aws

Guide des étiquettes personnalisées

Rekognition

Rekognition: Guide des étiquettes personnalisées

Table of Contents

Qu’est-ce qu’Étiquettes personnalisées Amazon Rekognition ?

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

Le développement d’un modèle personnalisé à des fins d’analyse d’images nécessite du temps, de l’expertise et des ressources. Il faut souvent plusieurs mois pour mener ce type d’entreprise à bien. 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 qu’il puisse prendre des décisions en toute connaissance de cause. La collecte de ces données peut prendre des mois et nécessiter de grandes équipes d’étiqueteurs pour les préparer en vue de leur utilisation pour le machine learning.

Étiquettes personnalisées Amazon Rekognition étend les fonctionnalités existantes d’Amazon Rekognition, qui sont déjà entraînées sur des dizaines de millions d’images dans de nombreuses catégories. Au lieu de milliers d’images, vous pouvez charger un petit jeu d’images d’entraînement (généralement quelques centaines d’images ou moins) propres à votre cas d’utilisation. Vous pouvez le faire à l'aide de la easy-to-use console. Si vos images sont déjà étiquetées, Étiquettes personnalisées Amazon Rekognition peut commencer à entraîner un modèle rapidement. Sinon, vous pouvez étiqueter les images directement dans l'interface d'étiquetage ou utiliser Amazon SageMaker AI Ground Truth pour les étiqueter à votre place.

Une fois qu’Étiquettes personnalisées Amazon Rekognition a commencé à s’entraîner à partir de votre jeu d’images, il peut générer un modèle d’analyse d’images personnalisé pour vous en quelques heures seulement. En arrière-plan, Étiquettes personnalisées Amazon Rekognition charge et inspecte automatiquement les données d’entraînement, sélectionne les bons algorithmes de machine learning, entraîne un modèle et fournit les indicateurs de performances du modèle. Vous pouvez ensuite utiliser votre modèle personnalisé via l’API Étiquettes personnalisées Amazon Rekognition et l’intégrer dans vos applications.

Rubriques

- Vous utilisez Étiquettes personnalisées Amazon Rekognition pour la première fois ?

Principaux avantages

Étiquetage des données simplifié

La console Étiquettes personnalisées Amazon Rekognition intègre une interface visuelle qui rend l'étiquetage de vos images simple et rapide. L'interface permet d'appliquer une étiquette à toute l'image. Vous pouvez également identifier et étiqueter des objets spécifiques dans les images à l'aide de cadres de délimitation dotés d'une click-and-drag interface. Si vous disposez d'un ensemble de données volumineux, vous pouvez également utiliser Amazon SageMaker Ground Truth pour étiqueter efficacement vos images à grande échelle.

Machine learning automatique

Aucune expertise en machine learning n'est requise pour créer votre modèle personnalisé. Étiquettes personnalisées Amazon Rekognition inclut des fonctionnalités de machine learning automatique (AutoML) qui se chargent du machine learning pour vous. Une fois les images d'entraînement fournies, Étiquettes personnalisées Amazon Rekognition peut automatiquement charger et inspecter les données, sélectionner les bons algorithmes de machine learning, entraîner un modèle et fournir les indicateurs de performances du modèle.

Simplification de l'évaluation et de l'inférence des modèles, ainsi que des commentaires

Vous évaluez les performances de votre modèle personnalisé sur votre jeu de test. Pour chaque image du set de test, vous pouvez voir la side-by-side comparaison entre la prédiction du modèle et l'étiquette attribuée. Vous pouvez également consulter les détails des indicateurs de performance (précision, rappel, scores F1, scores de confiance, etc.). Vous pouvez commencer à utiliser votre modèle immédiatement pour analyser des images, ou vous pouvez réentraîner de nouvelles versions avec davantage d'images pour améliorer les performances. Après avoir commencé à utiliser votre modèle, vous pouvez suivrez vos prédictions, corriger les éventuelles erreurs et utiliser les données des commentaires pour réentraîner de nouvelles versions du modèle et améliorer les performances.

Pourquoi utiliser Étiquettes personnalisées Amazon Rekognition ?

Amazon Rekognition propose deux fonctionnalités que vous pouvez utiliser pour rechercher des étiquettes (objets, scènes et concepts) dans des images : Étiquettes personnalisées Amazon

Rekognition et la détection d'étiquettes d'Image Amazon Rekognition. Utilisez les informations suivantes pour déterminer la fonctionnalité que vous devez utiliser.

Détection d'étiquettes d'Image Amazon Rekognition

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

Si votre application a besoin de rechercher des étiquettes courantes, nous vous recommandons d'utiliser la détection d'étiquettes d'Image Amazon Rekognition, car vous n'avez pas besoin d'entraîner de modèle. Pour obtenir la liste des étiquettes détectées par la fonctionnalité de détection d'étiquettes d'Image Amazon Rekognition, consultez Détection d'étiquettes.

Si votre application a besoin de rechercher des étiquettes non détectées par la fonctionnalité de détection d'étiquettes d'Image Amazon Rekognition (pièces de machine personnalisées sur une chaîne de montage, par exemple), nous vous recommandons d'utiliser Étiquettes personnalisées Amazon Rekognition.

Étiquettes personnalisées Amazon Rekognition

Vous pouvez utiliser Étiquettes personnalisées Amazon Rekognition pour entraîner un modèle de machine learning à rechercher des étiquettes (objets, logos, scènes et concepts) dans des images spécifiques aux besoins de votre entreprise.

Étiquettes personnalisées Amazon Rekognition peut classer les images (prédictions au niveau de l'image) ou détecter les emplacements d'objets dans une image (prédictions au niveau de l'objet/du cadre de délimitation).

Étiquettes personnalisées Amazon Rekognition offre une plus grande flexibilité quant aux types d'objets et de scènes détectables. Par exemple, vous pouvez utiliser la fonctionnalité de détection d'étiquettes d'Image Amazon Rekognition pour rechercher des plantes et des feuilles. Pour faire la distinction entre des plantes saines, abîmées et infectées, vous devez utiliser Étiquettes personnalisées Amazon Rekognition.

Les exemples suivants illustrent la manière dont vous pouvez utiliser Étiquettes personnalisées Amazon Rekognition.

- Identifier des logos d'équipes sur des maillots et casques de joueurs
- Distinguer des pièces de machine ou des produits spécifiques sur une chaîne de montage
- Identifier des personnages de dessins animés dans une médiathèque
- Localiser des produits d'une marque spécifique dans des rayons de magasins
- Classer des produits agricoles en fonction de leur qualité (pourris, mûrs ou crus)

> **Note**
>
> Étiquettes personnalisées Amazon Rekognition n'est pas conçu pour analyser des visages, détecter du texte ou du contenu inapproprié dans des images. Pour effectuer ces tâches, vous pouvez utiliser Image Amazon Rekognition. Pour plus d'informations, consultez Qu'est-ce qu'Amazon Rekognition ?

Vous utilisez Étiquettes personnalisées Amazon Rekognition pour la première fois ?

Si vous utilisez Étiquettes personnalisées Amazon Rekognition 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 apprendrez à configurer votre compte.
2. Présentation de la fonctionnalité Étiquettes personnalisées Amazon Rekognition : dans cette section, vous découvrirez le flux de création d'un modèle.
3. Mise en route sur Étiquettes personnalisées Amazon Rekognition : dans cette section, vous entraînerez un modèle à l'aide d'exemples de projets créés par Étiquettes personnalisées Amazon Rekognition.
4. Classification des images : dans cette section, vous allez apprendre à entraîner un modèle qui classe les images à l'aide des jeux de données que vous avez créés.

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 AWS compte

Au cours de cette étape, vous allez créer un AWS compte, créer un utilisateur administratif et apprendre à accorder un accès programmatique au AWS SDK.

Rubriques

- Inscrivez-vous pour un Compte AWS
- Création d’un utilisateur doté d’un accès administratif
- Accès par programmation

Inscrivez-vous pour un Compte AWS

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

Pour vous inscrire à un Compte AWS

1. Ouvrez l'https://portal.aws.amazon.com/billing/inscription.
2. Suivez les instructions en ligne.

   Une partie de la procédure d'inscription consiste à recevoir un appel téléphonique ou un message texte et à saisir un code de vérification sur le clavier du téléphone.

   Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWSest créé. Par défaut, seul l’utilisateur racine 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, et à utiliser uniquement l’utilisateur racine pour effectuer les tâches nécessitant un accès utilisateur racine.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. À tout moment, vous pouvez consulter l'activité actuelle de votre compte et gérer votre compte en accédant à https://aws.amazon.com/et en choisissant Mon compte.

Création d’un utilisateur doté d’un accès administratif

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

Sécurisez votre Utilisateur racine d'un compte AWS

1. Connectez-vous en AWS Management Consoletant que propriétaire du compte en choisissant Utilisateur root et en saisissant votre adresse Compte AWS e-mail. Sur la page suivante, saisissez votre mot de passe.

Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur racine, consultez Connexion en tant qu'utilisateur racine dans le Guide de l'utilisateur Connexion à AWS .

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

   Pour obtenir des instructions, voir Activer un périphérique MFA virtuel pour votre utilisateur Compte AWS root (console) dans le guide de l'utilisateur IAM.

Création d'un utilisateur doté d'un accès administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Activation 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.

   Pour un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, voir Configurer l'accès utilisateur par défaut Répertoire IAM Identity Center dans le Guide de AWS IAM Identity Center l'utilisateur.

Connexion en tant qu'utilisateur doté d'un accès 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 en utilisant un utilisateur d'IAM Identity Center, consultez la section Connexion au portail AWS d'accès dans le guide de l'Connexion à AWS utilisateur.

Attribution d'un accès à d'autres utilisateurs

1. Dans IAM Identity Center, créez un ensemble d'autorisations qui respecte la bonne pratique consistant à appliquer les autorisations de moindre privilège.

   Pour obtenir des instructions, consultez Création d'un ensemble d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .

2. Attribuez des utilisateurs à un groupe, puis attribuez un accès par authentification unique au groupe.

Pour obtenir des instructions, consultez Ajout de groupes dans le Guide de l'utilisateur AWS IAM Identity Center .

Accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'accorder un accès programmatique dépend du type d'utilisateur qui y accède 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 les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Configuration du AWS CLI à utiliser AWS IAM Identity Center dans le guide de AWS Command Line Interface l'utilisateur. • Pour AWS SDKs, outils, et AWS APIs, voir Authentification IAM Identity Center dans le guide de référence AWS SDKs et Tools.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Authentification à l'aide des informations d'identification utilisateur IAM dans le Guide de l'AWS Command Line Interface utilisateur. • Pour les outils AWS SDKs et, voir Authentifier à l'aide d'informations d'identif ication à long terme dans le guide de référence des outils AWS SDKs et. • Pour AWS APIs, voir 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 , AI 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 Amazon Rekognition Custom Labels pour la première fois dans une nouvelle AWS région, Amazon Rekognition Custom Labels crée un bucket (bucket de console) utilisé pour stocker les fichiers de projet. Vous pouvez également utiliser votre propre compartiment Amazon S3 (compartiment externe) pour télécharger les images ou le fichier manifeste sur la console. Pour utiliser un compartiment externe, ajoutez le bloc de stratégie suivant à la stratégie précédente. Remplacez `amzn-s3-demo-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:::amzn-s3-demo-bucket*"
            ]
        }
```

Attribution d'autorisations

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

- Utilisateurs et groupes dans AWS IAM Identity Center :

  Créez un jeu d'autorisations. Suivez les instructions de la rubrique Création d'un jeu d'autorisations du Guide de l'utilisateur AWS IAM Identity Center .

- Utilisateurs gérés dans IAM par un fournisseur d'identité :

  Créez un rôle pour la fédération d'identité. Suivez les instructions de la rubrique Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM.

- Utilisateurs IAM :
  - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d'un rôle pour un utilisateur IAM dans le Guide de l'utilisateur IAM.
  - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique Ajout d'autorisations à un utilisateur (console) du Guide de l'utilisateur IAM.

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 Amazon Rekognition Custom Labels pour la première fois dans une nouvelle AWS région, Amazon Rekognition Custom Labels crée un compartiment Amazon S3 (compartiment console) pour stocker vos projets. Vous devez noter le nom du compartiment de console à un endroit où vous pourrez vous y référer ultérieurement, car vous devrez peut-être l'utiliser dans les opérations du AWS SDK ou dans les tâches de console, telles que la création d'un ensemble de données.

Le format du nom du compartiment est `custom-labels-console-<region>-<random value>`. 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 de plus amples informations, veuillez consulter Autorisation de l'accès à la console.
2. Connectez-vous à la console Amazon Rekognition AWS Management Console et ouvrez-la 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 : Configurez le AWS CLI et AWS SDKs

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

Rubriques

- Installez les AWS SDK
- Octroi d'un accès par programmation
- Configuration des autorisations du kit SDK
- Appel d'une opération Étiquettes personnalisées Amazon Rekognition

Installez les AWS SDK

Suivez les étapes pour télécharger et configurer l' AWS SDKs.

Pour configurer le AWS CLI et le AWS SDKs

- Téléchargez et installez le AWS CLIet le AWS SDKs que vous souhaitez utiliser. Ce guide fournit des AWS CLI exemples pour Java et Python. Pour plus d'informations sur l'installation AWS SDKs, consultez la section Outils pour Amazon Web Services.

Octroi d’un accès par programmation

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

Rubriques

- Exécuter du code sur votre ordinateur local
- Exécution de code dans AWS des environnements

Exécuter du code sur votre ordinateur local

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

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'accorder un accès programmatique dépend du type d'utilisateur qui y accède 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 les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Configuration du AWS CLI à utiliser AWS IAM Identity Center dans le guide de AWS Command Line Interface l'utilisateur. • Pour AWS SDKs, outils, et AWS APIs, voir Authentification IAM Identity Center dans le guide de référence AWS SDKs et Tools.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Authentification à l'aide des informations d'identification utilisateur IAM dans le Guide de l'AWS Command Line Interface utilisateur.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
		• Pour les outils AWS SDKs et, voir Authentifier à l'aide d'informations d'identification à long terme dans le guide de référence des outils AWS SDKs et. • Pour AWS APIs, voir Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Utiliser un profil sur votre ordinateur local

Vous pouvez exécuter les exemples de code AWS CLI et de code présentés dans ce guide à l'aide des informations d'identification à court terme que vous avez créées dans ce guideExécuter 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 SDK Amazon Rekognition Custom Labels AWS et les autres opérations du SDK requises par les exemples. Pour de plus amples informations, veuillez consulter Configuration des autorisations du kit SDK. Pour attribuer des autorisations, consultez Configuration des autorisations du kit SDK.

Pour créer un profil qui fonctionne avec les exemples de code AWS CLI et, choisissez l'une des options suivantes. Assurez-vous que le nom du profil que vous créez est `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 de la section Configuration de l'interface de ligne de commande AWS à utiliser AWS IAM Identity Center.

Pour les exemples de code, nous recommandons d'utiliser un environnement de développement intégré (IDE), qui prend en charge le kit d'outils AWS permettant l'authentification via IAM Identity Center. Pour les exemples Java, voir Commencer à créer avec Java. Pour les exemples Java, voir Commencer à créer avec Python. Pour plus d'informations, consultez informations d'identification du IAM Identity Center.

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

Exécution de code dans AWS des environnements

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

- AWS Lambda function — Utilisez les informations d'identification temporaires que Lambda fournit automatiquement à votre fonction lorsqu'elle assume le rôle d'exécution de la fonction Lambda. Les informations d'identification sont disponibles dans les variables d'environnement Lambda. Vous n'avez pas besoin d'indiquer de profil. Pour plus d'informations, consultez Rôle d'exécution Lambda.
- Amazon EC2 — Utilisez le fournisseur d'informations d'identification du point de terminaison des métadonnées de l' EC2 instance Amazon. Le fournisseur génère et actualise automatiquement vos informations d'identification à l'aide du profil d' EC2 instance Amazon que vous attachez à l' EC2 instance Amazon. Pour plus d'informations, consultez Utiliser un rôle IAM pour accorder des autorisations aux applications exécutées 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 politiques relatives à l'utilisation du 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 politique `AmazonRekognitionCustomLabelsFullAccess` AWS gérée pour obtenir un accès complet à l'API Amazon Rekognition Custom Labels. 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 politiques relatives à l'utilisation du AWS SDK

Pour utiliser le AWS SDK avec la dernière version d'Amazon Rekognition Custom Labels, vous n'avez plus besoin d'autoriser Amazon Rekognition Custom Labels à accéder au compartiment Amazon S3 qui contient vos images de formation 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. Suivez les instructions de la rubrique Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM.
- Utilisateurs IAM :
  - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d'un rôle pour un utilisateur IAM dans le Guide de l'utilisateur IAM.
  - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique Ajout d'autorisations à un utilisateur (console) du Guide de l'utilisateur IAM.

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 AWS compte, dans la AWS région

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 AWS région que vous utilisez car les étiquettes personnalisées Amazon Rekognition n'est pas disponible dans toutes les régions. AWS

Pour appeler une opération Étiquettes personnalisées Amazon Rekognition

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
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).
- Utilise ton AWS KMS key.

> **Note**
>
> Le principal IAM qui appelle a besoin d'autorisations pour déchiffrer les fichiers. Pour plus d'informations, consultez Déchiffrer des 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.

Déchiffrer des fichiers chiffrés avec AWS Key Management Service

Si vous utilisez AWS Key Management Service (KMS) pour chiffrer vos fichiers manifestes et vos fichiers image Amazon Rekognition Custom Labels, ajoutez le responsable IAM qui appelle les étiquettes personnalisées Amazon Rekognition à la politique clé 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 plus d'informations, consultez 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 Étiquettes personnalisées Amazon Rekognition.

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. L'association automatique d'un ensemble de données à un projet ne se produit pas si vous utilisez le 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 Présentation de la fonctionnalité 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 Copier le contenu d'un ensemble de données existant.

3. Suivez les instructions de Suppression d’un projet Étiquettes personnalisées Amazon Rekognition (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.

Présentation de la fonctionnalité Étiquettes personnalisées Amazon Rekognition

Cette section présente un aperçu du flux de travail permettant de former et d'utiliser un modèle d'étiquettes personnalisées Amazon Rekognition avec la console et le SDK. AWS

> **Note**
>
> Étiquettes personnalisées Amazon Rekognition gère désormais l’association de jeux de données à un projet. Vous pouvez créer des ensembles de données pour vos projets à l'aide de la console et du AWS SDK. Si vous avez déjà utilisé Étiquettes personnalisées Amazon Rekognition, vos anciens jeux de données devront peut-être être associés à un nouveau projet. Pour plus d’informations, consultez Étape 6 (Facultatif) : Associer des jeux de données précédents à de nouveaux projets.

Rubriques

Choix de votre type de modèle

Vous décidez d’abord du type de modèle que vous souhaitez entraîner en fonction des objectifs de votre entreprise. Par exemple, vous pouvez entraîner un modèle à rechercher votre logo dans des publications des réseaux sociaux, à identifier vos produits dans des rayons de magasins ou à classer les pièces d’une machine sur une chaîne de montage.

Étiquettes personnalisées Amazon Rekognition peut entraîner les types de modèles suivants :

- Recherche d’objets, de scènes et de concepts
- Recherche des emplacements d’objets

- Recherche de l’emplacement de marques

Pour vous aider à choisir le type de modèle à entraîner, Étiquettes personnalisées Amazon Rekognition fournit des exemples de projet que vous pouvez utiliser. Pour plus d’informations, consultez Mise en route sur Étiquettes personnalisées Amazon Rekognition.

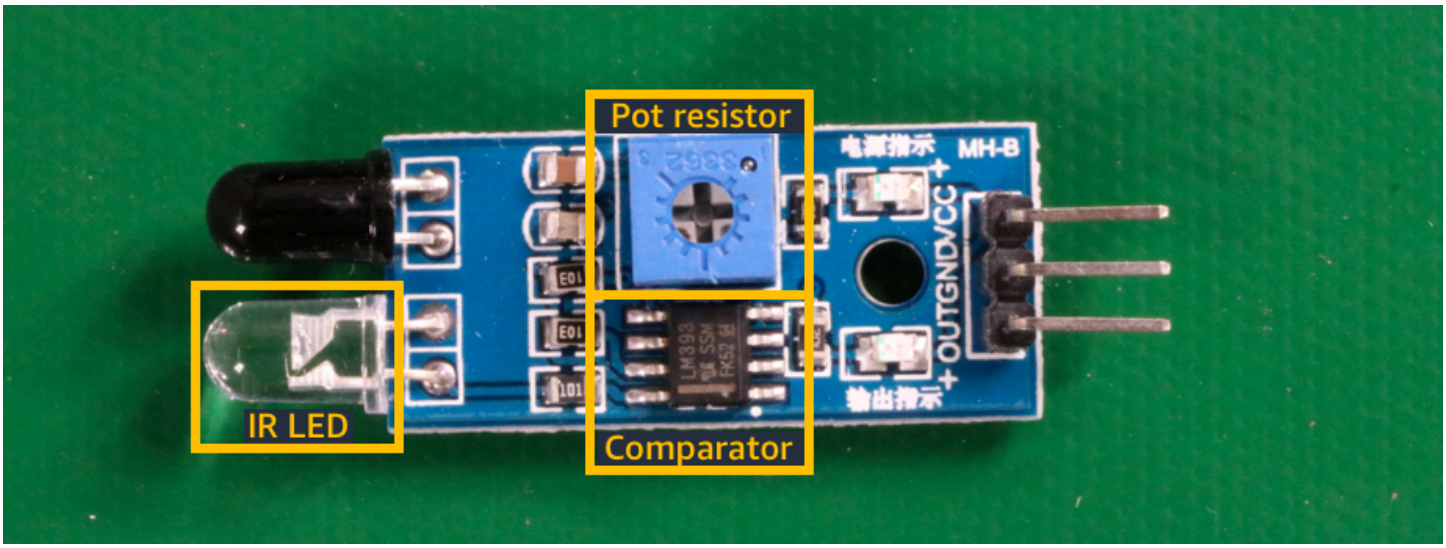
Recherche d’objets, de scènes et de concepts

Le modèle prédit des classifications pour les objets, les scènes et les concepts associés à une image entière. Par exemple, vous pouvez entraîner un modèle à déterminer si une image contient une attraction touristique ou non. Pour un exemple de projet, consultez Classification d’images. L'image suivante d'un lac est un exemple du type d'image dans laquelle vous pouvez reconnaître des objets, des scènes et des concepts.

Vous pouvez également entraîner un modèle à classer les images en plusieurs catégories. Par exemple, l’image précédente peut comporter des catégories telles que couleur du ciel, reflet ou lac. Pour un exemple de projet, consultez Classification des images à plusieurs étiquettes.

Recherche des emplacements d’objets

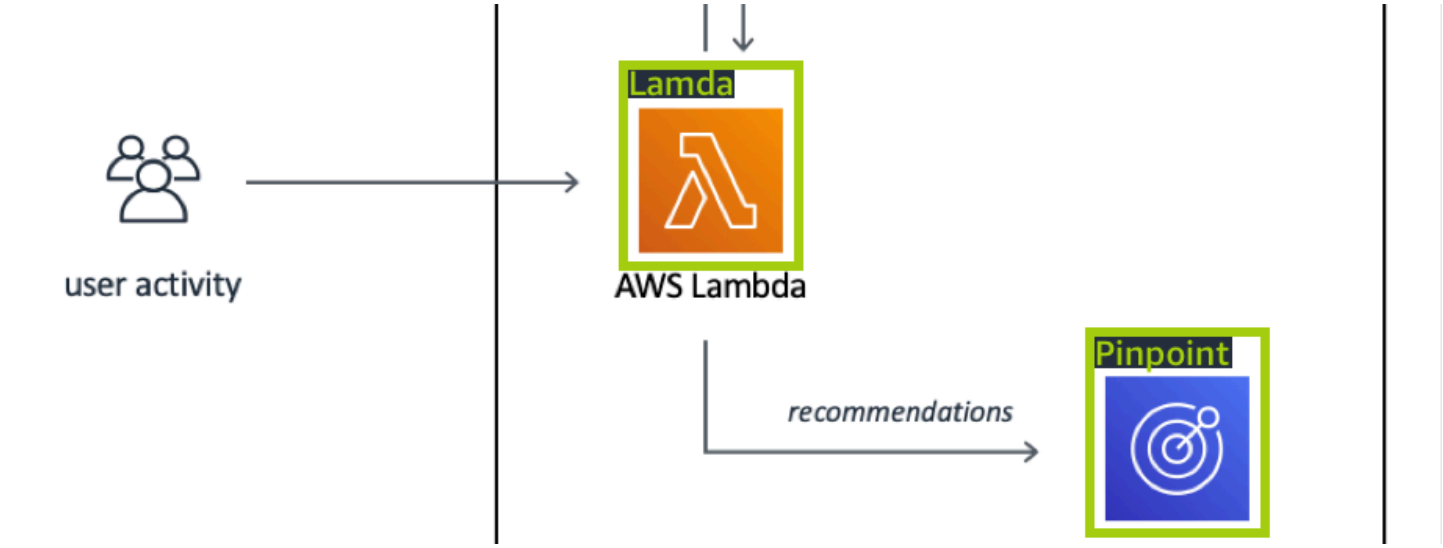
Le modèle prédit l’emplacement d’un objet sur une image. La prédiction comprend des informations sur le cadre de délimitation de l’emplacement d’objet et une étiquette qui identifie l’objet dans le cadre de délimitation. Par exemple, l’image suivante montre des cadres de délimitation autour de différents composant d’un circuit imprimé, comme comparateur ou potentiomètre à résistance.



L’exemple de projet Localisation d’objets montre comment Étiquettes personnalisées Amazon Rekognition utilise des cadres de délimitation étiquetés pour entraîner un modèle à rechercher les emplacements d’objets.

Recherche de l’emplacement de marques

