersionAPI`. Pour plus d'informations, veuillez consulter [Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition](#).

Arrêtez votre modèle (console)

Pour arrêter un modèle en cours d'exécution à l'aide de la console, voir [Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition \(console\)](#).

Arrêtez votre modèle (SDK)

Pour arrêter un modèle en cours d'exécution, appelez `StopProjectVersion`. Pour plus d'informations, veuillez consulter [Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition \(kit SDK\)](#).

Entraînement votre modèle

Projets

- [CreateProject](#)— Crée votre projet Amazon Rekognition Custom Labels qui consiste en un regroupement logique de ressources (images, étiquettes, modèles) et d'opérations (formation, évaluation et détection).
- [DeleteProject](#)— Actorie un projet Amazon Rekognition Custom Labels.
- [DescribeProjects](#)— Renvoie une liste de tous vos projets Amazon Rekognition Custom Labels.

Politiques d'un projet

- [PutProjectPolicy](#)— Attache une politique de projet à un projet Amazon Rekognition Custom Labels sur unAWS compte de confiance.
- [ListProjectPolicies](#)— Renvoie la liste des politiques de projet associées à un projet.
- [DeleteProjectPolicy](#)— Supprime une politique de projet existante.

Jeux de données

- [CreateDataset](#)— Crée un jeu de données Amazon Rekognition Custom Labels.
- [DeleteDataset](#)— Actorie un jeu de données Amazon Rekognition Custom Labels.
- [DescribeDataset](#)— Déploie un jeu de données Amazon Rekognition Custom Labels.
- [DistributeDatasetEntries](#)— Réploie les entrées (images) d'un jeu de données de formation dans le jeu de données de formation et le jeu de données de test d'un projet
- [ListDatasetEntries](#)— Renvoie la liste des entrées (images) d'un jeu de données Amazon Rekognition Custom Labels.
- [ListDatasetLabels](#)— Renvoie la liste des étiquettes attribuées à un ensemble de données Amazon Rekognition Custom Labels.
- [UpdateDatasetEntries](#)— Ajoute ou met à jour des entrées (images) dans un jeu de données Amazon Rekognition Custom Labels.

Modèles

- [CreateProjectVersion](#)— Entraîne votre modèle Amazon Rekognition Custom Labels.
- [CopyProjectVersion](#)— Copie votre modèle Amazon Rekognition Custom Labels.
- [DeleteProjectVersion](#)— Supprime un modèle Amazon Rekognition Custom Labels.
- [DescribeProjectVersions](#)— Renvoie la liste de tous les modèles d'étiquettes personnalisées Amazon Rekognition au sein d'un projet spécifique.

Étiquettes

- [TagResource](#)— Ajoute une ou plusieurs balises clé-valeur à un modèle d'étiquettes personnalisées Amazon Rekognition.
- [UntagResource](#)— Supprime une ou plusieurs balises d'un modèle d'étiquettes personnalisées Amazon Rekognition.

Utilisation votre modèle

- [DetectCustomLabels](#)— Analyse une image à l'aide de votre modèle d'étiquettes personnalisées.
- [StartProjectVersion](#)— Démarre votre modèle d'étiquettes personnalisées.
- [StopProjectVersion](#)— Arrête votre modèle d'étiquettes personnalisées.

Historique du document pour Amazon Rekognition Custom Labels

Le tableau ci-après décrit les modifications importantes dans chaque édition du Guide du développeur Amazon Rekognition Custom Labels. Pour recevoir les notifications des mises à jour de cette documentation, abonnez-vous à un flux RSS.

- Dernière mise à jour de la documentation : 19 avril 2023

Modification	Description	Date
Ajout d'une rubrique sur la durée du modèle	Indique comment obtenir le nombre d'heures d'exécution et les unités d'inférence utilisées par un modèle. Pour plus d'informations, consultez la section Rapports sur la durée d'exécution et les unités d'inférence utilisées .	19 avril 2023
Contenu du jeu de données réorganisé	Le contenu de création du fichier manifeste a été déplacé vers le fichier manifeste . Rubriques de conversion de jeux de données déplacées vers Conversion d'autres formats de jeux de données en un fichier manifeste .	20 février 2023
Mise à jour des recommandations IAM pour AWS WAF	Guide mis à jour pour s'aligner sur les bonnes pratiques IAM. Pour de plus amples informations, veuillez consulter Bonnes pratiques de sécurité dans IAM .	15 février 2023

[Afficher la matrice de confusion d'un modèle de classification](#)

La console Amazon Rekognition Custom Labels n'affiche pas la matrice de confusion associée à un modèle de classification. Au lieu de cela, vous pouvez utiliser leAWS SDK pour obtenir et afficher une matrice de confusion. Pour plus d'informations, voir [Affichage de la matrice de confusion d'un modèle](#).

4 janvier 2023

[Exemple de fonction Lambda mis à jour](#)

L'exemple de fonction Lambda montre désormais comment analyser les images transmises depuis un fichier local ou un compartiment Amazon S3. Pour plus d'informations, consultez [Analyse d'images à l'aide d'une fonction AWS Lambda](#).

2 décembre 2022

[Les étiquettes personnalisées Amazon Rekognition peuvent désormais copier des modèles entraînés](#)

Vous pouvez désormais copier un modèle entraîné d'unAWS compte vers un autreAWS au sein de la mêmeAWS région. Pour plus d'informations, consultez [Copier un modèle d'étiquettes personnalisées \(SDK\) Amazon Rekognition](#).