Étiquettes personnalisées Amazon Rekognition peut entraîner un modèle à rechercher l’emplacement de marques (logos, par exemple) sur une image. La prédiction comprend des informations sur le cadre de délimitation de l’emplacement de la marque et une étiquette qui identifie l’objet dans le cadre de délimitation. Pour un exemple de projet, consultez Détection des marques. L'image suivante est un exemple de certaines marques que le modèle peut détecter.



Création d’un modèle

La création d’un modèle inclut la création d’un projet, la création de jeux de données d’entraînement et de test, et l’entraînement du modèle.

Création d’un projet

Un projet Étiquettes personnalisées Amazon Rekognition regroupe les ressources nécessaires à la création et à la gestion d’un modèle. Un projet gère les éléments suivants :

- Jeux de données : images et étiquettes d’images utilisées pour entraîner un modèle. Un projet comprend un jeu de données d’entraînement et un jeu de données de test.
- Modèles : logiciel que vous entraînez à rechercher des concepts, des scènes et des objets propres à votre entreprise. Un projet peut comprendre plusieurs versions d’un modèle.

Nous vous recommandons d’utiliser un projet pour un seul cas d’utilisation (recherche de composants sur un circuit imprimé, par exemple).

Vous pouvez créer un projet à l'aide de la console Amazon Rekognition Custom Labels et de l'API. CreateProject Pour de plus amples informations, veuillez consulter Création d’un 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. Dans votre projet, vous créez un jeu de données d'entraînement et un jeu de données de test qu'Étiquettes personnalisées Amazon Rekognition utilise pour entraîner et 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 soit attribuées à une image entière (au niveau de l'image), soit à un cadre de délimitation qui entoure un objet sur une image.

> **Important**
>
> La façon dont vous étiquetez les images des jeux de données détermine le type de modèle créé par Étiquettes personnalisées Amazon Rekognition. Par exemple, pour entraîner un modèle à rechercher des objets, des scènes et des concepts, vous attribuez des étiquettes de niveau image aux images de vos jeux de données d'entraînement et de test. Pour plus d'informations, consultez 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, consultez Préparation des images.

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

Vous pouvez démarrer un projet avec un seul jeu de données, ou avec des jeux de données d'entraînement et 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 les images qui sont utilisées pour l'entraînement et le test. 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 les jeux de données d'un projet, vous devez importer les images de l'une des manières suivantes :

- Importer des images de votre ordinateur local.
- Importer des images d'un compartiment S3. Étiquettes personnalisées Amazon Rekognition peut étiqueter les images en utilisant les noms des dossiers qui les contiennent.

- Importez un fichier manifeste Amazon SageMaker AI Ground Truth.
- Copier un jeu de données Étiquettes personnalisées Amazon Rekognition existant.

Pour plus d'informations, consultez Création de jeux de données d'entraînement et de test avec des images.

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 AI 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.

Pour créer vos jeux de données d'entraînement et de test avec la console, consultez Création de jeux de données d'entraînement et de test avec des images. Pour accéder à un tutoriel sur la création de jeux de données d'entraînement et de test, consultez Classification des images.

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

Pour créer vos jeux de données d'entraînement et de test, vous utilisez l'API `CreateDataset`. Vous pouvez créer un jeu de données en utilisant un fichier manifeste au format Amazon SageMaker ou en copiant un jeu de données Étiquettes personnalisées Amazon Rekognition existant. Pour plus d'informations, consultez 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, consultez the section called "Création d'un fichier manifeste".

Entraînement de votre modèle

Entraînez votre modèle avec le jeu de données d'entraînement. Une nouvelle version du modèle est créée chaque fois que celui-ci est entraîné. Pendant l'entraînement, Étiquettes personnalisées Amazon Rekognition teste les performances de votre modèle. Vous pouvez utiliser les résultats pour évaluer et améliorer votre modèle. L'entraînement prend un certain temps. Seuls les entraînements de modèles réussis vous sont facturés. Pour plus d'informations, consultez Entraînement d'un modèle Étiquettes personnalisées Amazon Rekognition. Si l'entraînement du modèle échoue, Étiquettes personnalisées Amazon Rekognition fournit des informations de débogage que vous pouvez utiliser. Pour plus d'informations, consultez Débogage d'un entraînement de modèle en échec.

Entraînement de votre modèle (console)

Pour entraîner votre modèle avec la console, consultez Entraînement d'un modèle (console).

Entraînement d'un modèle (kit SDK)

Vous formez un modèle d'étiquettes personnalisées Amazon Rekognition en appelant. CreateProjectVersion Pour de plus amples informations, veuillez consulter Entraînement d'un modèle (kit SDK).

Amélioration de votre modèle

Au cours du test, Étiquettes personnalisées Amazon Rekognition crée des métriques d'évaluation que vous pouvez utiliser pour améliorer votre modèle entraîné.

Évaluation de votre modèle

Évaluez les performances de votre modèle à l'aide des indicateurs de performance créés lors du test. Les indicateurs de performance (score F1, précision, rappel, etc.) vous permettent de comprendre les performances de votre modèle entraîné et de déterminer si vous pouvez l'utiliser en production. Pour plus d'informations, consultez Métriques d'évaluation du modèle.

Évaluation d'un modèle (console)

Pour visualiser les indicateurs de performance, consultez Accès aux métriques d'évaluation (console).

Évaluation d'un modèle (kit SDK)

Pour obtenir des mesures de performance, vous appelez DescribeProjectVersionspour obtenir les résultats des tests. Pour de plus amples informations, veuillez consulter Accès aux métriques d'évaluation d'Étiquettes personnalisées Amazon Rekognition (kit SDK). Les résultats des tests incluent des métriques qui ne sont disponibles dans la console (matrice de confusion pour les résultats de classification, par exemple). 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, consultez F1.
- Emplacement du fichier de synthèse : le résumé des tests inclut toutes les métriques d'évaluation du jeu de données de test et les métriques de chaque étiquette. `DescribeProjectVersions`

renvoie le compartiment S3 et l'emplacement du dossier du fichier de synthèse. Pour plus d'informations, consultez Accès au fichier récapitulatif du modèle.

- Emplacement de l'instantané du manifeste d'évaluation : l'instantané contient des informations sur les résultats des tests, notamment les indices 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 instantanés. Pour plus d'informations, consultez Interprétation de l'instantané du manifeste d'évaluation.

Amélioration de votre modèle

Si des améliorations sont nécessaires, vous pouvez ajouter d'autres images d'entraînement ou améliorer l'étiquetage des jeux de données. Pour plus d'informations, consultez Amélioration d'un modèle Étiquettes personnalisées Amazon Rekognition. Vous pouvez également donner votre avis sur les prédictions de votre modèle et l'utiliser pour améliorer votre modèle. Pour plus d'informations, consultez Améliorer un modèle grâce aux commentaires sur le modèle.

Amélioration de votre modèle (console)

Pour ajouter des images à un jeu de données, consultez Ajout d'autres images à un jeu de données. Pour ajouter ou modifier des étiquettes, consultez the section called "Étiquetage des images".

Pour réentraîner votre modèle, consultez Entraînement d'un modèle (console).

Amélioration de votre modèle (kit SDK)

Pour ajouter des images à un jeu de données ou modifier l'étiquetage d'une image, utilisez l'API `UpdateDatasetEntries`. `UpdateDatasetEntries` met à jour le fichier manifeste ou y ou ajoute des lignes JSON. Chaque ligne JSON contient des informations sur une seule image (étiquettes attribuées, informations sur les cadres de délimitation, par exemple). Pour plus d'informations, consultez Ajout d'autres images (kit SDK). Pour afficher les entrées d'un jeu de données, utilisez l'API `ListDatasetEntries`.

Pour réentraîner votre modèle, consultez Entraînement d'un modèle (kit SDK).

Démarrage de votre modèle

Avant de pouvoir utiliser votre modèle, vous devez le démarrer à l'aide de la console Étiquettes personnalisées Amazon Rekognition ou de l'API `StartProjectVersion`. La durée de

fonctionnement de votre modèle vous est facturée. Pour plus d'informations, consultez Exécution d'un modèle entraîné.

Démarrage de votre modèle (console)

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

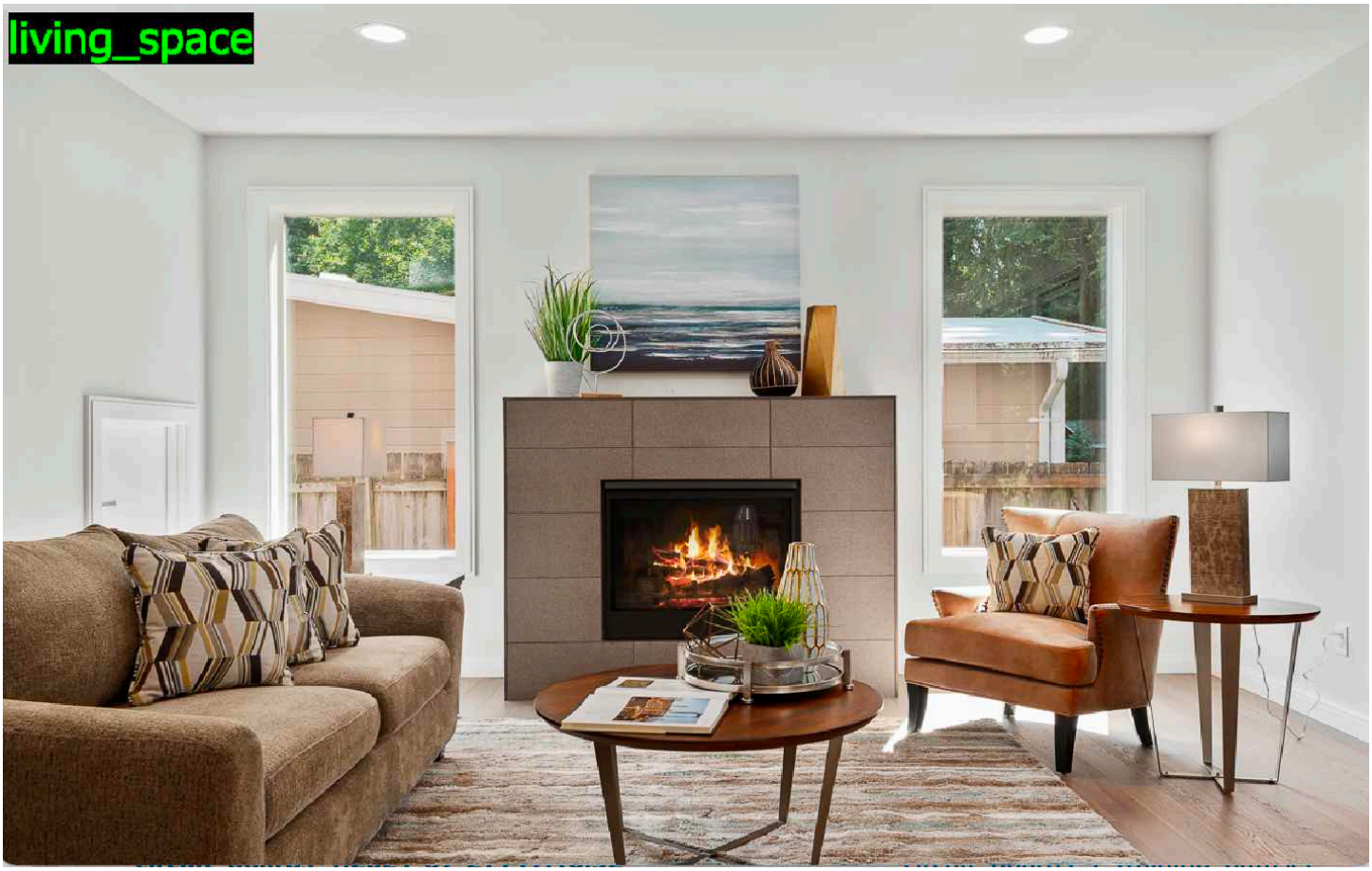
Démarrage de votre modèle

Vous commencez votre appel de mannequins StartProjectVersion. Pour de plus amples informations, veuillez consulter Démarrage d'un modèle Étiquettes personnalisées Amazon Rekognition (kit SDK).

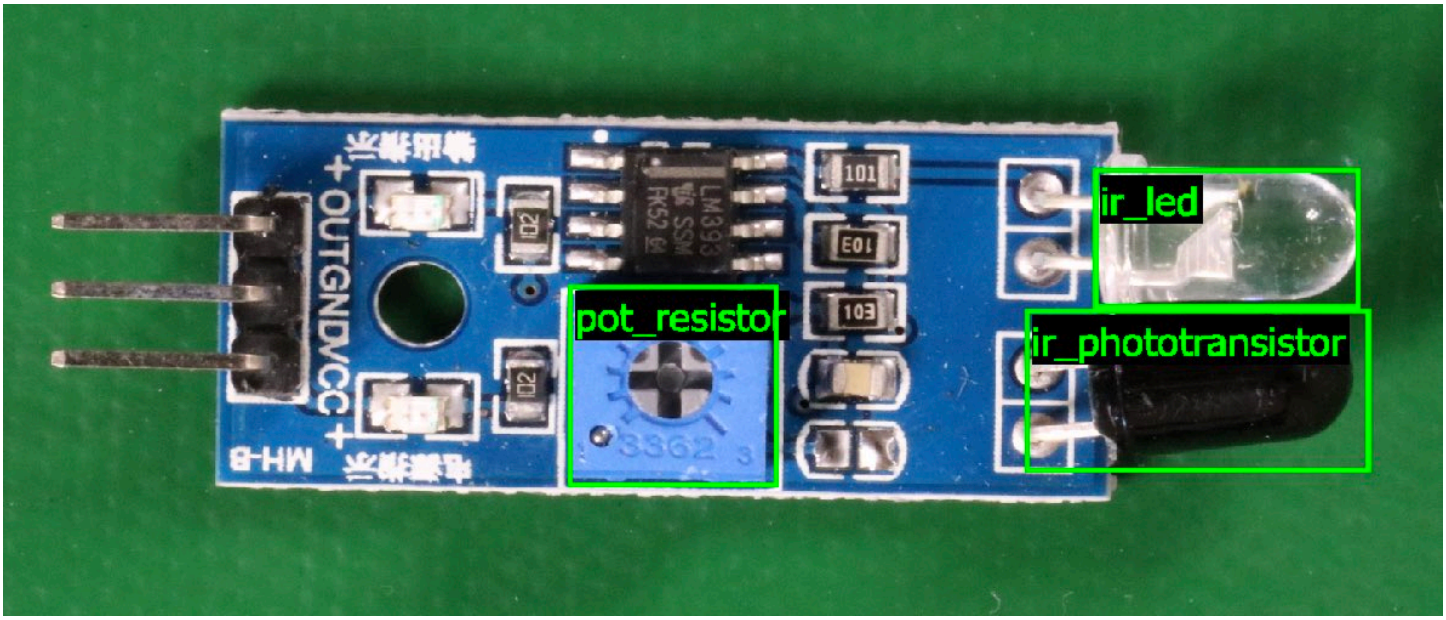
Analyse d'une image

Pour analyser une image avec votre modèle, vous utilisez l'API `DetectCustomLabels`. 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 détecte des objets, des scènes et des concepts, la réponse inclut la 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 détectées à l'aide d'un exemple de projet Pièces.



Si le modèle détecte des emplacements d'objets, la réponse inclut la liste des cadres de délimitation étiquetés trouvés 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 en dessiner un autour d'un objet. Par exemple, l'image suivante montre des cadres de délimitation autour des composants détectés à l'aide de l'exemple de projet Circuits imprimés.



Pour de plus amples informations, veuillez consulter Analyse d'une image avec un modèle entraîné.

Arrêt de votre modèle

La durée de fonctionnement de votre modèle vous est facturée. Si vous n'utilisez plus votre modèle, arrêtez-le à l'aide de la console Étiquettes personnalisées Amazon Rekognition ou de l'API `StopProjectVersion`. Pour plus d'informations, consultez Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition.

Arrêt de votre modèle (console)

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

Arrêt de votre modèle (kit SDK)

Pour arrêter un modèle en cours d'exécution, appelez StopProjectVersion. Pour de plus amples informations, veuillez consulter Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition (kit SDK).

Mise en route sur Étiquettes personnalisées Amazon Rekognition

Avant de commencer ces instructions de Mise en route, nous vous recommandons de lire Présentation de la fonctionnalité Étiquettes personnalisées Amazon Rekognition.

Vous utilisez Étiquettes personnalisées Amazon Rekognition pour entraîner un modèle de machine learning. Un modèle entraîné analyse des images pour rechercher des objets, des scènes et des concepts propres aux besoins de votre entreprise. Par exemple, vous pouvez entraîner un modèle pour classer des images de maisons ou à localiser des composants électroniques sur un circuit imprimé.

Pour vous aider à démarrer, Étiquettes personnalisées Amazon Rekognition comprend des didacticiels vidéo et des exemples de projets.

> **Note**
>
> Pour plus d'informations sur les AWS régions et les points de terminaison pris en charge par Amazon Rekognition Custom Labels, consultez la section Points de terminaison et quotas de Rekognition.

Didacticiels vidéo

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

Pour visionner les didacticiels vidéo

1. Connectez-vous à la console Amazon Rekognition AWS Management Console et ouvrez-la 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. Si l'option Utiliser Custom Labels n'apparaît pas, vérifiez que la Région AWS que vous utilisez prend en charge Étiquettes personnalisées Amazon Rekognition.
3. Dans le volet de navigation, choisissez Démarrer.

4. Dans Qu'est qu'Amazon Rekognition Custom Labels ?, choisissez la vidéo de présentation pour la visionner.
5. Dans le volet de navigation, choisissez Tutoriels.
6. Sur la page Tutoriels, choisissez les didacticiels vidéo que vous souhaitez regarder.

Exemples de projets

La fonctionnalité Étiquettes personnalisées Amazon Rekognition fournit les exemples de projets suivants.

Classification d'images

Le projet de classification d'images (Pièces) entraîne un modèle qui trouve une ou plusieurs pièces sur une image, tels que le jardin, la cuisine et le patio. Les images d'entraînement et de test représentent un emplacement unique. Chaque image est étiquetée avec une seule étiquette au niveau de l'image, telle que kitchen, patio ou living_space. Pour une image analysée, le modèle entraîné renvoie une ou plusieurs étiquettes correspondantes à partir de l'ensemble d'étiquettes au niveau de l'image utilisé pour l'entraînement. Par exemple, le modèle peut trouver l'étiquette living_space dans l'image suivante. Pour de plus amples informations, veuillez consulter Recherche d'objets, de scènes et de concepts.

Classification des images à plusieurs étiquettes

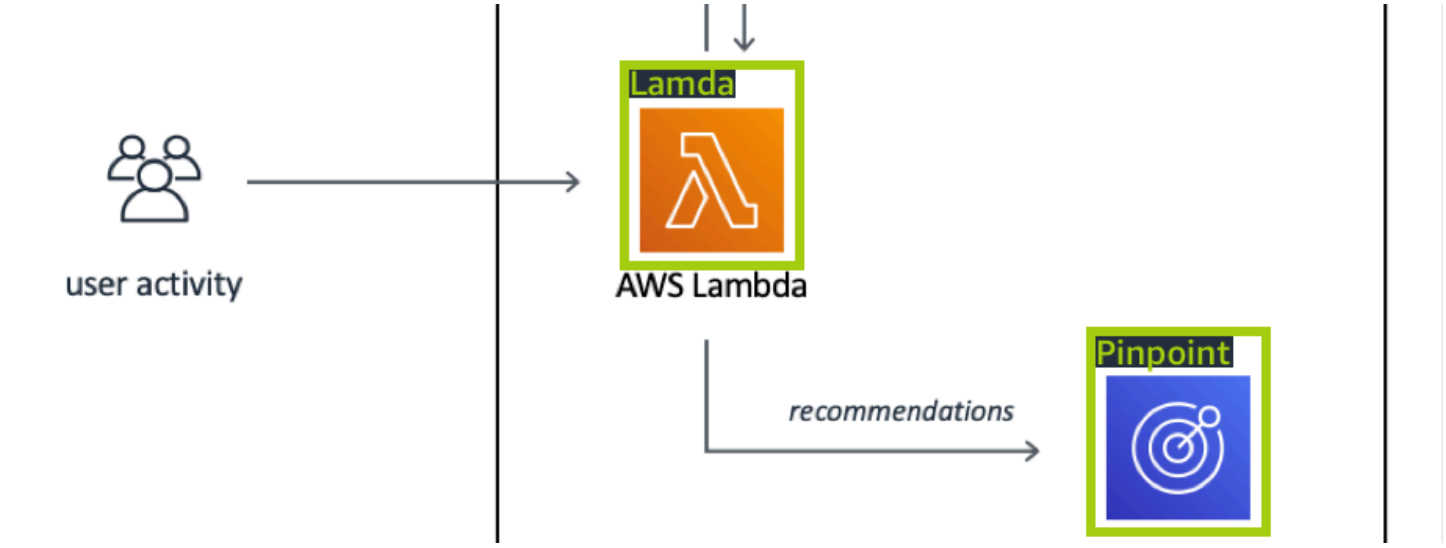
Le projet de classification d'images à plusieurs étiquettes (Fleurs) entraîne un modèle qui classe les images de fleurs en trois concepts (type de fleur, présence des feuilles et stade de croissance).

Les images d'entraînement 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 avec des feuilles et pousse_terminée pour une fleur qui a terminé de pousser.

Pour une image analysée, le modèle entraîné renvoie des étiquettes correspondantes à partir du jeu d'étiquettes au niveau de l'image utilisé pour l'entraînement. Par exemple, le modèle renvoie les étiquettes euphorbe_characias et avec_feuilles pour l'image suivante. Pour de plus amples informations, veuillez consulter Recherche d'objets, de scènes et de concepts.

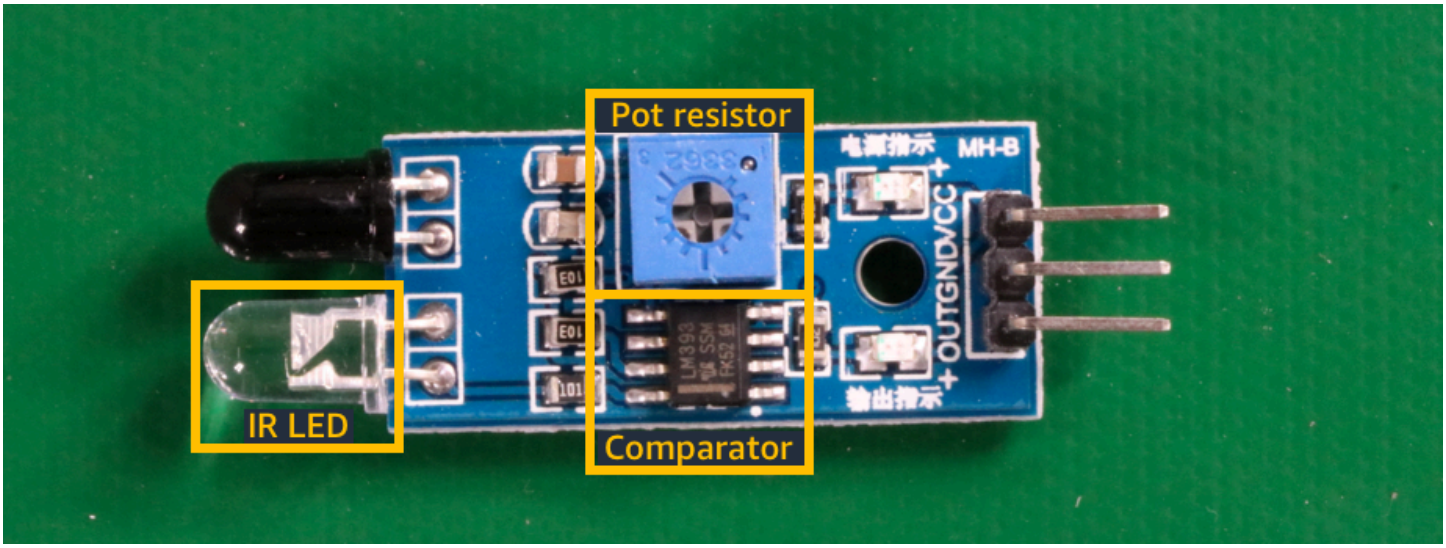
Détection des marques

Le projet de détection de marque (Logos) forme un modèle qui trouve l'emplacement de certains AWS logos tels que Amazon Textract et AWS Lambda. Les images d'entraînement représentent uniquement le logo et comportent une seule étiquette au niveau de l'image, telle que lambda ou textract. Il est également possible d'entraîner un modèle de détection des marques à l'aide d'images d'entraînement dotées de cadres de délimitation indiquant les 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 délimitation étiqueté pour chaque logo trouvé. Pour de plus amples informations, veuillez consulter Recherche de l'emplacement d'une marque.



Localisation d’objets

Le projet de localisation d’objets (Circuits imprimés) entraîne un modèle qui trouve l’emplacement de pièces sur un circuit imprimé, comme un comparateur ou une diode électroluminescente infrarouge. Les images d’entraînement et de test incluent des cadres de délimitation qui entourent les pièces du circuit imprimé et une étiquette identifiant la pièce à l’intérieur du cadre de délimitation. Dans l'exemple d'image suivant, les noms d'étiquette sont ir_phototransistor, ir_led, pot_resistor et comparator. Le modèle entraîné trouve les pièces du circuit imprimé et renvoie une délimitation étiquetée pour chaque pièce de circuit trouvée. Pour de plus amples informations, veuillez consulter Recherche des emplacements d’objets.



Utilisation 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 Étiquettes personnalisées Amazon Rekognition. Elles vous montrent 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, consultez Étape 1 : Choisir un exemple de projet.

La fonctionnalité Étiquettes personnalisées Amazon Rekognition utilise des jeux 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 lesquels toutes les images sont étiquetées. Vous

n'avez pas besoin d'apporter des modifications avant d'entraîner votre modèle. Les exemples de projets montrent les deux manières dont la fonctionnalité Étiquettes personnalisées Amazon Rekognition utilise les étiquettes pour entraîner différents types de modèles.

- au niveau de l'image : l'étiquette identifie un objet, une scène ou un concept qui représente l'image entière.
- cadre 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 devez créer des jeux de données d'entraînement et de test, et également étiqueter vos images. Pour plus d'informations, consultez Choix de votre type de modèle.

Formation du modèle

Une fois que la fonctionnalité Étiquettes personnalisées Amazon Rekognition a créé l'exemple de projet, vous pouvez entraîner le modèle. Pour plus d'informations, consultez Étape 2 : Entraîner 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 à hautes performances et il n'est pas nécessaire d'évaluer le modèle avant de l'exécuter. Pour plus d'informations, consultez Amélioration d'un modèle entraîné Étiquettes personnalisées Amazon Rekognition.

Utilisation du modèle

Ensuite, vous démarrez le modèle. Pour plus d'informations, consultez Étape 3 : Démarrer 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, consultez Étape 4 : Analyser une image avec votre modèle.

La durée de fonctionnement de votre modèle vous est facturée. Lorsque vous avez fini d'utiliser le modèle d'exemple, vous devez arrêter le modèle. Pour plus d'informations, consultez Étape 5 : Arrêter votre modèle.

Étapes suivantes

Lorsque vous êtes prêt, vous pouvez créer vos propres projets. Pour plus d'informations, consultez Étape 6 : étapes suivantes.

Étape 1 : Choisir un exemple de projet

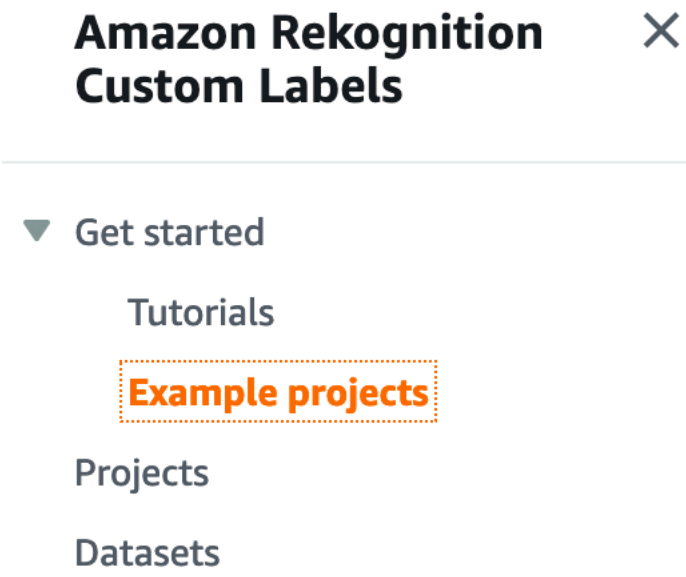
Dans cette étape, vous choisissez un exemple de projet. La fonctionnalité Étiquettes personnalisées Amazon Rekognition crée ensuite un projet et un jeu de données pour vous. Un projet gère les fichiers utilisés pour entraîner votre modèle. Pour plus d'informations, consultez Gestion d'un projet Étiquettes personnalisées Amazon Rekognition. 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, consultez the section called "Gestion des jeux de données".

Pour plus d'informations sur les exemples de projet, consultez Exemples de projets.

Choix d'un exemple de projet

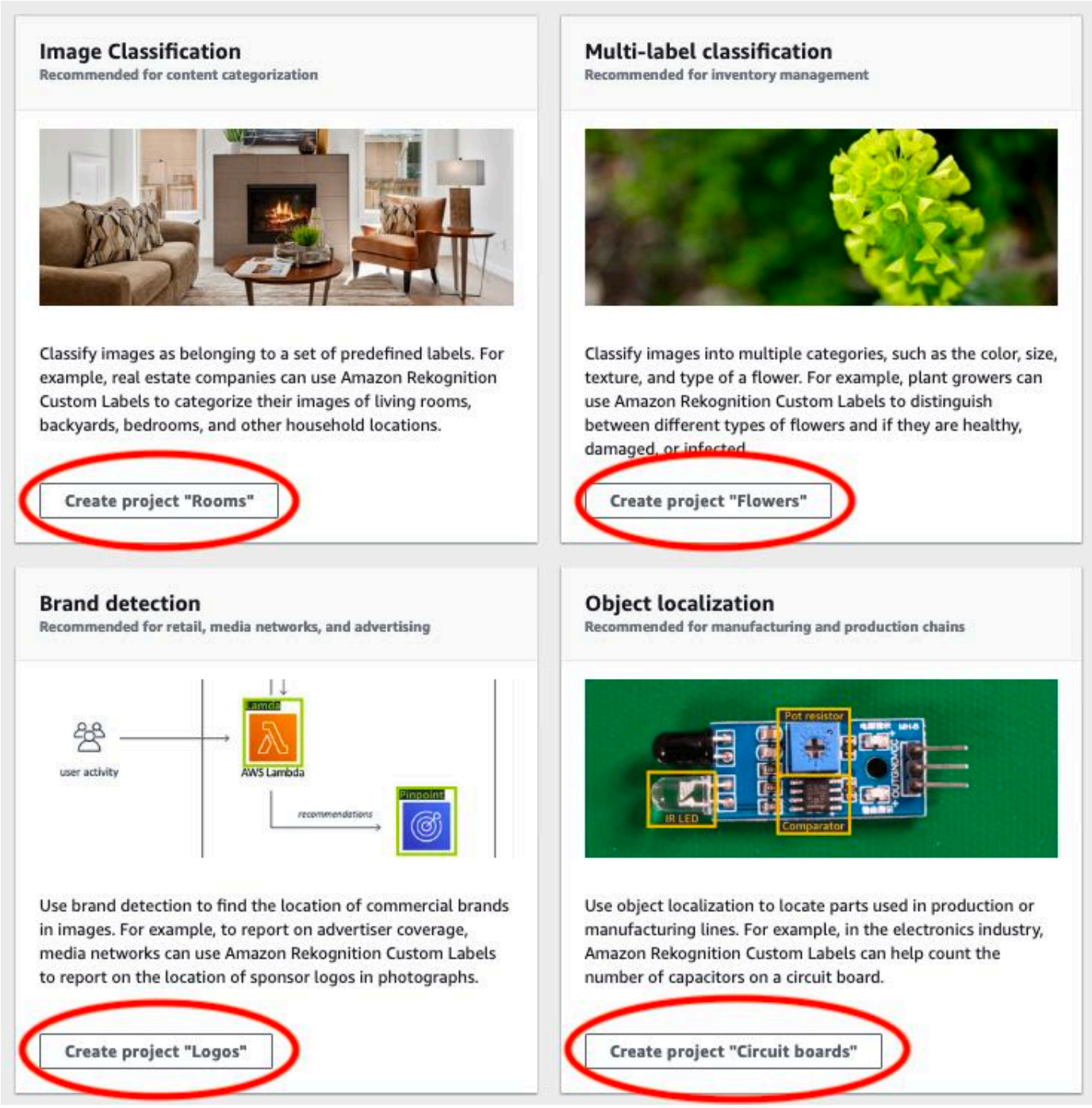
1. Connectez-vous à la console Amazon Rekognition AWS Management Console et ouvrez-la à l'adresse. https://console.aws.amazon.com/rekognition/
2. Dans le volet de gauche, choisissez Utiliser Custom Labels. La page d'accueil d'Étiquettes personnalisées Amazon Rekognition s'affiche. Si l'option Utiliser Custom Labels n'apparaît pas, vérifiez que la Région AWS que vous utilisez prend en charge Étiquettes personnalisées Amazon Rekognition.
3. Choisissez Démarrer.

   Section des étiquettes personnalisées Amazon Rekognition présentant la mise en route, des didacticiels avec « Exemples de projets » surlignés, des projets et des ensembles de données.

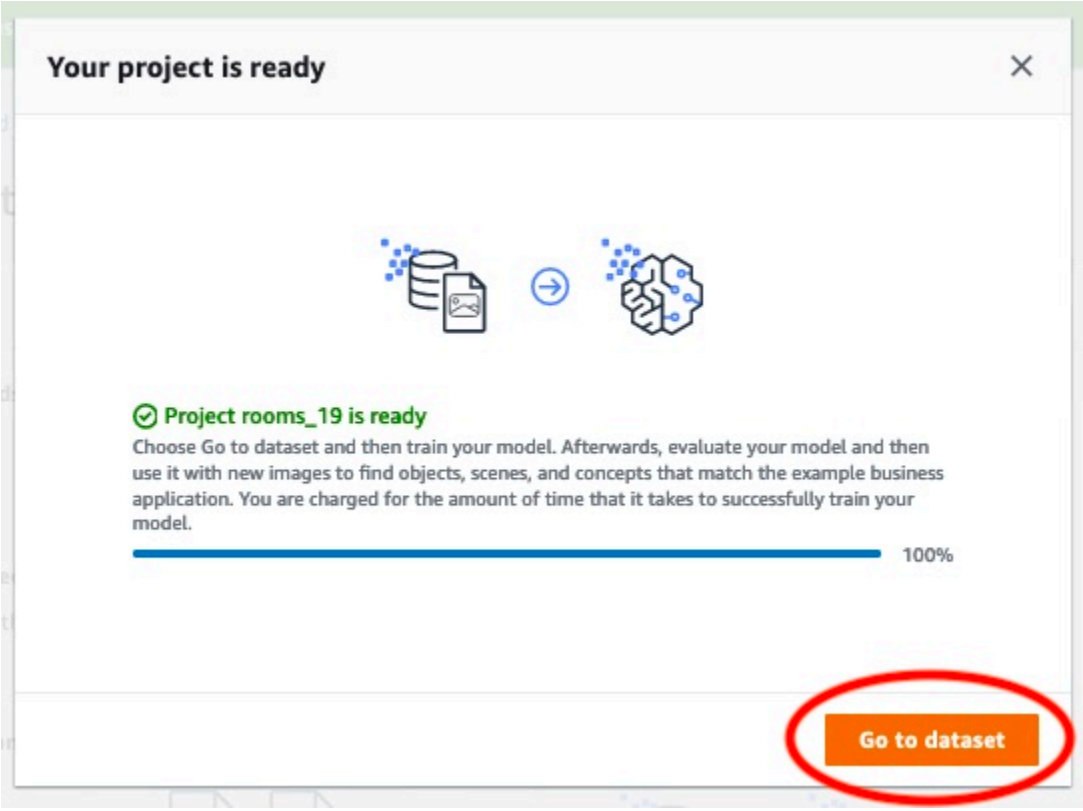


4. Dans Explorer des exemples de projets, choisissez Tester des exemples de projets.
5. Choisissez le projet que vous souhaitez utiliser et choisissez Créer un projet « ***`project name`*** » dans la section d'exemple. La fonctionnalité Étiquettes personnalisées Amazon Rekognition crée ensuite l’exemple de projet pour vous.

> **Note**
>
> Si c'est la première fois que vous ouvrez la console dans la AWS région actuelle, la boîte de dialogue Première configuration s'affiche. Procédez comme suit :
>
> 1. Notez le nom du compartiment Amazon S3 qui s’affiche.
> 2. Choisissez Continuer pour permettre à Étiquettes personnalisées Amazon Rekognition de créer un compartiment Amazon S3 (compartiment de console) en votre nom. L'image de la console ci-dessous montre des exemples de boutons « Créer un projet » pour la classification d'images (pièces), la classification multi-étiquettes (fleurs), la détection de marques (logos) et la localisation d'objets (circuits imprimés).



6. Lorsque votre projet est prêt, choisissez Accéder au jeu de données. L'image suivante montre à quoi ressemble le panneau de projet lorsque le projet est prêt.

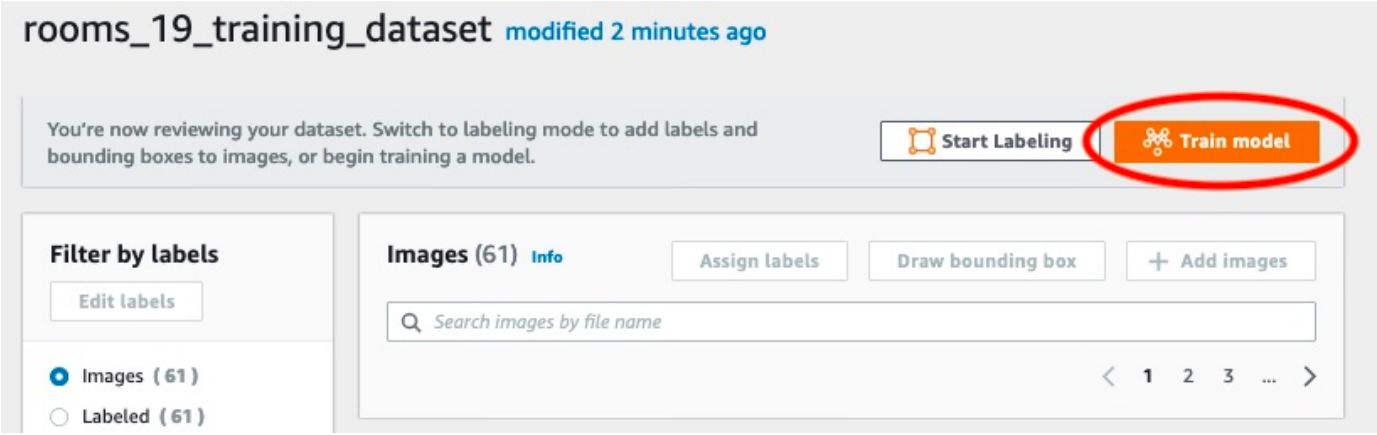


Étape 2 : Entraîner votre modèle

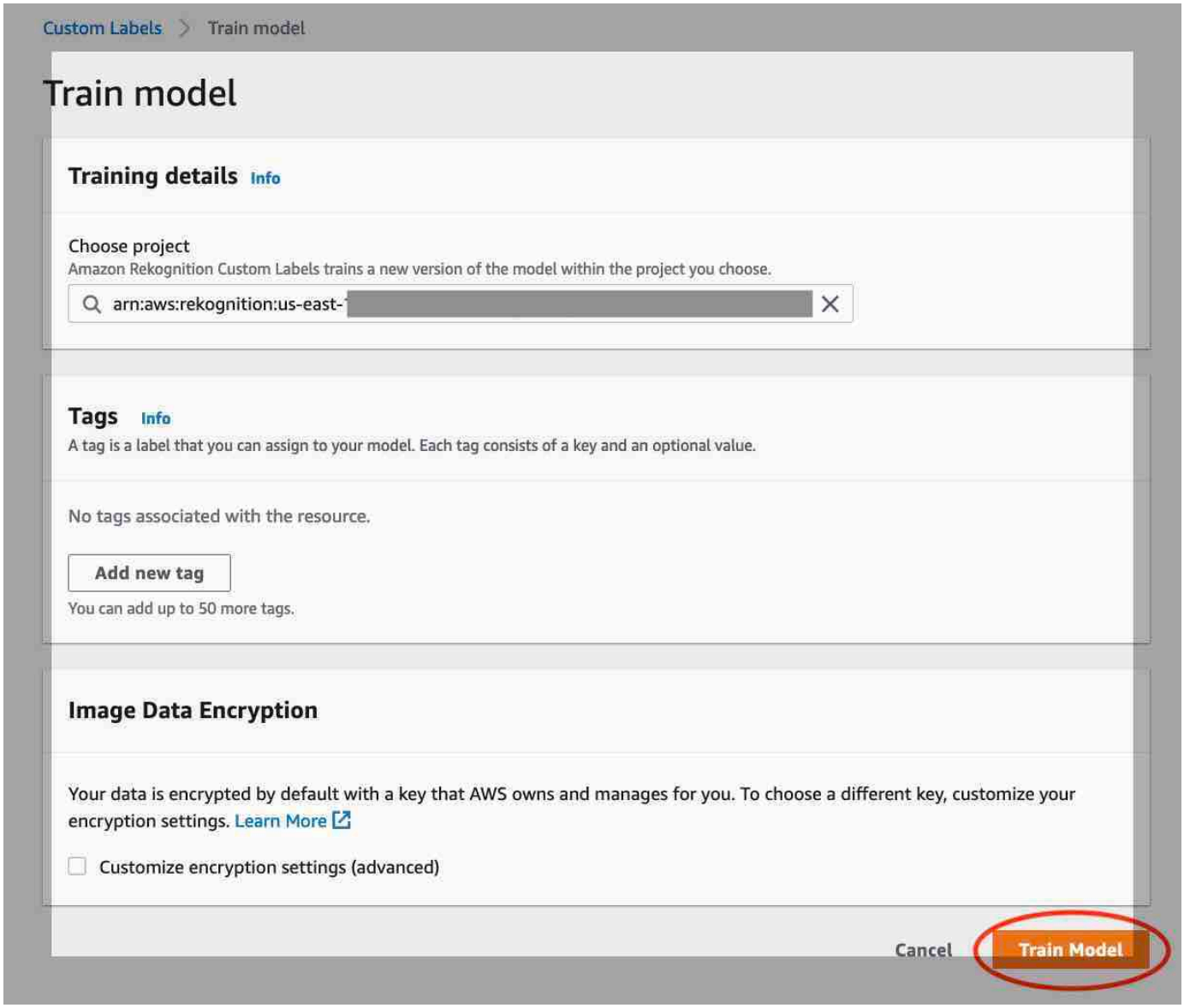
Au cours de cette étape, vous entraînez votre modèle. Les jeux de données d'entraînement et de test sont automatiquement configurés pour vous. Une fois l'entraînement terminé, vous pouvez voir les résultats globaux de l'évaluation et les résultats de l'évaluation des images de test individuelles. Pour plus d'informations, consultez Entraînement d'un modèle Étiquettes personnalisées Amazon Rekognition.

Pour former votre modèle

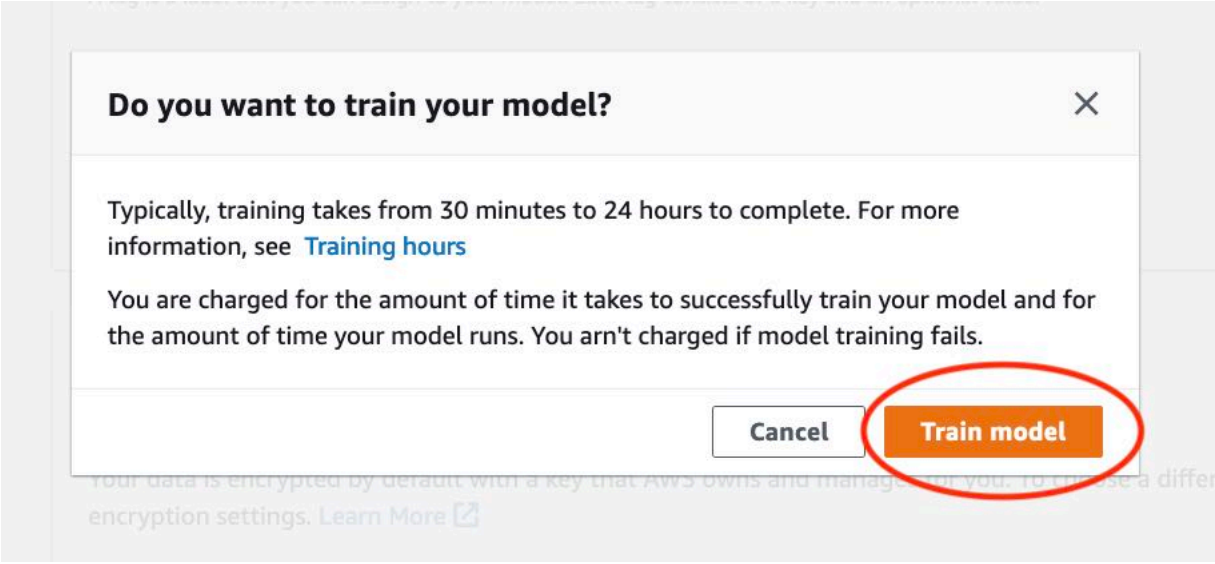
1. Sur la page du jeu de données, choisissez le modèle de train. L'image suivante montre la console avec le bouton du modèle de train.



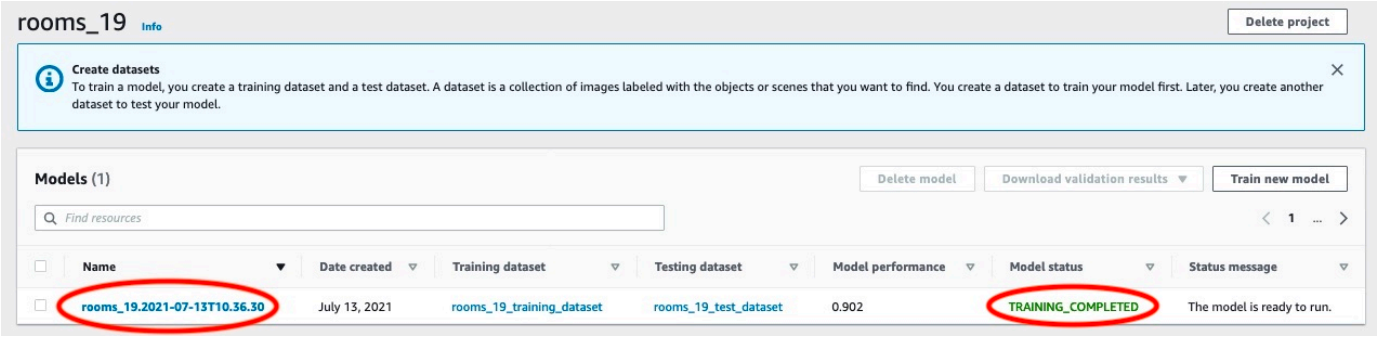
2. Sur la page Entraîner un modèle, choisissez Entraîner un modèle. L'image ci-dessous montre le bouton Train model. Notez que le nom de ressource Amazon (ARN) de votre projet se trouve dans la zone d'édition Choose project.



3. Dans la section Voulez-vous entraîner votre modèle ? dans la boîte de dialogue, illustrée dans l'image suivante, choisissez le modèle de train.

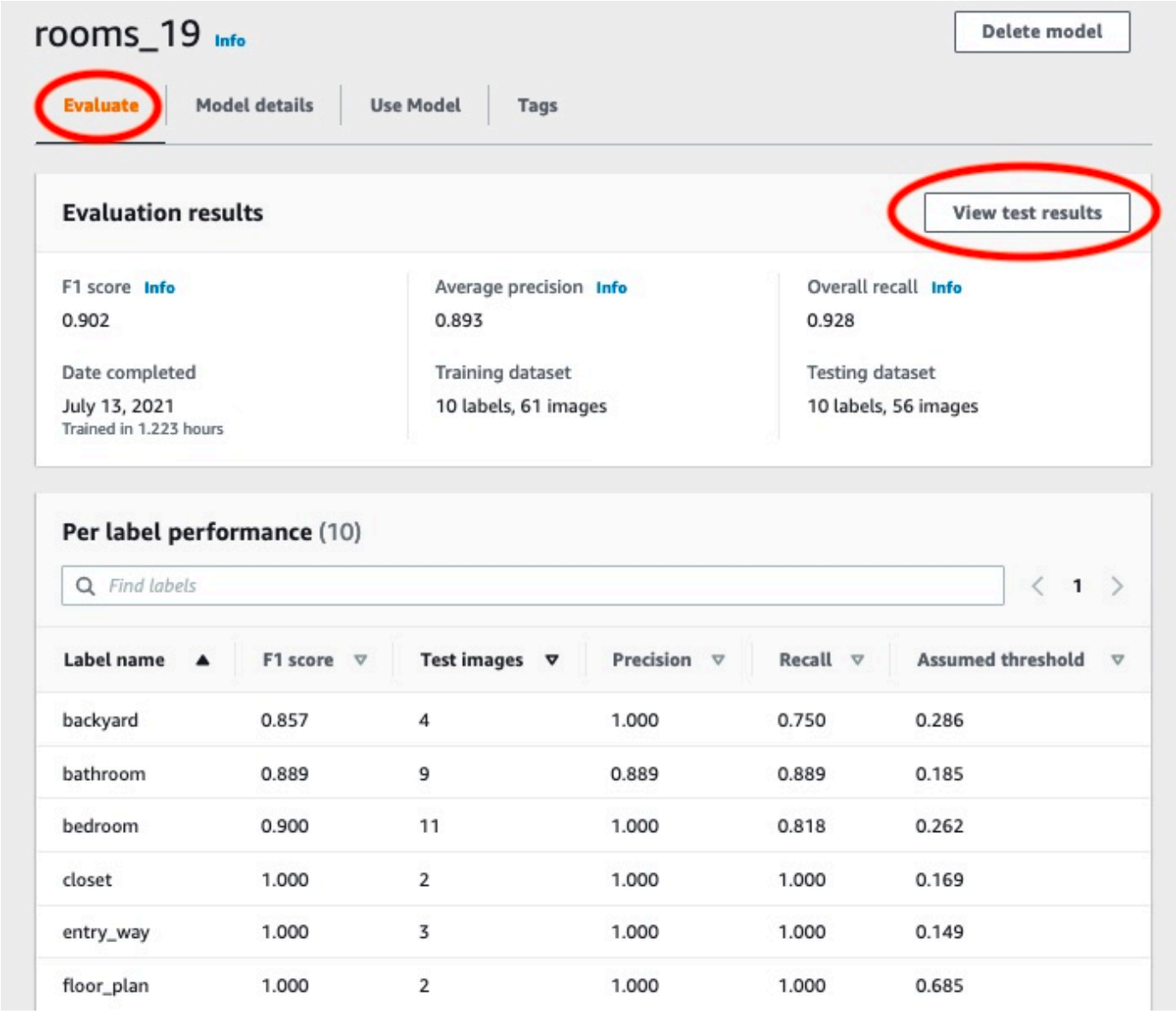


4. Une fois l'entraînement terminé, choisissez le nom du modèle. L'entraînement est terminé lorsque le statut du modèle est TRAINING_COMPLETED, comme le montre la capture d'écran de console suivante.

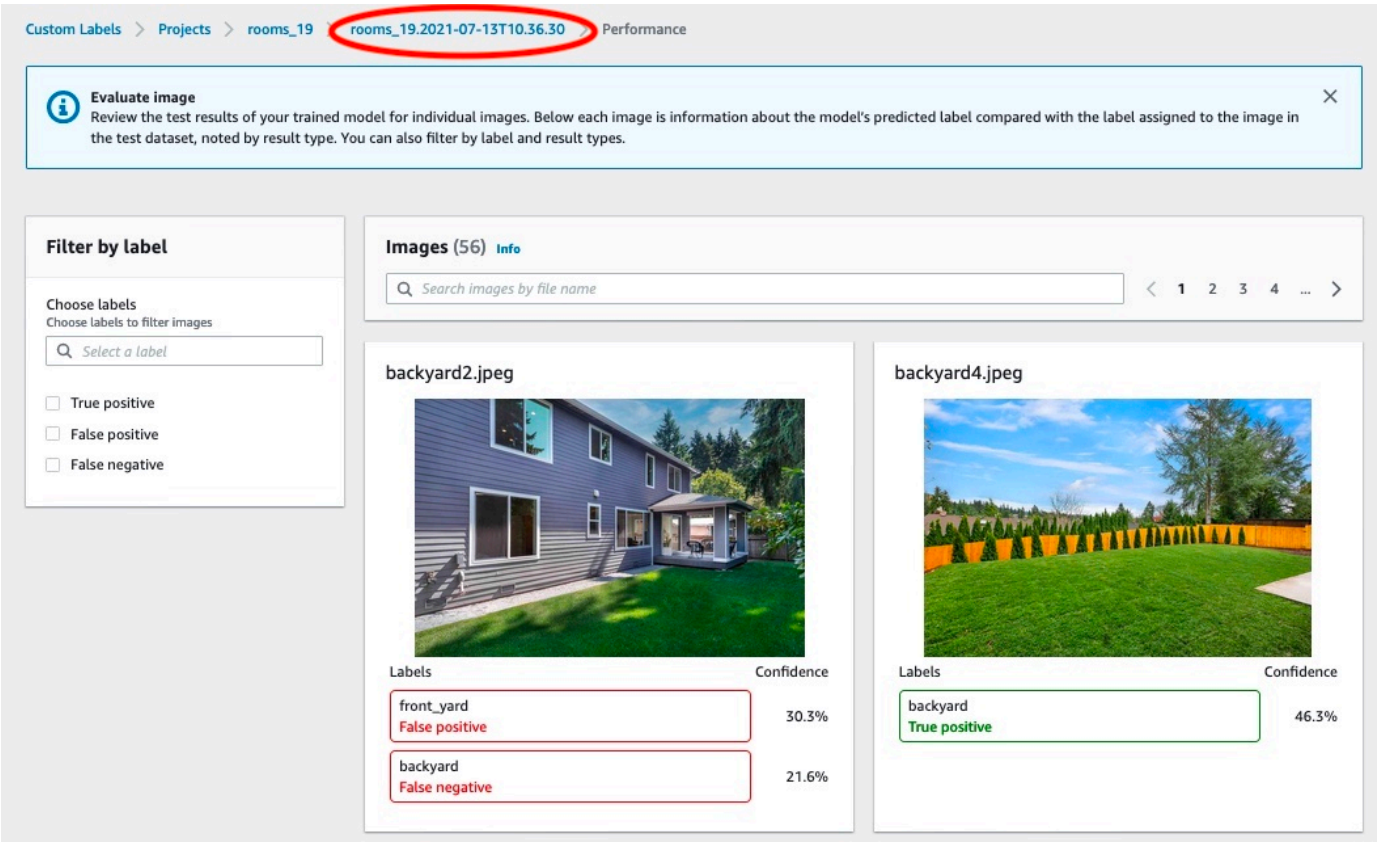


5. Cliquez sur le bouton Évaluer pour voir les résultats de l'évaluation. Pour plus d'informations sur l'évaluation d'un modèle, consultez Amélioration d'un modèle entraîné Étiquettes personnalisées Amazon Rekognition.

6. Choisissez Afficher les résultats des tests pour visualiser les résultats des images des tests. Comme le montre la capture d'écran suivante, le tableau de bord d'évaluation affiche des indicateurs tels que le score F1, la précision et le rappel pour chaque étiquette, ainsi que le nombre d'images de test. Des indicateurs globaux tels que la moyenne, la précision et le rappel sont également affichés.



7. Après avoir consulté les résultats des tests, choisissez le nom du modèle pour revenir à la page du modèle. La capture d'écran suivante du tableau de bord des performances sur laquelle vous pouvez cliquer pour revenir à la page du modèle.



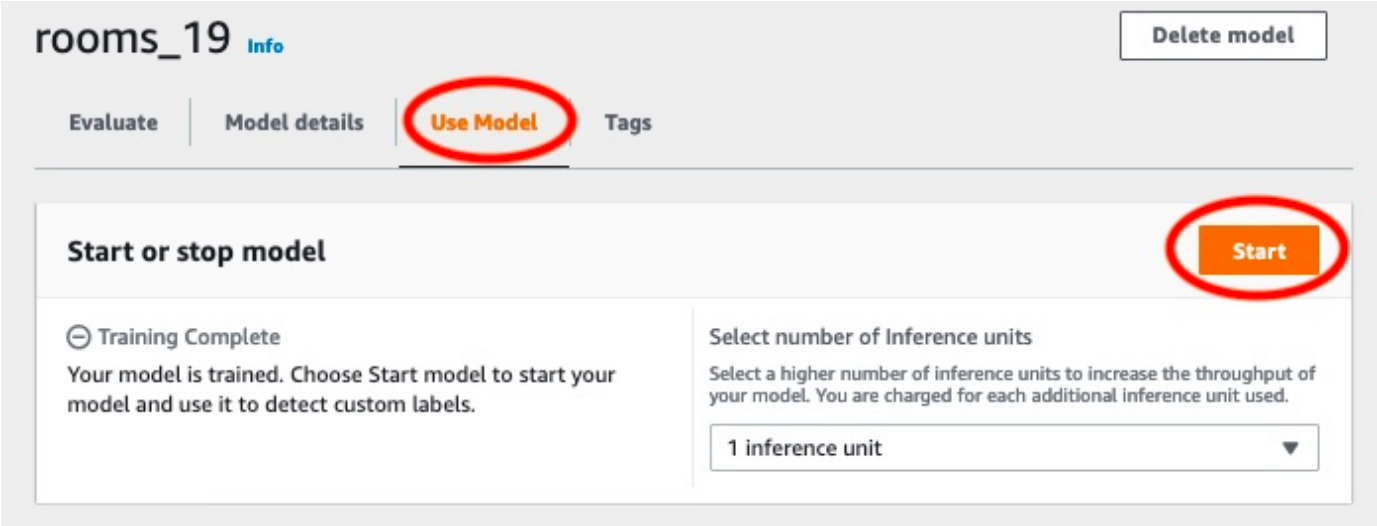
Étape 3 : Démarrer votre modèle

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

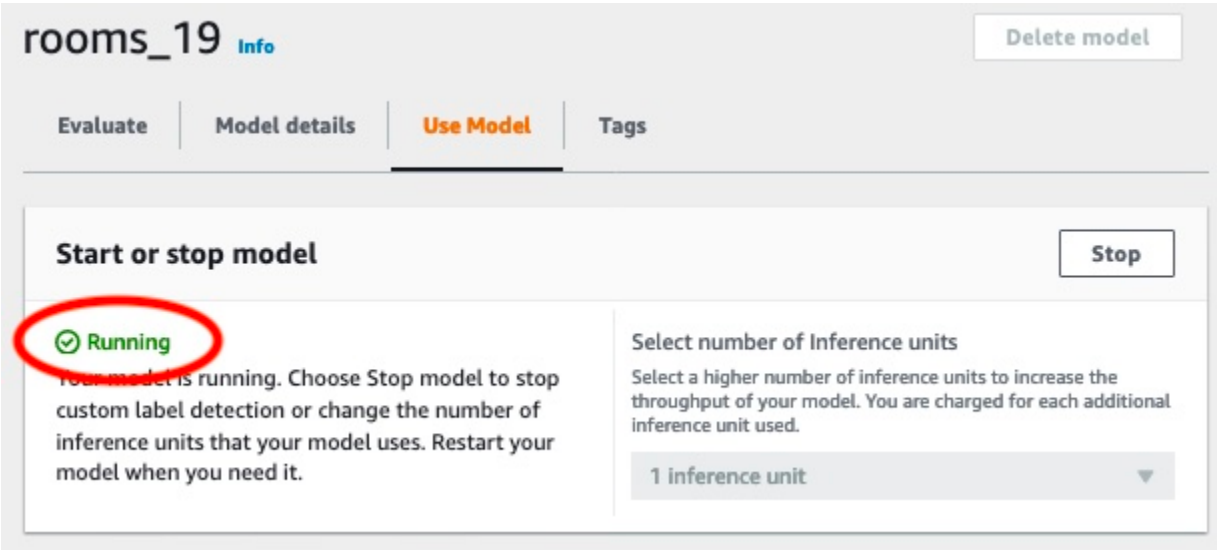
La durée de fonctionnement de votre modèle vous est facturée. Arrêtez votre modèle si vous n'avez pas besoin d'analyser d'images. Vous pourrez le redémarrer ultérieurement. Pour plus d'informations, consultez Exécution d'un modèle Étiquettes personnalisées Amazon Rekognition entraîné.

Pour démarrer votre modèle

1. Choisissez l'onglet Utiliser le modèle sur la page du modèle.
2. Dans la section Démarrer ou arrêter le modèle, procédez comme suit :
    a. Sélectionnez Démarrer.
    b. Dans la boîte de dialogue Démarrer le modèle, choisissez Démarrer. L'image suivante montre le bouton Démarrer du panneau de commande du modèle.



3. Patientez jusqu'à l'exécution du modèle. La capture d'écran suivante montre la console pendant que le modèle est en cours d'exécution, où le statut indiqué dans la section Démarrer ou arrêter le modèle est En cours d'exécution.



4. Utilisez votre modèle pour classer des images. Pour plus d'informations, consultez Étape 4 : Analyser une image avec votre modèle.

Étape 4 : Analyser une image avec votre modèle

Vous analysez une image en appelant l'DetectCustomLabelsAPI. Dans cette étape, vous utilisez la commande `detect-custom-labels` AWS Command Line Interface (AWS CLI) pour analyser un exemple d'image. Vous obtenez la AWS CLI commande depuis la console Amazon Rekognition

Custom Labels. La console configure la AWS CLI commande 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 un exemple de code Python.

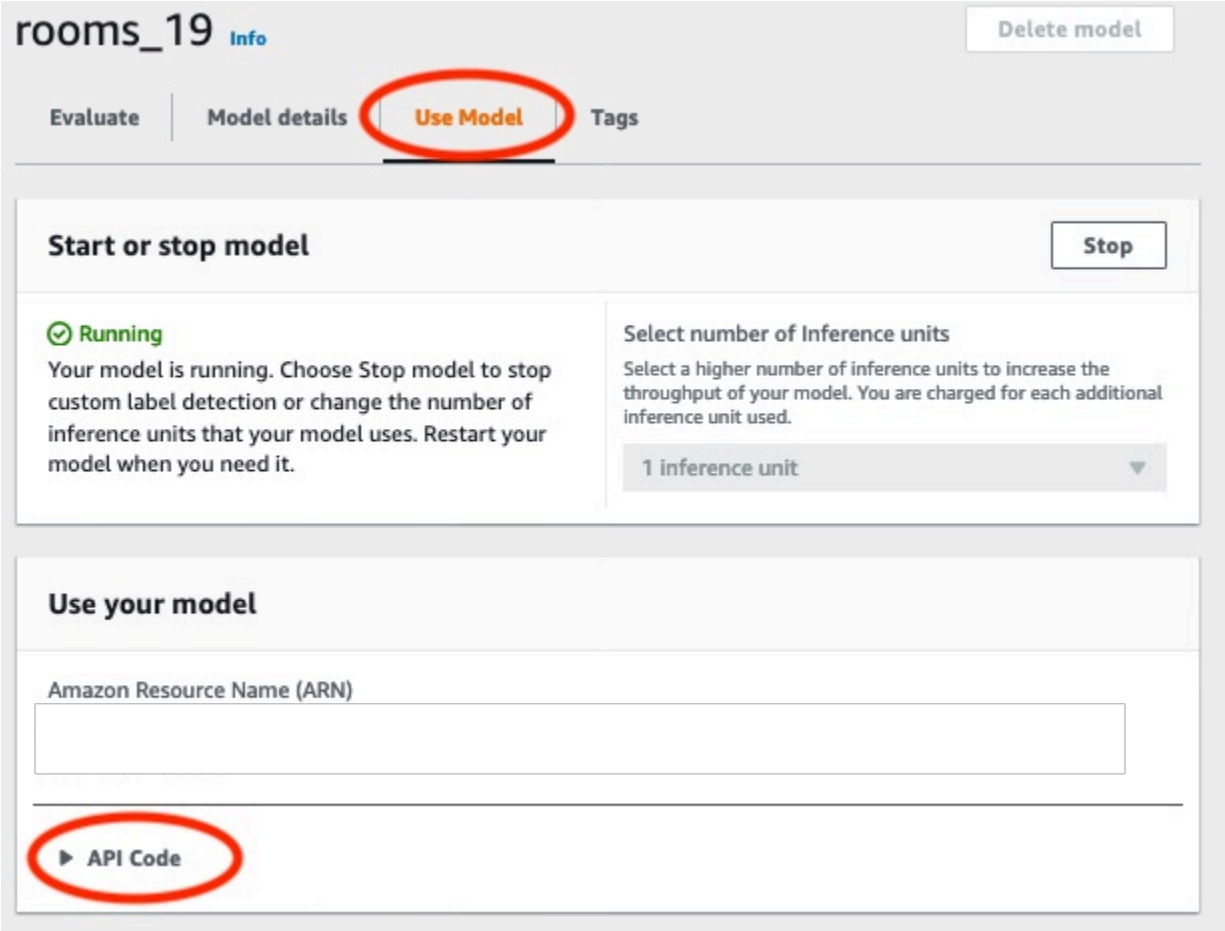
Le résultat de `detect-custom-labels` inclut la liste des étiquettes détectées dans l'image, les cadres de délimitation (si le modèle recherche les emplacements d'objets) et le score de confiance du modèle concernant la précision des prédictions.

Pour plus d'informations, consultez Analyse d'une image avec un modèle entraîné.

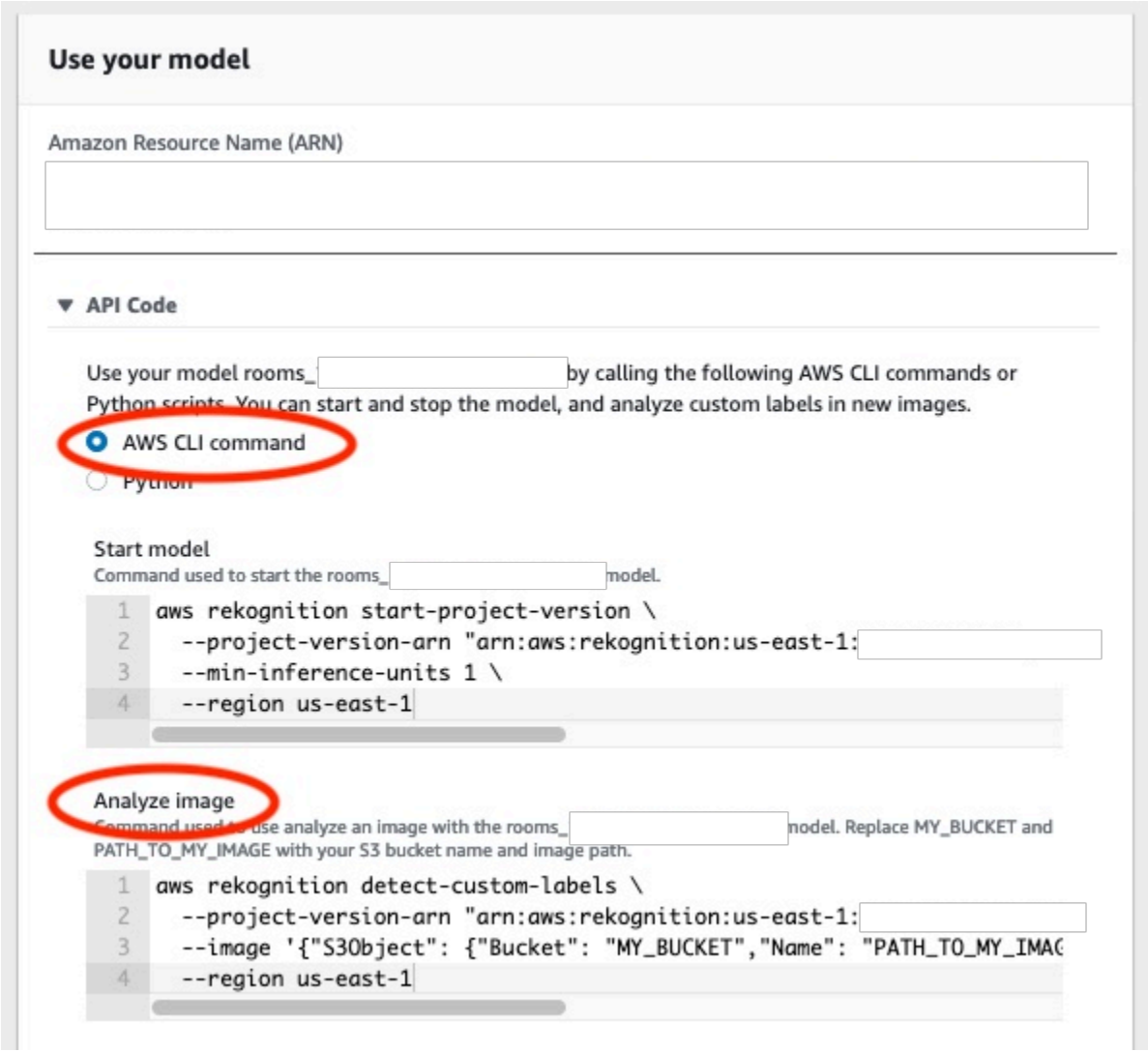
Pour analyser une image (console)

1. L'<textobject><phrase>état du modèle s'affiche comme étant en cours d'exécution, avec le bouton Stop pour arrêter le modèle en cours d'exécution. </phrase></textobject>

   Si ce n'est pas déjà fait, configurez le AWS CLI. Pour obtenir des instructions, veuillez consulter the section called "Étape 4 : Configurez le AWS CLI et AWS SDKs".
2. Commencez à exécuter votre modèle si vous ne l'avez pas déjà fait. Pour plus d'informations, consultez Étape 3 : Démarrer votre modèle.
3. Choisissez l'onglet Utiliser le modèle, puis Code de l'API. Le panneau d'état du modèle illustré ci-dessous indique que le modèle est en cours d'exécution, avec un bouton Stop pour arrêter le modèle en cours d'exécution et une option pour afficher l'API.



4. Choisissez Commande de l’interface de ligne de commande AWS.
5. Dans la section Analyser l'image, copiez la AWS CLI commande qui appelle`detect-custom-labels`. L'image suivante de la console Rekognition montre la section « Analyser l'image » avec la commande AWS CLI pour détecter les étiquettes personnalisées sur une image à l'aide d'un modèle d'apprentissage automatique, ainsi que des instructions pour démarrer le modèle et fournir des détails sur l'image.



6. Chargez un exemple d'image dans un compartiment Amazon S3. Pour obtenir des instructions, veuillez consulter Obtention d'un exemple d'image.
7. À l'invite de commande, entrez la AWS CLI commande que vous avez copiée à l'étape précédente. Elle doit ressembler à l'exemple suivant.

   La valeur de `--project-version-arn` doit être l'Amazon Resource Name (ARN) de votre modèle. La valeur de `--region` doit être la région AWS dans laquelle vous avez créé le modèle.

   Remplacez `MY_BUCKET` et `PATH_TO_MY_IMAGE` par le compartiment Amazon S3 et l'image que vous avez utilisés à l'étape précédente.

   Si vous utilisez le custom-labels-accessprofil 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 la commande AWS CLI doit ressembler à ce qui suit. Name est le nom de l'étiquette au niveau de l'image trouvée par le modèle. `Confidence` (0 à 100) est le niveau de confiance du modèle dans l'exactitude de la prédiction.

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

Si le modèle trouve des emplacements d'objets ou trouve une marque, les cadres de délimitation étiquetés sont renvoyés. `BoundingBox` contient l'emplacement d'un cadre qui entoure l'objet. Name est l'objet que le modèle a trouvé dans le cadre de délimitation. `Confidence` est le niveau de confiance du modèle que le cadre de délimitation contient l'objet.