16 août 2022

<u>Les étiquettes personnalisées Amazon Rekognition peuvent désormais dimensionner automatiquement les unités d'inférence.</u>	Pour faire face aux pics de demande, les étiquettes personnalisées Amazon Rekognition peuvent désormais adapter le nombre d'unités d'inférence utilisées par votre modèle. Pour plus d'informations, consultez <u>Exécuter un modèle d'étiquettes personnalisées Amazon Rekognition expérimenté.</u>	16 août 2022
<u>Création d'un fichier manifeste à partir d'un fichier CSV</u>	Vous pouvez désormais simplifier la création d'un fichier manifeste en utilisant un script qui lit les informations de classification des images à partir d'un fichier CSV. Pour plus d'informations, voir <u>Création d'un fichier manifeste à partir d'un fichier CSV.</u>	2 février 2022
<u>Amazon Rekognition Custom Labels gère désormais des ensembles de données avec des projets</u>	Vous pouvez utiliser des projets pour gérer les ensembles de données d'entraînement et de test que vous utilisez pour créer un modèle. Pour plus d'informations, consultez <u>la section Présentation des étiquettes personnalisées Amazon Rekognition.</u>	1er novembre 2021

Les étiquettes personnalisées Amazon Rekognition sont intégrées àAWS CloudFormation	Vous pouvez l'utiliserAWS CloudFormation pour provisionner et configurer les projets Amazon Rekognition Custom Labels. Pour plus d'informations, consultez Création d'un projet avecAWS CloudFormation .	21 octobre 2021
Expérience de démarrage mise à jour	La console Amazon Rekognition Custom Labels inclut désormais des didacticiels vidéo et des exemples de projets. Pour plus d'informations, consultez Mise en route avec Amazon Rekognition Custom Labels .	22 juillet 2021
Informations mises à jour sur les seuils et l'utilisation des métriques	Informations sur la définition d'une valeur de seuil souhaitée à l'aide du paramètre <code>MinConfidence</code> d'entrée pour <code>DetectCustomLabels</code> . Pour plus d'informations, consultez la section Analyse d'une image à l'aide d'un modèle entraîné .	8 juin 2021
AWS KMS keySupport ajouté	Vous pouvez désormais utiliser votre propre clé KMS pour chiffrer vos images d'entraînement et de test. Pour plus d'informations, consultez Entraînement d'un modèle .	19 mai 2021

Ajout d'un balisage	Amazon Rekognition Custom Labels prend désormais en charge le balisage. Vous pouvez utiliser des balises pour identifier, organiser, rechercher et filtrer vos modèles d'étiquettes personnalisées Amazon Rekognition. Pour plus d'informations, consultez Balisage d'un modèle .	25 mars 2021
Rubrique de configuration mise à jour	Informations de configuration mises à jour sur la façon de crypter les fichiers d'entraînement. Pour plus d'informations, voir Étape 5 : (Facultatif) Chiffrement des fichiers de formation .	18 mars 2021
Rubrique de copie du jeu de données ajoutée	Informations sur la manière de copier un jeu de données vers une autreAWS région. Pour plus d'informations, consultez la section Copie d'un jeu de données vers une autreAWS région .	5 mars 2021

[Ajout d' SageMaker GroundTruth une rubrique sur la transformation du manifeste multilabel Amazon](#)

Informations sur la façon de transformer un manifeste au format SageMaker GroundTruth multi-étiquette Amazon en un fichier manifeste au format Amazon Rekognition Custom Labels. Pour plus d'informations, consultez la section [Transformation de fichiers manifestes SageMaker Ground Truth à étiquettes multiples](#).

22 février 2021

[Ajout d'informations de débogage pour la formation des modèles](#)

Vous pouvez désormais utiliser les manifestes des résultats de validation pour obtenir des informations de débogage détaillées sur les erreurs d'entraînement des modèles. Pour plus d'informations, consultez Échec de la section [Mise au point d'un modèle d'entraînement qui a échoué](#).

8 octobre 2020

[Ajout d'informations et d'un exemple de transformation COCO](#)

Informations sur la façon de transformer un ensemble de données au format de détection d'objets COCO en un fichier manifeste d'étiquettes personnalisées Amazon Rekognition. Pour plus d'informations, consultez [Transformer les ensembles de données COCO](#).

2 septembre 2020

Amazon Rekognition Custom Labels prend désormais en charge l'apprentissage sur un seul objet	Pour créer un modèle d'étiquettes personnalisées Amazon Rekognition qui trouve l'emplacement d'un seul objet, vous pouvez désormais créer un ensemble de données qui ne nécessite qu'une seule étiquette. Pour plus d'informations, consultez Dessiner des cadres de délimitation .	25 juin 2020
Opérations de suppression de projets et de modèles ajoutées	Vous pouvez désormais supprimer des projets et des modèles Amazon Rekognition Custom Labels à l'aide de la console et de l'API. Pour plus d'informations, consultez Supprimer un modèle d'étiquettes personnalisées Amazon Rekognition et Supprimer un projet d'étiquettes personnalisées Amazon Rekognition	1er avril 2020
Exemples Java ajoutés	Ajout d'exemples Java couvrant la création de projets, la formation de modèles, l'exécution de modèles et l'analyse d'images.	13 décembre 2019
Nouvelle fonctionnalité et guide	Il s'agit de la version initiale de la fonctionnalité Amazon Rekognition Custom Labels et du guide du développeur Amazon Rekognition Custom Labels.	3 décembre 2019

Glossaire AWS

Pour connaître la terminologie la plus récente d'AWS, consultez le [Glossaire AWS](#) dans la Référence Glossaire AWS.

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.