```
{
    "CustomLabels": [
        {
            "Name": "textract",
            "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, consultez Étape 5 : Arrêter votre modèle.

Obtention d'un exemple d'image

Vous pouvez utiliser les images suivantes avec l'opération `DetectCustomLabels`. 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

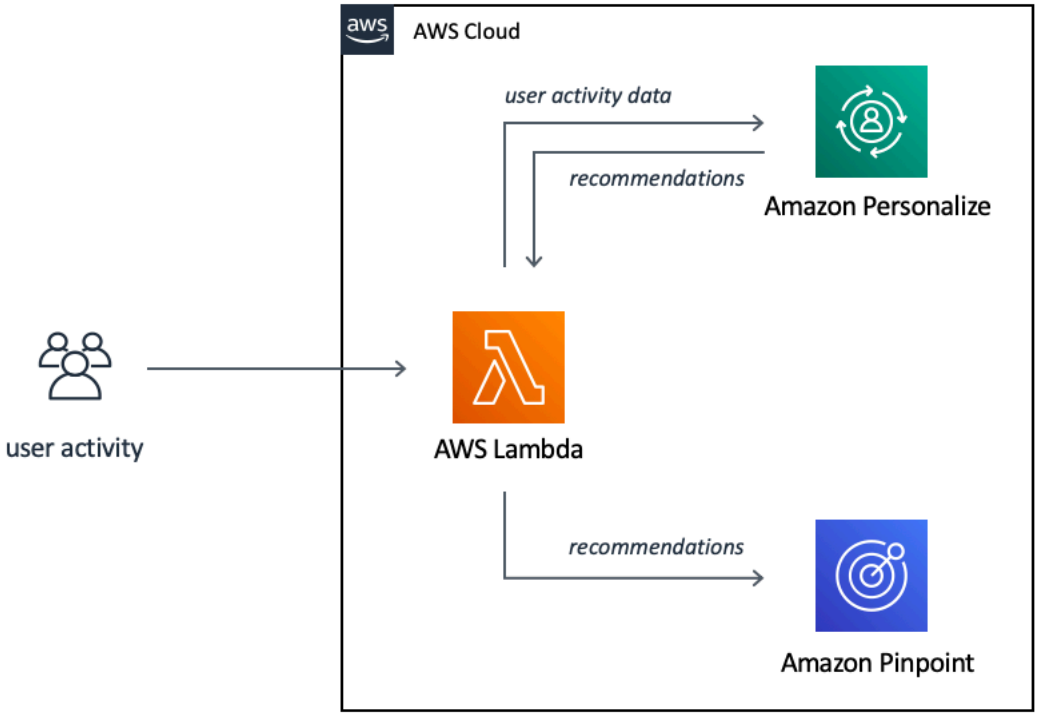
1. Cliquez avec le bouton droit sur l'image suivante qui correspond à l'exemple de projet que vous utilisez. Choisissez ensuite Enregistrer l'image pour enregistrer l'image sur votre ordinateur. L'option du menu peut être différente selon le navigateur que vous utilisez.
2. Téléchargez l'image dans un compartiment Amazon S3 appartenant à votre AWS compte et situé dans la même AWS région que celle dans laquelle vous utilisez les étiquettes personnalisées Amazon Rekognition.

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

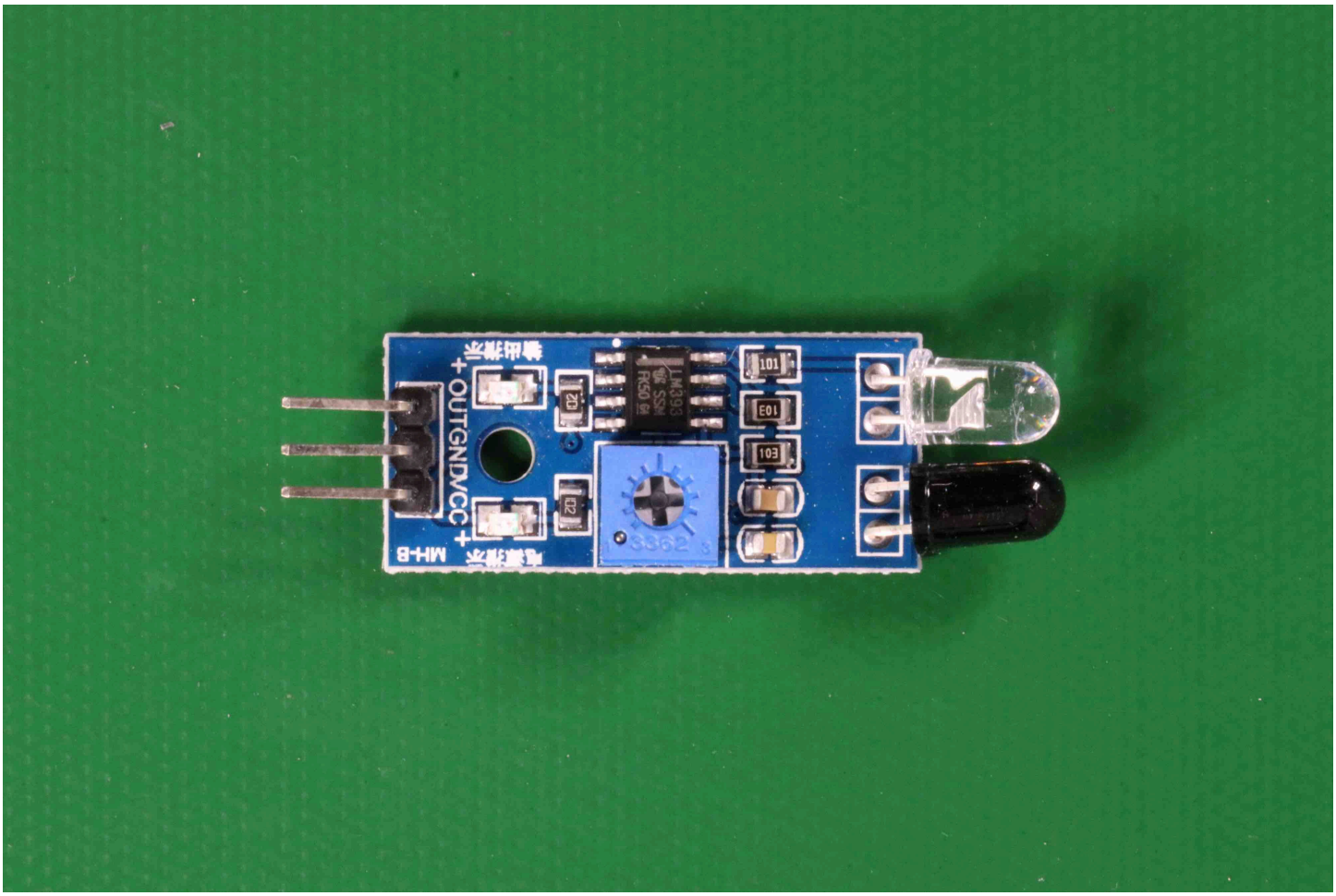
Classification d'images

Classification à plusieurs étiquettes

Détection des marques



Localisation d’objets

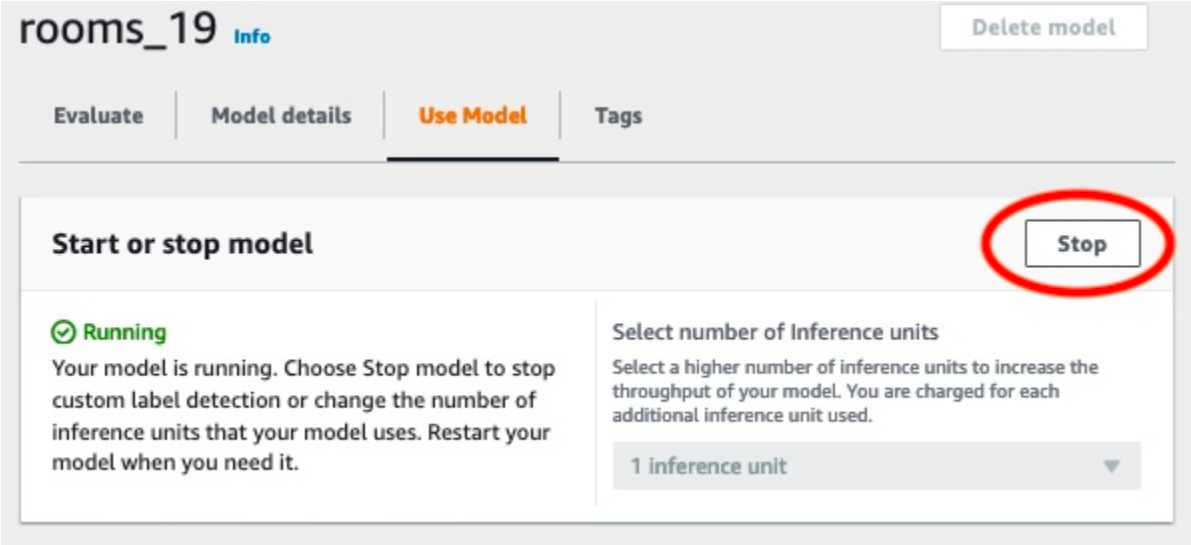


Étape 5 : Arrêter votre modèle

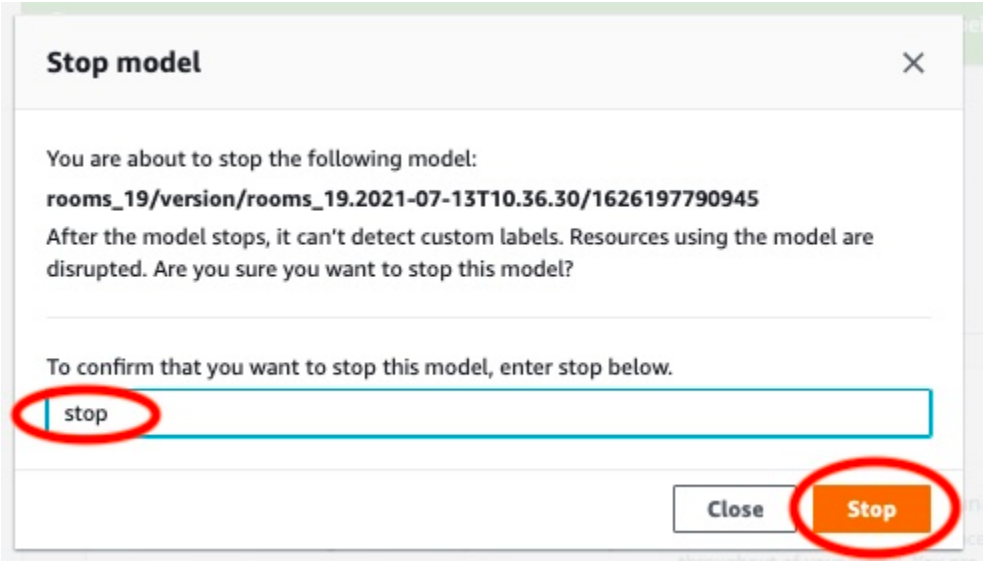
Au cours de cette étape, vous allez arrêter d’exécuter votre modèle. La durée de fonctionnement de votre modèle vous est facturée. Si vous avez fini d’utiliser le modèle, vous devez l’arrêter.

Pour arrêter votre modèle

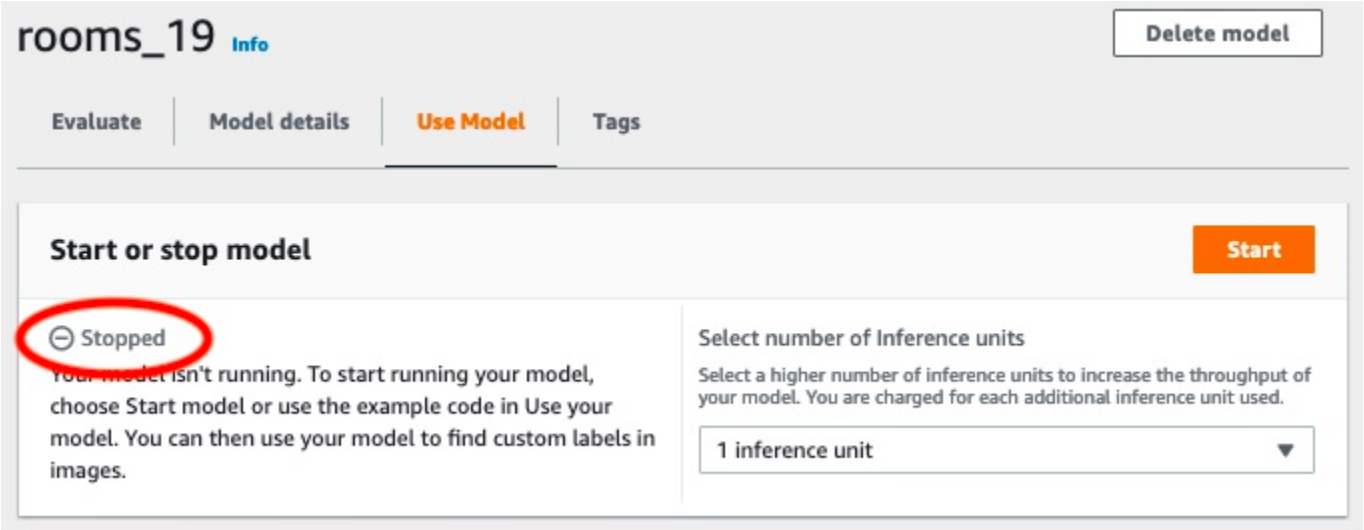
1. Dans la section Démarrer ou arrêter le modèle, choisissez Arrêter.



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. Le modèle est arrêté lorsque le statut indiqué dans la section Démarrer ou arrêter le modèle est Arrêté. Dans la capture d'écran suivante, la section Interface utilisateur permet de démarrer ou d'arrêter un modèle d'apprentissage automatique. L'état du modèle s'affiche comme « Arrêté » avec un bouton « Démarrer » pour démarrer le modèle et une liste déroulante pour sélectionner le nombre d'unités d'inférence.



Étape 6 : étapes suivantes

Après avoir terminé d'essayer les exemples de projets, vous pouvez utiliser vos propres images et jeux de données pour créer votre modèle. Pour plus d'informations, consultez Présentation de la fonctionnalité 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 d'images (Pièces)	1 étiquette au niveau de l'image par image	1 étiquette au niveau de l'image par image
Classification à plusieurs étiquettes (Fleurs)	Plusieurs étiquettes au niveau de l'image par image	Plusieurs étiquettes au niveau de l'image par image
Détection des marques (Logos)	étiquettes au niveau de l'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 Classification des images vous montre comment créer un projet, des jeux 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'entraînement, consultez Création d'un modèle Étiquettes personnalisées Amazon Rekognition.

Classification des images

Ce tutoriel explique comment créer le projet et les jeux de données d'un modèle qui classe les objets, les scènes et les concepts détectés dans une image. Le modèle classe l'image dans son intégralité. Par exemple, en suivant ce tutoriel, vous pouvez entraîner un modèle afin qu'il reconnaisse les pièces d'une maison (salon, cuisine, etc.). Le tutoriel explique également comment utiliser le modèle pour analyser des images.

Nous vous recommandons de lire Présentation de la fonctionnalité Étiquettes personnalisées Amazon Rekognition avant de commencer le tutoriel.

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

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

> **Note**
>
> Vous pouvez créer des modèles à différentes fins, par exemple pour rechercher l'emplacement d'objets sur une image. Pour plus d'informations, consultez Choix de votre type de modèle.

Étape 1 : Collecter vos images

Vous avez besoin de deux séries d'images. Une série à ajouter à votre jeu de données d'entraînement. Une autre série à ajouter à votre jeu de données de test. Les images doivent représenter les objets, les scènes et les concepts que vous souhaitez que votre modèle classe. L'image doit être au format JPEG ou PNG. Pour plus d'informations, consultez Préparation des images.

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

Si vous n'avez pas encore d'images, utilisez celles de l'exemple de projet de classification Pièces. Une fois le projet créé, les images d'entraînement et de test se trouvent dans les compartiments Amazon S3 suivants :

- Images d'entraînement : `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 la AWS région dans laquelle vous utilisez la console Amazon Rekognition Custom Labels. `numbers`est une valeur que la console attribue au nom du compartiment. `Version number`est le numéro de version de l'exemple de projet, à partir de 1.

La procédure suivante permet de stocker les images du projet Pièces dans des dossiers locaux de votre ordinateur nommés `training` et `test`.

Pour télécharger les fichiers image de l'exemple de projet Pièces

1. Créez le projet Pièces. Pour plus d'informations, consultez Étape 1 : Choisir un exemple de projet.
2. Ouvrez l'invite de commande et saisissez 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 commande, saisissez 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 d'entraînement vers un dossier distinct de votre choix. Vous allez utiliser les images pour essayer le modèle que vous avez entraîné à l'Étape 9 : Analyser une image avec votre modèle.

Étape 2 : Choisir vos classes

Dressez une liste des classes que vous souhaitez que votre modèle recherche. Par exemple, si vous apprenez à un modèle à reconnaître les pièces d'une maison, vous pouvez classer l'image suivante dans `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 jeux de données d'entraînement et de test.

Si vous utilisez les images de l'exemple de projet Pièces, les étiquettes au niveau de l'image sont backyard (jardin arrière), bathroom (salle de bains), bedroom (chambre à coucher), closet (placard), entry_way (entrée), floor_plan (plan d'étage), front_yard (jardin avant), kitchen (cuisine), living_space (pièce de vie) et patio.

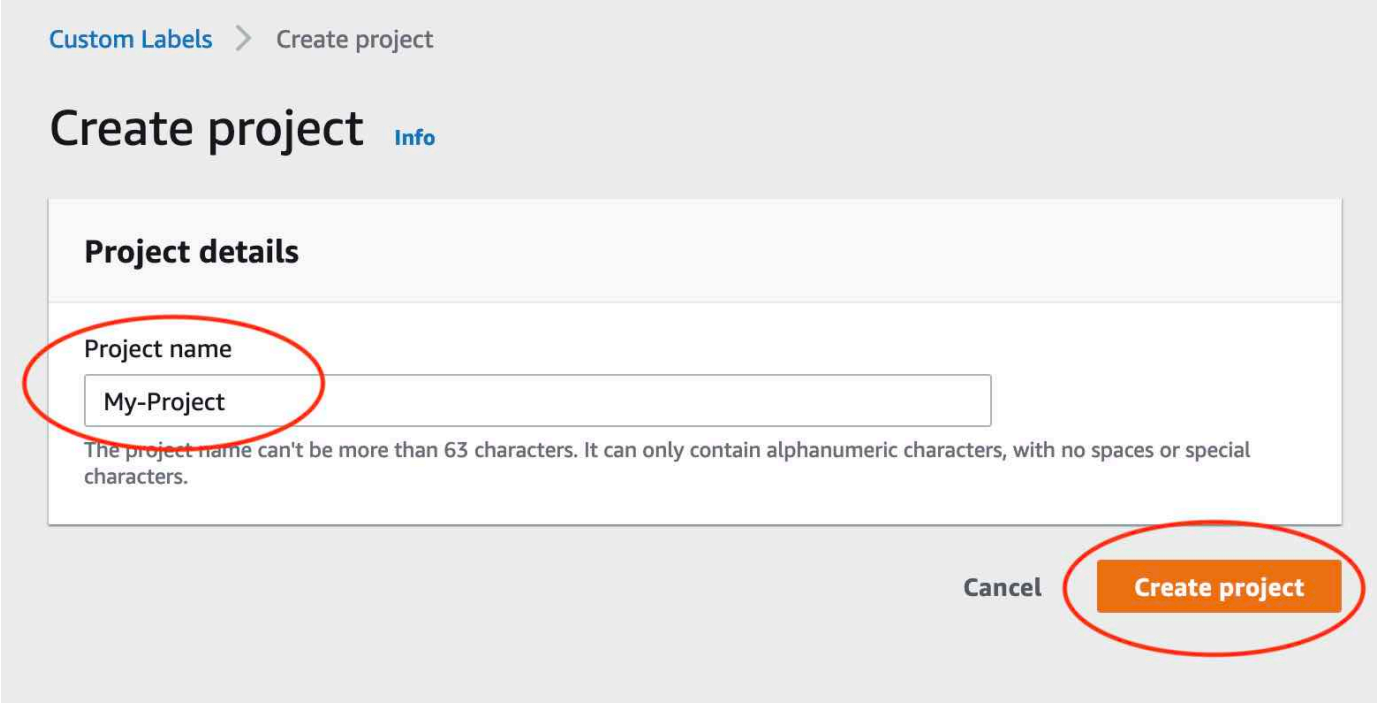
Étape 3 : Créer un projet

Pour gérer vos jeux de données et vos modèles, vous devez créer un projet. Chaque projet doit porter sur un cas d'utilisation unique (reconnaissance des pièces d'une maison, par exemple).

Pour créer un projet (console)

1. Si vous ne l'avez pas encore fait, configurez la console Étiquettes personnalisées Amazon Rekognition. Pour de plus amples informations, veuillez consulter Configuration d'Étiquettes personnalisées Amazon Rekognition.
2. Connectez-vous à la console Amazon Rekognition AWS Management Console et ouvrez-la à l'adresse. https://console.aws.amazon.com/rekognition/

3. Dans le volet de gauche, choisissez Utiliser Custom Labels. La page d'accueil d'Étiquettes personnalisées Amazon Rekognition s'affiche.
4. Sur la page d'accueil d'Étiquettes personnalisées Amazon Rekognition, choisissez Commencer
5. Dans le volet de navigation de gauche, choisissez Projets.
6. Sur la page Projets, choisissez Créer un projet.
7. Dans Project name (Nom de projet), saisissez un nom pour votre projet.
8. Choisissez Créer un projet pour créer votre projet.



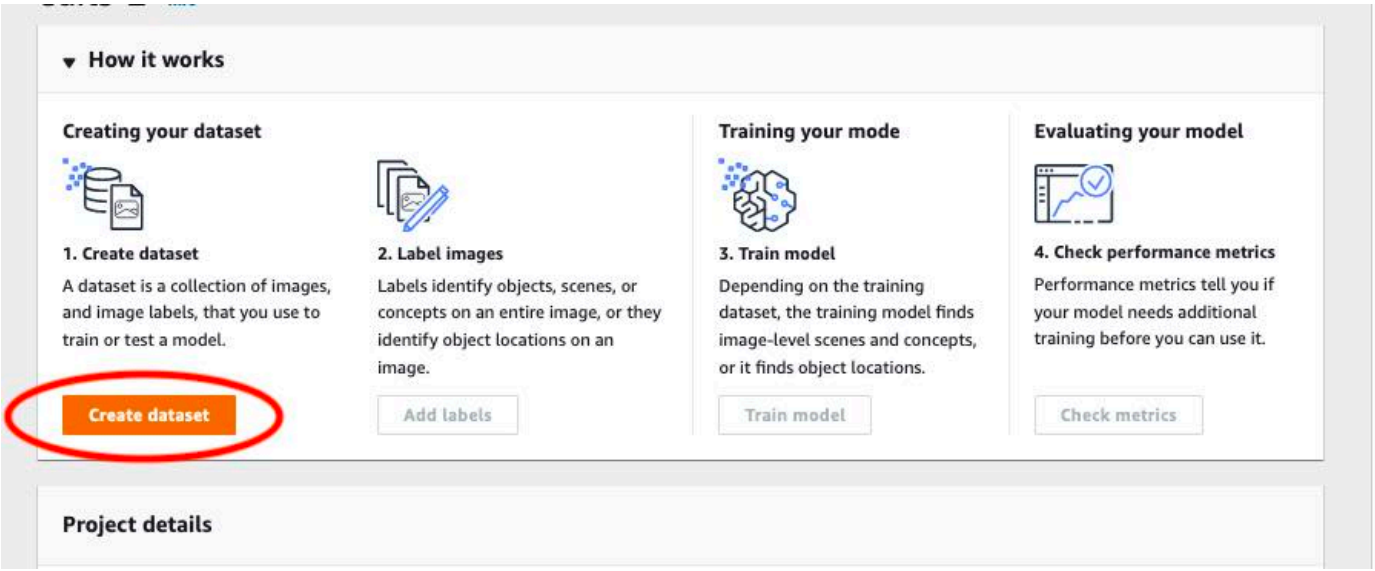
Étape 4 : Créer des jeux de données d'entraînement et de test

Dans cette étape, vous allez créer un jeu de données d'entraînement et un jeu de données de test en chargeant des images depuis votre ordinateur local. Vous pouvez charger jusqu'à 30 images à la fois. Si vous avez beaucoup d'images à charger, vous pouvez les importer depuis un compartiment Amazon S3. Pour plus d'informations, consultez Importation d'images depuis un compartiment Amazon S3.

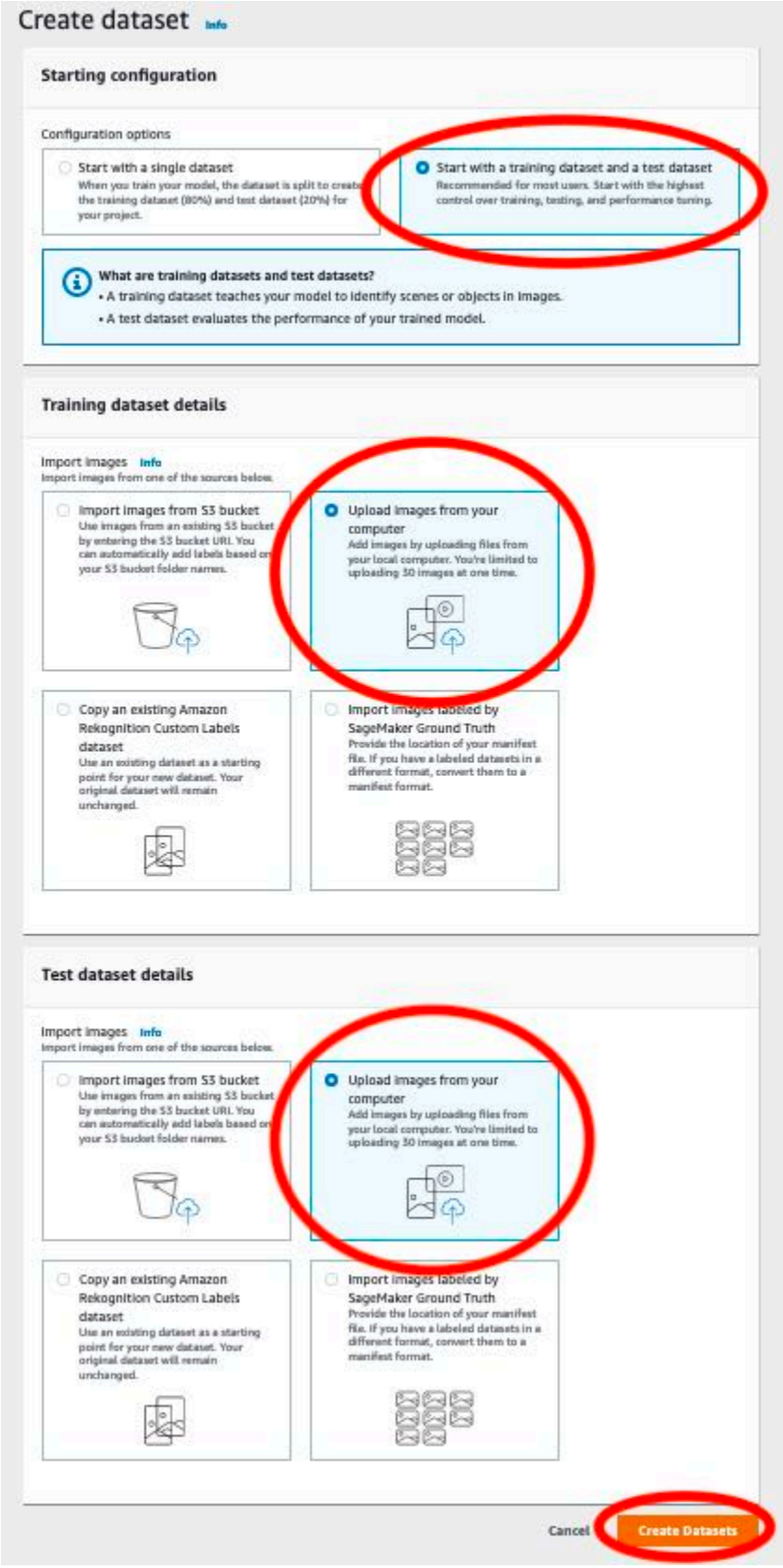
Pour plus d'informations sur les jeux de données, consultez Gestion des jeux de données.

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

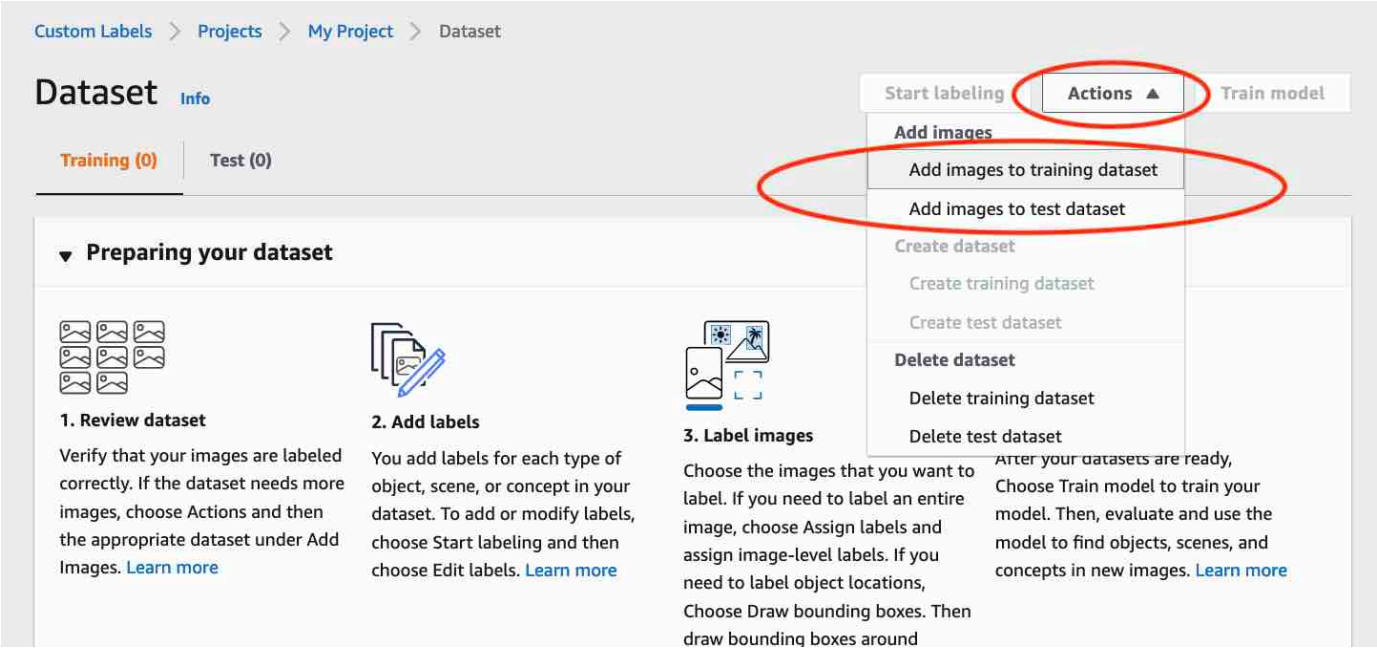
1. Sur la page des détails du projet, choisissez Créer un jeu de données.



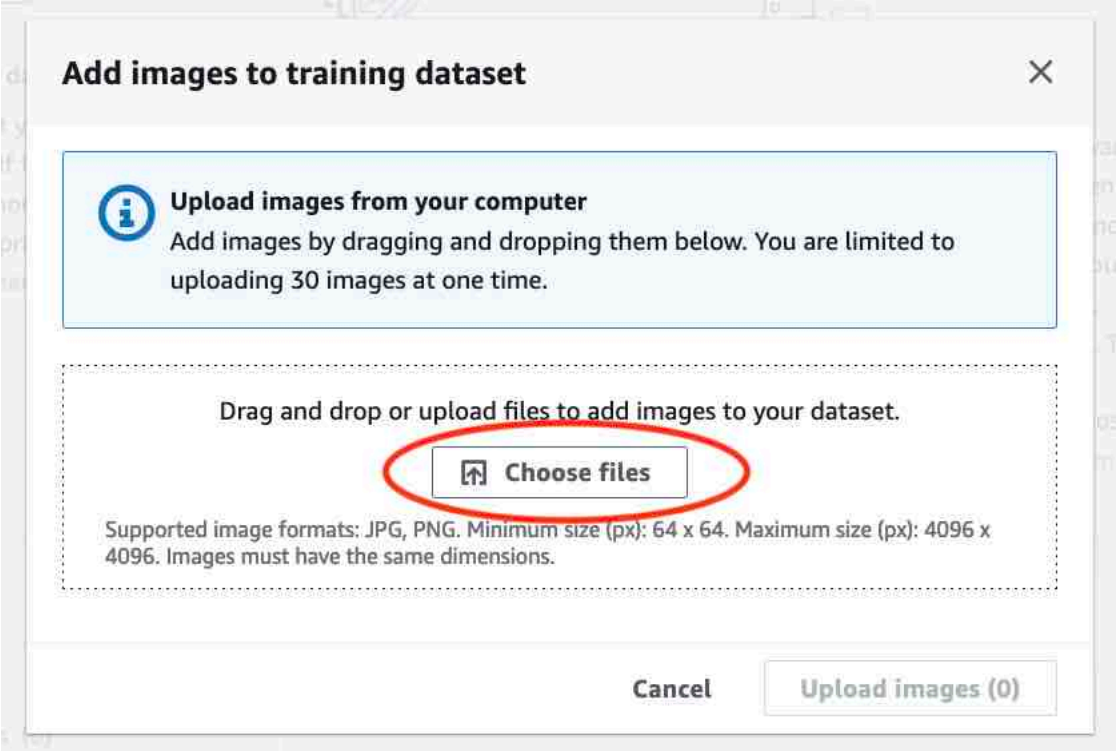
2. Dans la section Démarrage de la configuration, choisissez Démarrer avec un jeu de données d'entraînement et un jeu de données de test.
3. Dans la section Détails du jeu de données d'entraînement, choisissez Charger des images depuis votre ordinateur.
4. Dans la section Détails du jeu de données de test, choisissez Charger des images depuis votre ordinateur.
5. Choisissez Créer des 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 30 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 30 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
Create Datasets

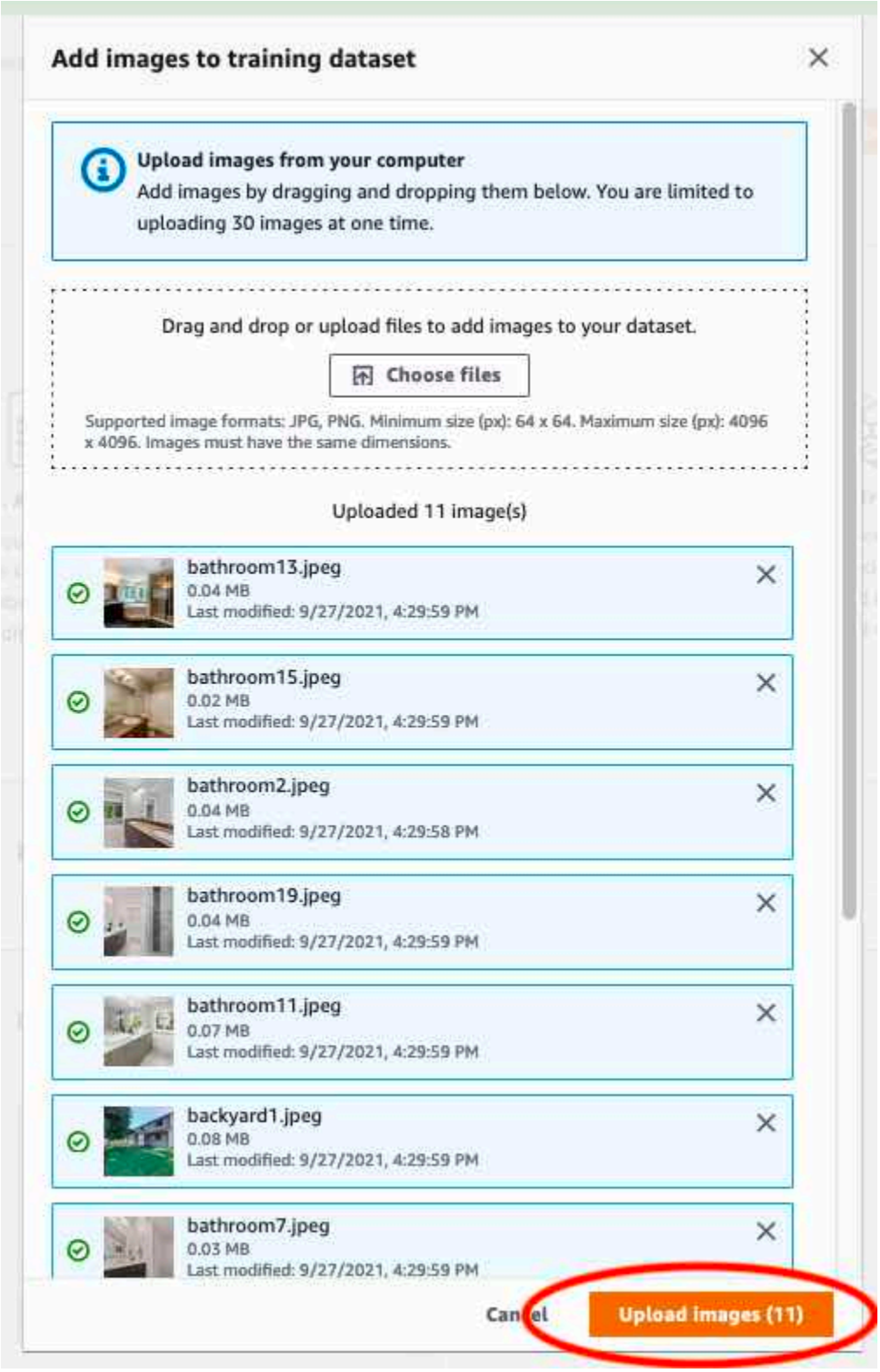
6. Une page s'affiche avec un onglet Entraînement et un onglet Test pour vos jeux de données respectifs.
7. Sur la page du jeu de données, choisissez l'onglet Entraînement.
8. Choisissez Actions, puis sélectionnez Ajouter des images au jeu de données d'entraînement.



9. Dans la boîte de dialogue Ajouter des images au jeu de données d'entraînement, choisissez Choisir des fichiers.



10. Choisissez les images que vous souhaitez charger dans le jeu de données. Vous pouvez charger jusqu'à 30 images à la fois.
11. Choisissez Charger des images. Étiquettes personnalisées Amazon Rekognition peut avoir besoin de quelques instants pour ajouter les images dans le jeu de données.



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.

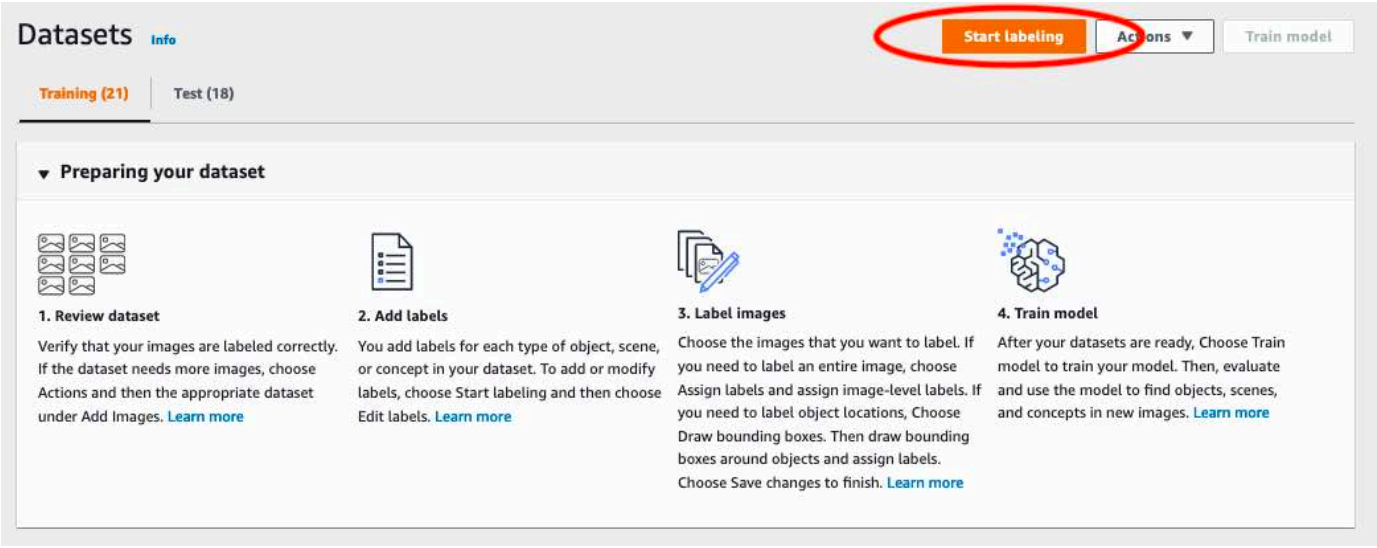
14. Répétez les étapes 8 à 12 pour ajouter des images au jeu de données de test. Pour l'étape 8, choisissez Actions, puis sélectionnez Ajouter des images au jeu de données de test.

Étape 5 : Ajouter des étiquettes au projet

Au cours de cette étape, vous allez ajouter une étiquette au projet pour chacune des classes que vous avez identifiées à l'étape Étape 2 : Choisir vos classes.

Pour ajouter une nouvelle étiquette (console)

1. Sur la page de la galerie de jeux de données, choisissez Commencer l'étiquetage pour passer en mode d'étiquetage.



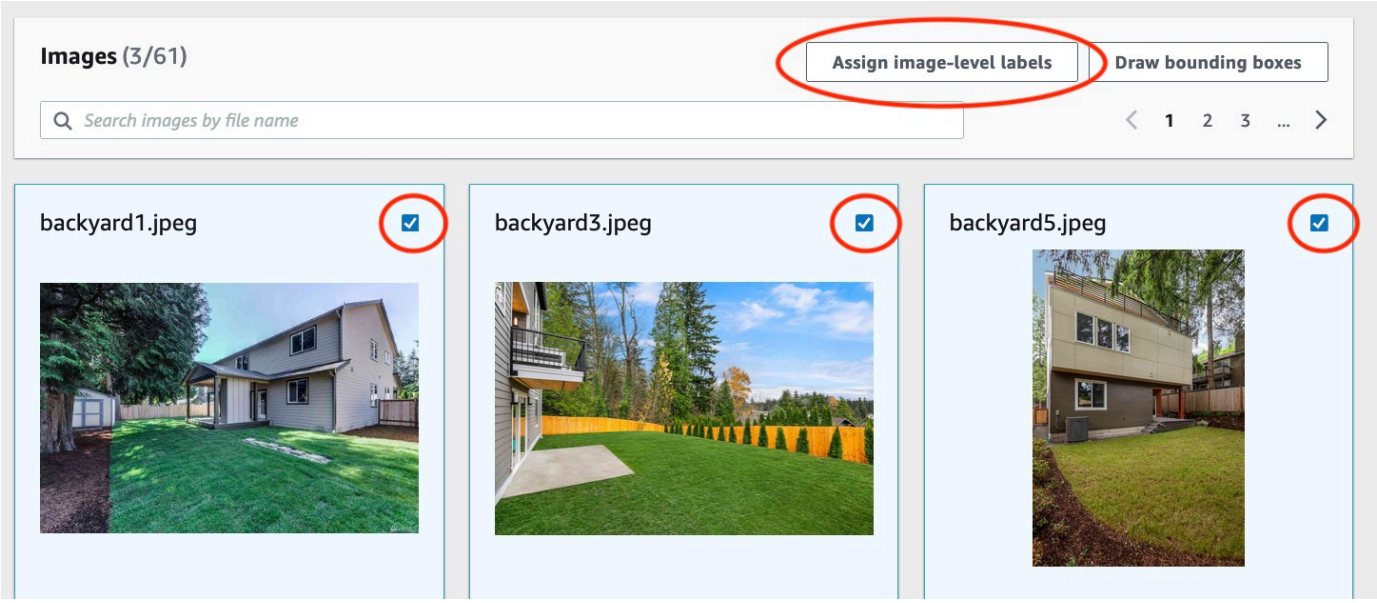
2. Dans la section Étiquettes de la galerie de jeux de données, choisissez Modifier les étiquettes pour ouvrir la boîte de dialogue Gérer les étiquettes.
3. Dans la zone d'édition, saisissez un nouveau nom d'étiquette.
4. Choisissez Ajouter une étiquette.
5. Répétez les étapes 3 et 4 jusqu'à ce que vous 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 jeux de données d’entraînement et de test

Au cours de cette étape, vous allez attribuer un seul niveau d’image à chaque image de vos jeux de données d’entraînement et de test. L’étiquette au niveau de l’image correspond à la classe que chaque image représente.

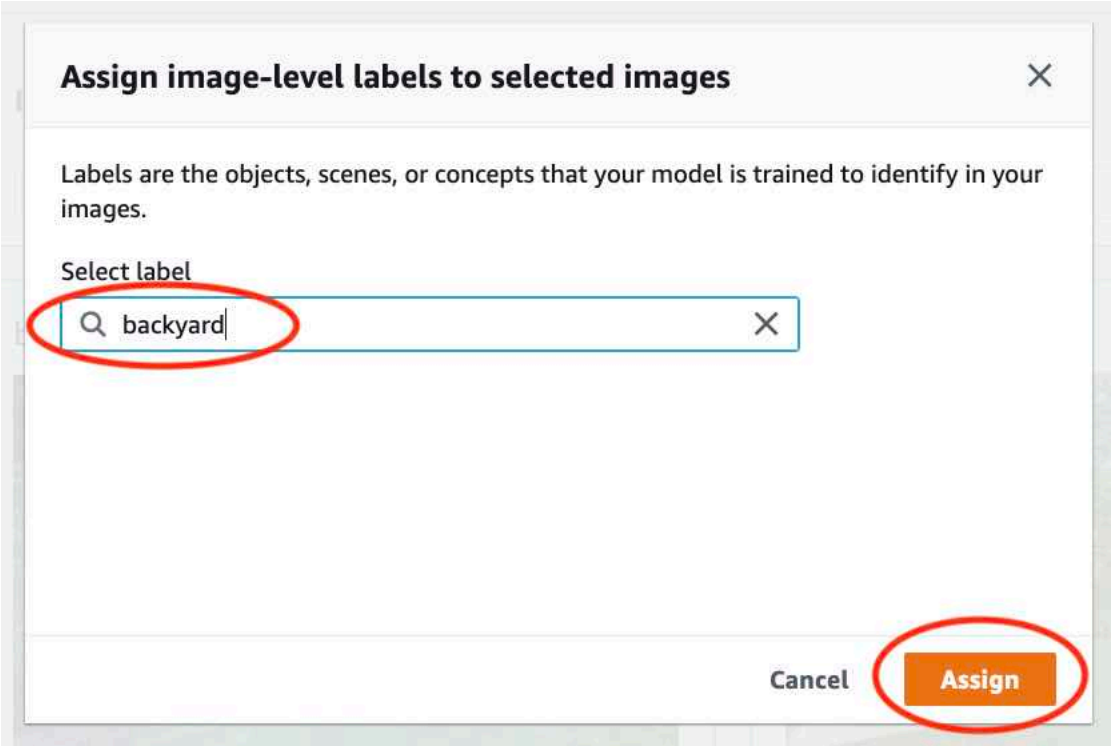
Pour attribuer des étiquettes au niveau de l’image à une image (console)

1. Sur la page Ensembles de données, choisissez l’onglet Entraînement.
2. Choisissez Commencer l’étiquetage pour passer en mode d’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 Maj et maintenez-la enfoncée.
   c. Sélectionnez la deuxième image. Les images situées entre la première et la deuxième image sont également sélectionnées.
   d. Relâchez la touche Maj.
4. Choisissez Attribuer des étiquettes au niveau de l’image.



5. Dans la boîte de dialogue Attribuer une étiquette au niveau de l’image aux images sélectionnées, sélectionnez l’étiquette que vous souhaitez attribuer à l’image ou aux images.

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



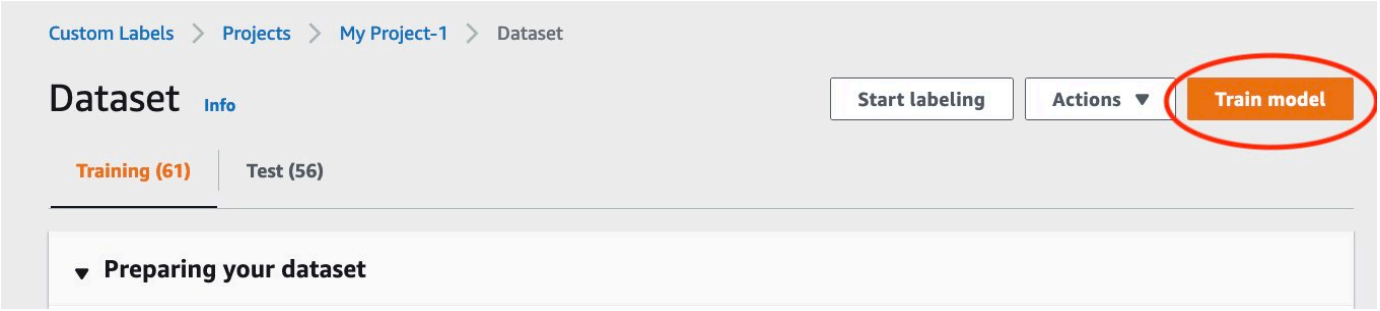
7. Répétez l'étiquetage jusqu'à ce que chaque image soit annotée avec les étiquettes requises.
8. Choisissez l'onglet Test.
9. Répétez les étapes pour attribuer des étiquettes au niveau de l'image aux images du jeu de données de test.

Étape 7 : Entraîner votre modèle

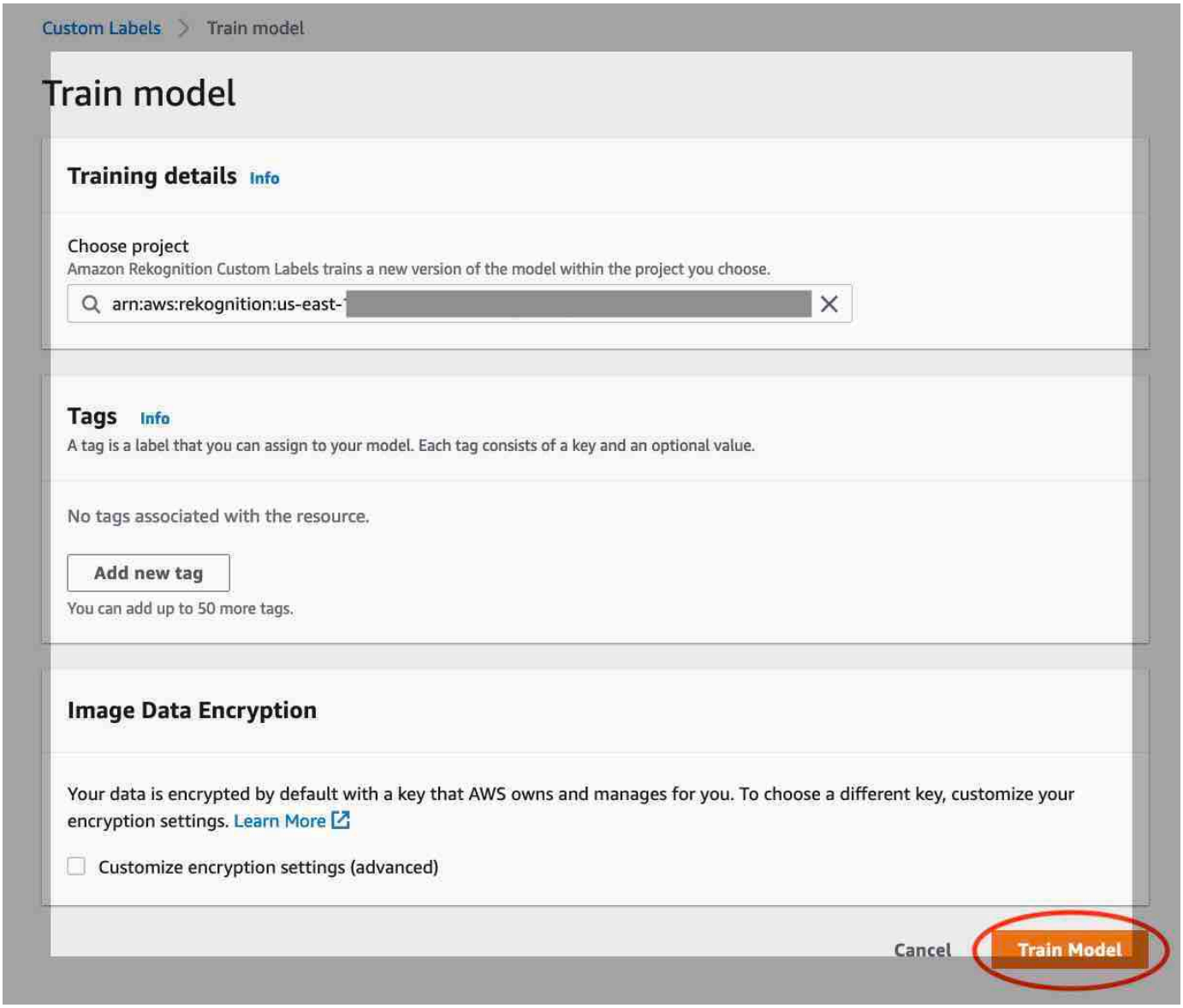
Suivez les étapes suivantes pour entraîner votre modèle. Pour plus d'informations, consultez Entraînement d'un modèle Étiquettes personnalisées Amazon Rekognition.

Pour entraîner votre modèle (console)

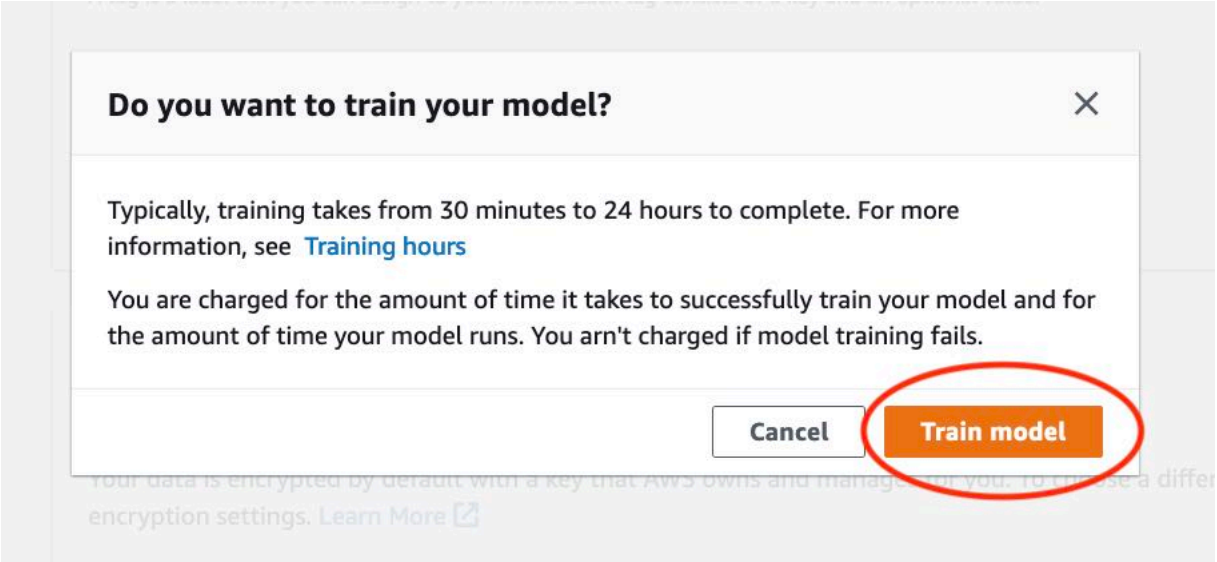
1. Sur la page Ensemble de données, sélectionnez Entraîner un modèle.



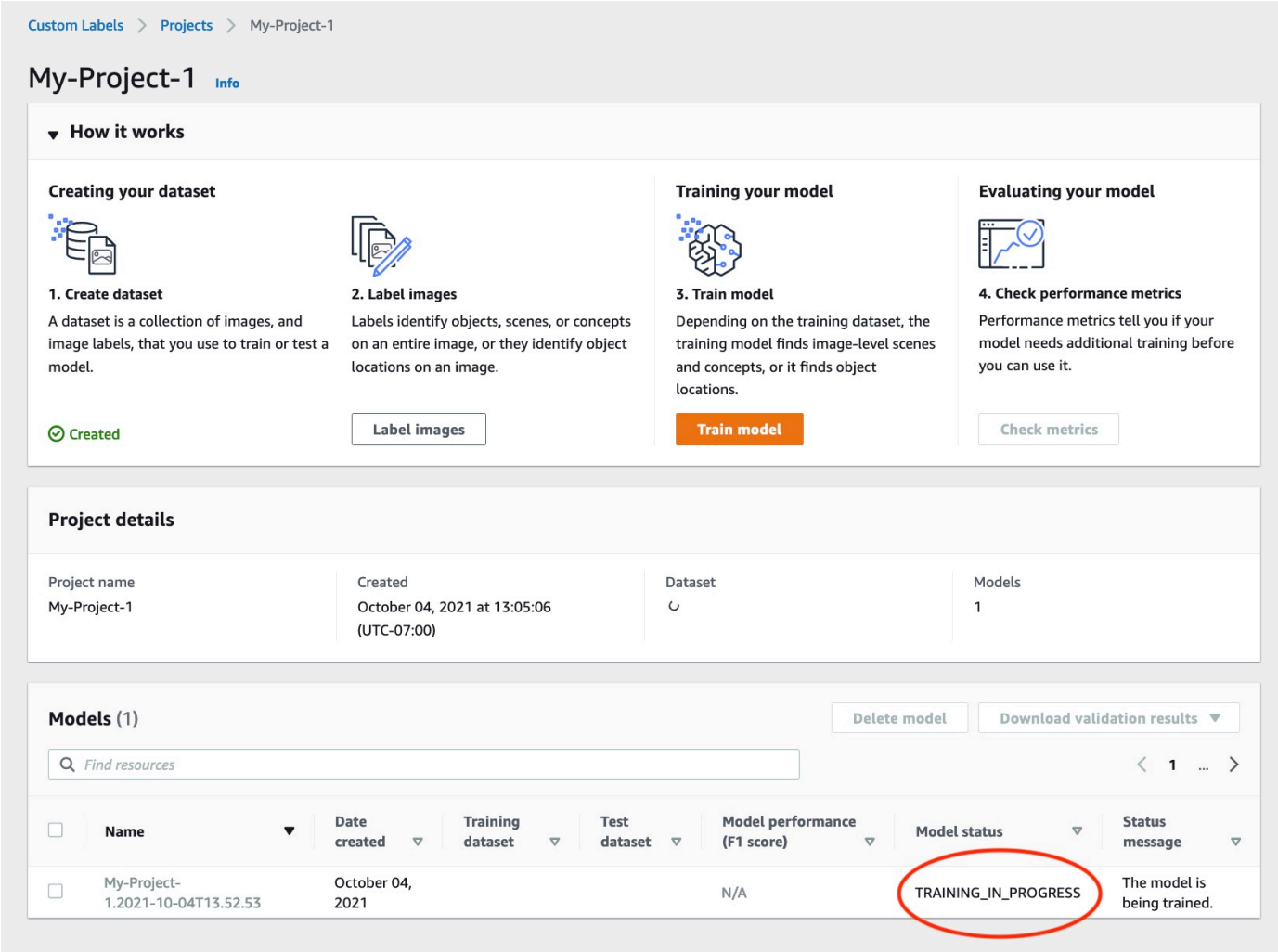
2. Sur la page Entraîner un modèle, choisissez Entraîner un modèle. L'Amazon Resource Name (ARN) de votre projet se trouve dans la zone d'édition Choisir un projet.



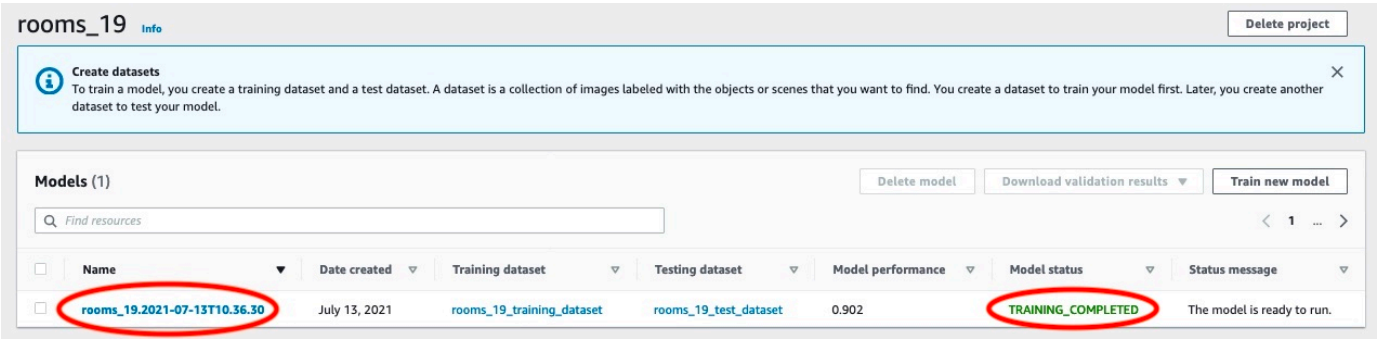
3. Dans la boîte de dialogue Voulez-vous entraîner votre modèle ?, choisissez Entraîner un modèle.



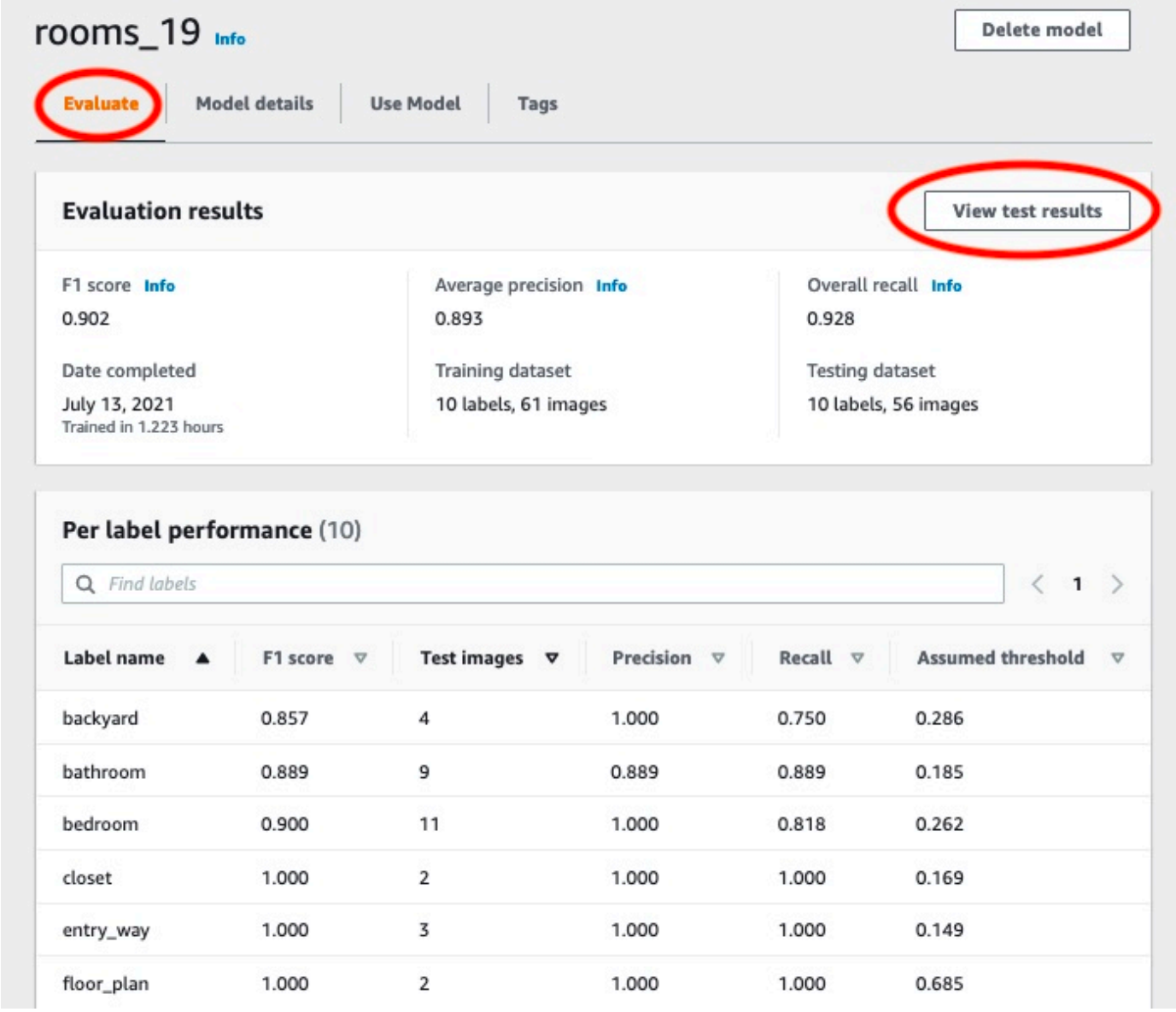
4. Dans la section Modèles de la page du projet, vous pouvez voir que l'entraînement est en cours. Vous pouvez vérifier le statut de l'entraînement dans la colonne `Model Status` de la version du modèle. L'entraînement d'un modèle peut nécessiter un certain temps.



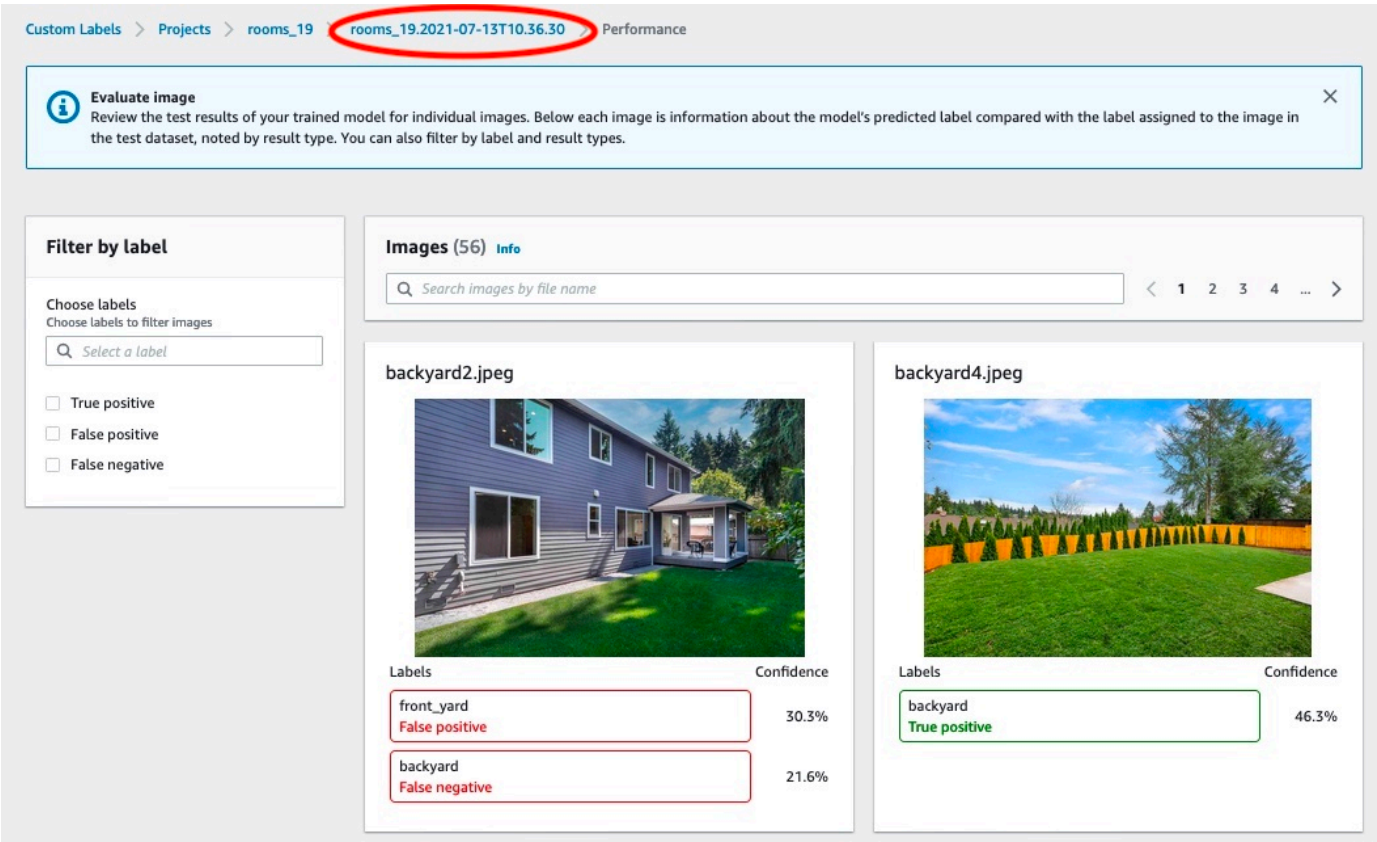
5. Une fois l'entraînement terminé, choisissez le nom du modèle. L'entraînement est terminé lorsque le statut du modèle est TRAINING_COMPLETED.



6. Cliquez sur le bouton Évaluer pour voir les résultats de l'évaluation. Pour plus d'informations sur l'évaluation d'un modèle, consultez Amélioration d'un modèle entraîné Étiquettes personnalisées Amazon Rekognition.
7. Choisissez Afficher les résultats des tests pour visualiser les résultats des images des tests. Pour de plus amples informations, veuillez consulter Métriques d'évaluation du modèle.



8. Après avoir consulté les résultats des tests, choisissez le nom du modèle pour revenir à la page du modèle.



Étape 8 : Démarrer votre modèle

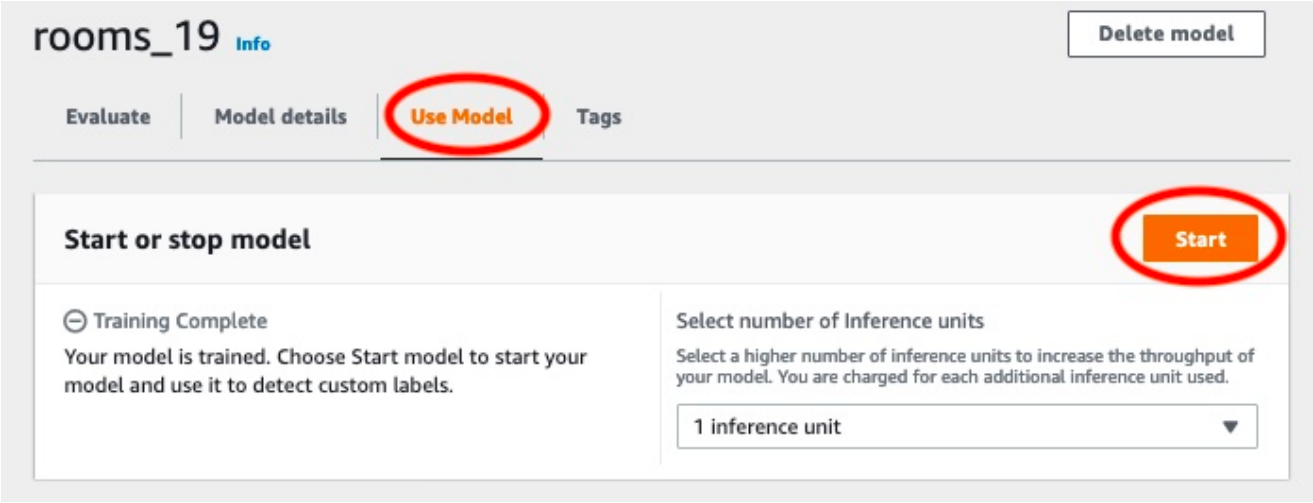
Au cours de cette étape, vous allez démarrer votre modèle. Une fois votre modèle démarré, vous pouvez l'utiliser pour analyser des images.

La durée de fonctionnement de votre modèle vous est facturée. Arrêtez votre modèle si vous n'avez pas besoin d'analyser d'images. Vous pourrez le redémarrer ultérieurement. Pour plus d'informations, consultez Exécution d'un modèle Étiquettes personnalisées Amazon Rekognition entraîné.

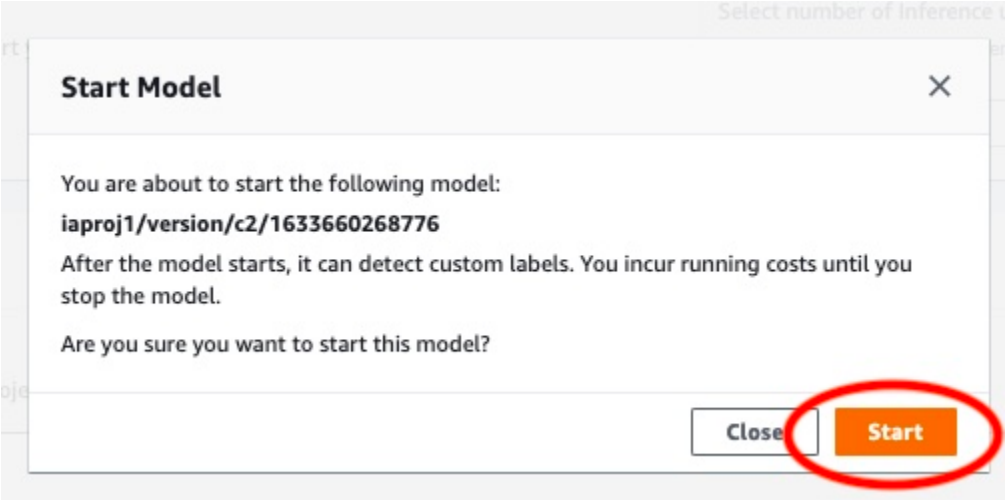
Pour démarrer votre modèle

1. Choisissez l'onglet Utiliser le modèle sur la page du modèle.
2. Dans la section Démarrer ou arrêter le modèle, procédez comme suit :

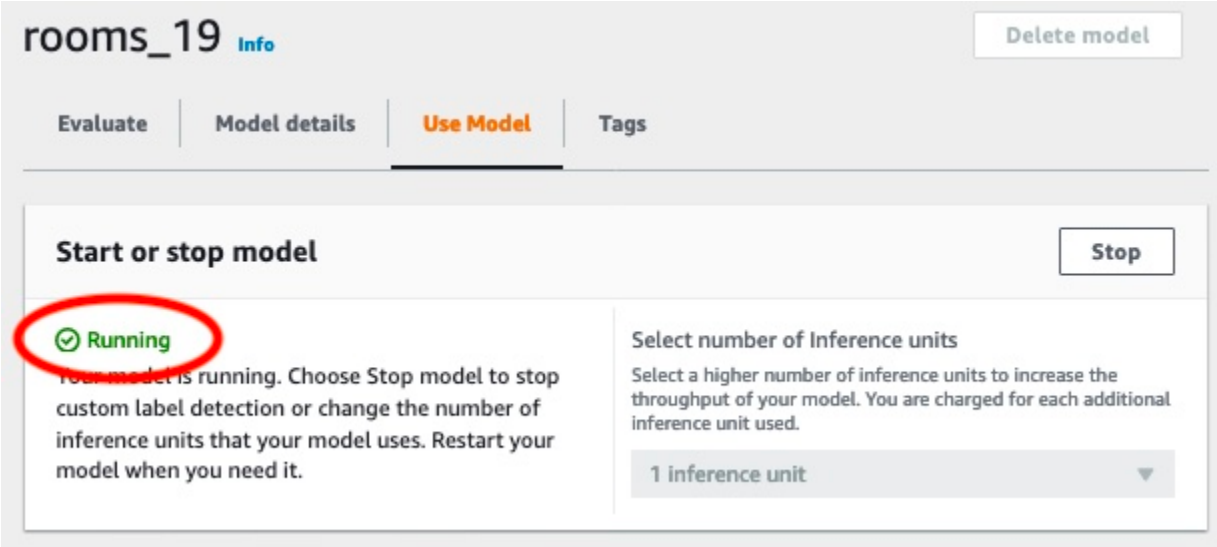
    a. Sélectionnez Démarrer.



b. Dans la boîte de dialogue Démarrer le modèle, choisissez Démarrer.



3. Patientez jusqu'à l'exécution du modèle. Le modèle est en cours d'exécution lorsque le statut indiqué dans la section Démarrer ou arrêter le modèle est En cours d'exécution.



Étape 9 : Analyser une image avec votre modèle

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

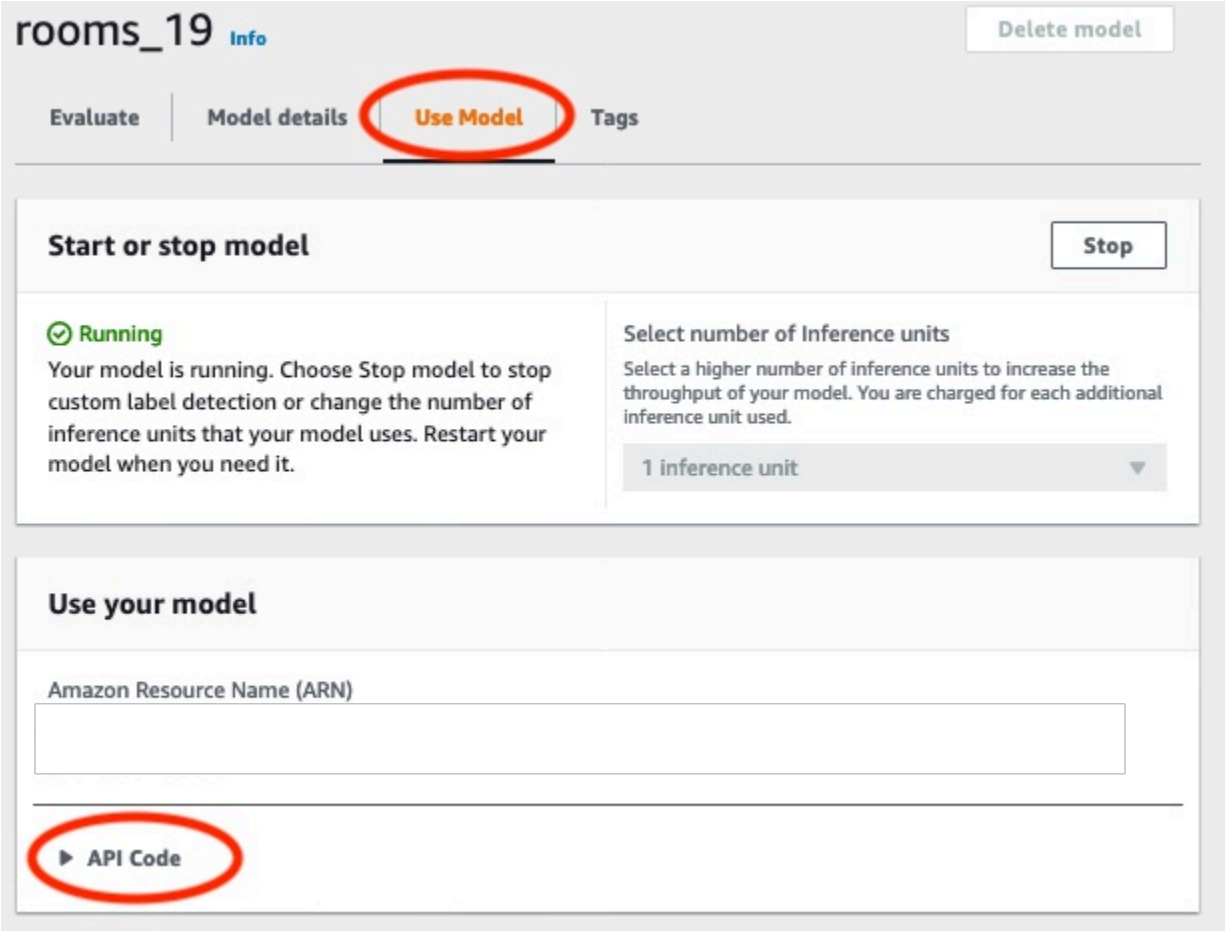
> **Note**
>
> La console fournit également un exemple de code Python.

Le résultat de `detect-custom-labels` inclut la liste des étiquettes détectées dans l'image, les cadres de délimitation (si le modèle recherche les emplacements d'objets) et le score de confiance du modèle concernant la précision des prédictions.

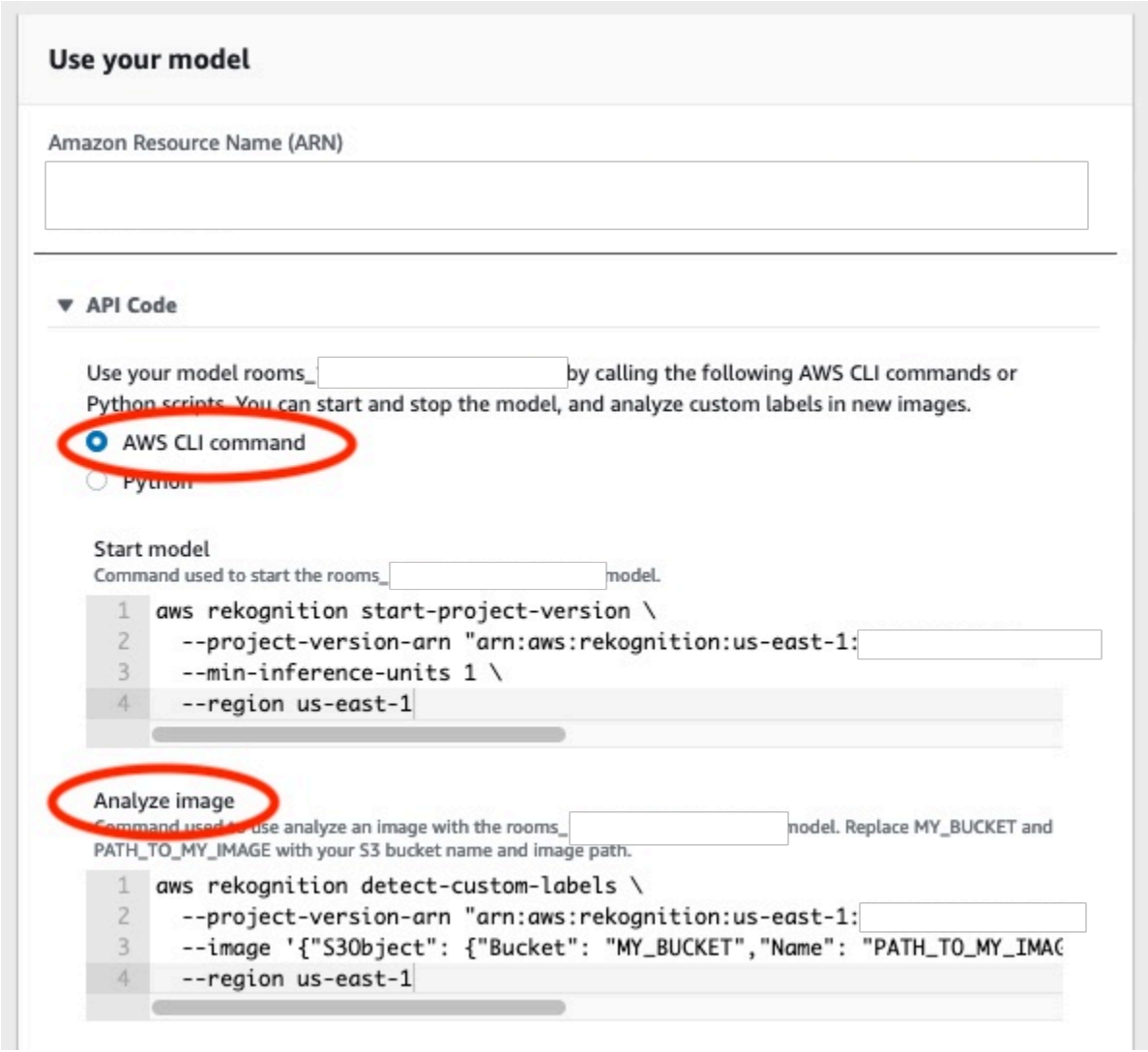
Pour plus d'informations, consultez Analyse d'une image avec un modèle entraîné.

Pour analyser une image (console)

1. Si ce n'est pas déjà fait, configurez le AWS CLI. Pour obtenir des instructions, veuillez consulter the section called "Étape 4 : Configurez le AWS CLI et AWS SDKs".
2. Choisissez l'onglet Utiliser le modèle, puis Code de l'API.



3. Choisissez Commande de l’interface de ligne de commande AWS.
4. Dans la section Analyser l'image, copiez la AWS CLI commande qui appelle`detect-custom-labels`.



5. Chargez une image dans un compartiment Amazon S3. Pour en savoir plus, consultez Chargement d'objets dans Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service. Si vous utilisez des images du projet Pièces, utilisez l'une des images que vous avez déplacées dans un dossier distinct à l'Étape 1 : Collecter vos images.

6. À l'invite de commande, entrez la AWS CLI commande que vous avez copiée à l'étape précédente. Elle doit ressembler à l'exemple suivant.

   La valeur de `--project-version-arn` doit être l'Amazon Resource Name (ARN) de votre modèle. La valeur de `--region` doit être la région AWS dans laquelle vous avez créé le modèle.

   Remplacez `MY_BUCKET` et `PATH_TO_MY_IMAGE` par le compartiment Amazon S3 et l'image que vous avez utilisés à l'étape précédente.

Si vous utilisez le custom-labels-accessprofil 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 la commande AWS CLI doit ressembler à ce qui suit. `Name` est le nom de l'étiquette au niveau de l'image détectée par le modèle. `Confidence` (0-100) est le niveau de confiance du modèle dans l'exactitude de la prédiction.

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

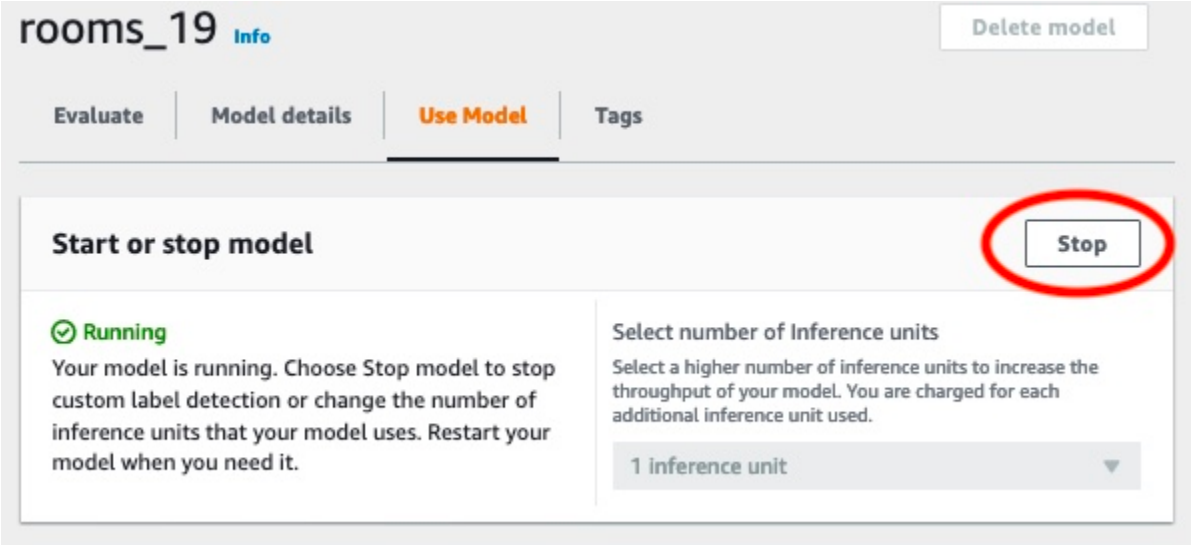
7. Continuez à utiliser le modèle pour analyser d'autres images. Arrêtez le modèle si vous ne l'utilisez plus.

Étape 10 : Arrêter votre modèle

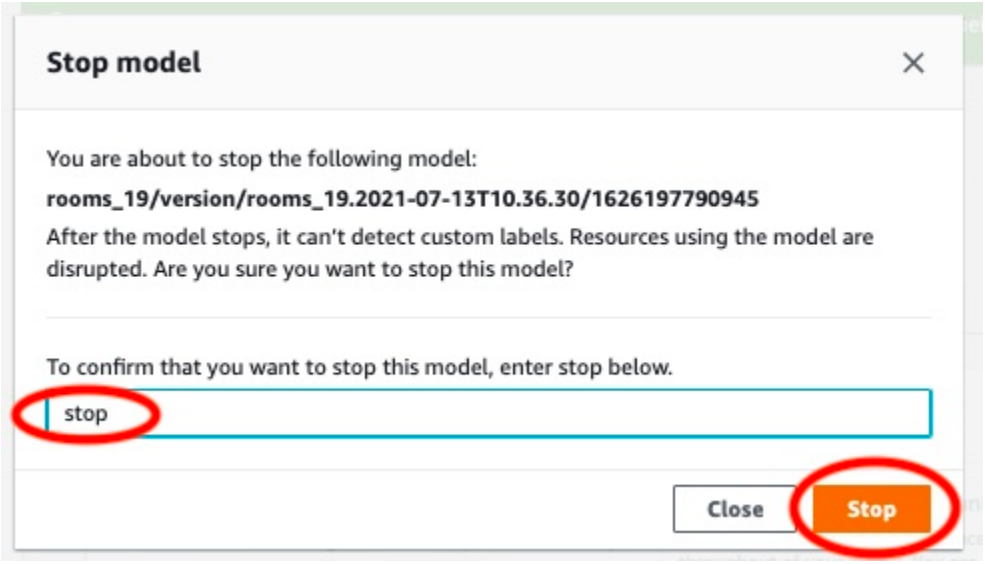
Au cours de cette étape, vous allez arrêter d'exécuter votre modèle. La durée de fonctionnement de votre modèle vous est facturée. Si vous avez fini d'utiliser le modèle, vous devez l'arrêter.

Pour arrêter votre modèle

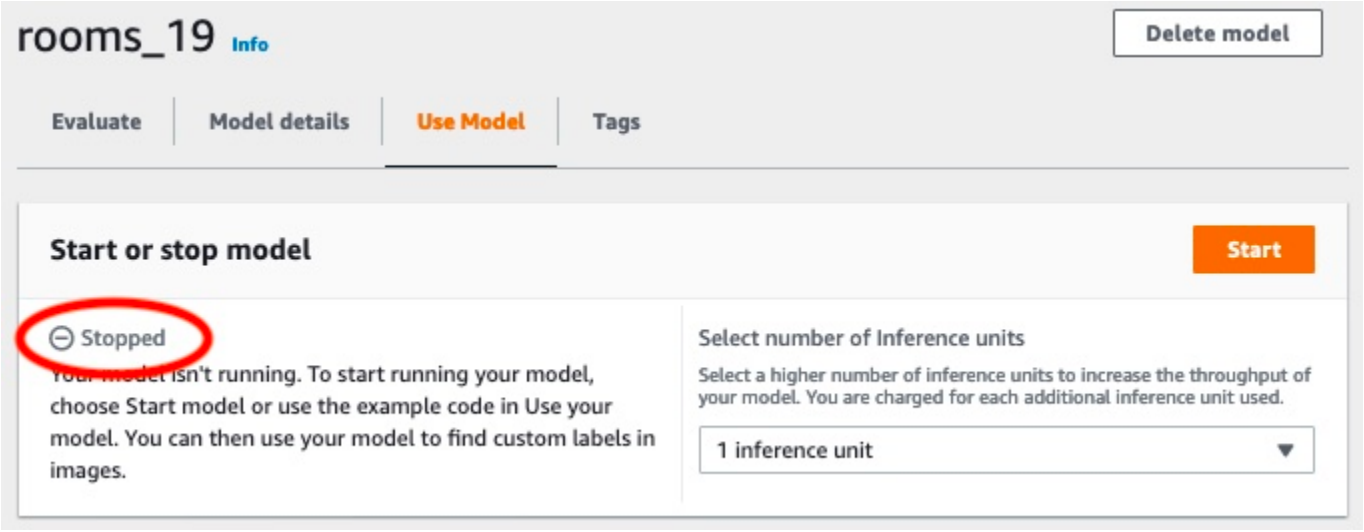
1. Dans la section Démarrer ou arrêter le modèle, choisissez Arrêter.



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. Le modèle est arrêté lorsque le statut indiqué dans la section Démarrer ou arrêter le modèle est Arrêté.


rooms_19 Info
Delete model
Evaluate
Model details
Use Model
Tags
Start or stop model
Start
Stopped
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

Création d’un modèle Étiquettes personnalisées Amazon Rekognition

Un modèle est le logiciel que vous entraînez à rechercher des concepts, des scènes et des objets propres à votre entreprise. Vous pouvez créer un modèle à l'aide de la console Amazon Rekognition Custom Labels ou du SDK. AWS Avant de créer un modèle Étiquettes personnalisées Amazon Rekognition, nous vous recommandons de lire Présentation de la fonctionnalité Étiquettes personnalisées Amazon Rekognition.

Cette section fournit des informations sur la console et le kit SDK concernant la création d’un projet et de jeux de données d’entraînement et de test pour différents types de modèles, ainsi que l’entraînement d’un modèle. Les sections suivantes vous montrent comment améliorer et utiliser votre modèle. Pour un didacticiel qui vous montre comment créer et utiliser un type spécifique de modèle avec la console, consultez Classification des images.

Rubriques

Création d’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 à l’aide de la console Étiquettes personnalisées Amazon Rekognition ou de l’API. Pour les autres tâches du projet, telles que la suppression d’un projet, consultez Gestion d’un projet Étiquettes personnalisées Amazon Rekognition.

Vous pouvez utiliser des balises pour classer et gérer vos ressources Amazon Rekognition Custom Labels, y compris vos projets.

L'CreateProjectopération vous permet de spécifier éventuellement des balises lors de la création d'un nouveau projet, en fournissant les balises sous forme de paires clé-valeur que vous pouvez utiliser pour classer et gérer vos ressources.

Création d’un projet Étiquettes personnalisées Amazon Rekognition (console)

Vous pouvez utiliser la console Étiquettes personnalisées Amazon Rekognition 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 console) dans votre compte. AWS Le compartiment est utilisé pour stocker les fichiers du projet. Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition, à moins que le compartiment de console soit créé.

Vous pouvez utiliser la console Étiquettes personnalisées Amazon Rekognition pour créer un projet.

Pour créer un projet (console)

1. Connectez-vous à la console Amazon Rekognition AWS Management Console et ouvrez-la 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. Sur la page d’accueil d’Étiquettes personnalisées Amazon Rekognition, choisissez Commencer.
4. Dans le volet de gauche, choisissez Projets.
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 dans Création de jeux de données d’entraînement et de test pour créer les jeux de données d’entraînement et de test pour votre projet.

Création d’un projet Étiquettes personnalisées Amazon Rekognition (kit SDK)

Vous créez un projet d'étiquettes personnalisées Amazon Rekognition en appelant. CreateProject La réponse est un Amazon Resource Name (ARN) qui identifie le projet. Après avoir créé un projet, vous créez des jeux de données pour l’entraînement et le test d’un modèle. Pour plus d’informations, consultez Création de jeux de données d’entraînement et de test avec des images.

Pour créer un projet (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez le code suivant pour créer un projet.

   AWS CLI

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

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

```
aws rekognition create-project --project-name my_project \
 --profile custom-labels-access --"CUSTOM_LABELS" --
tags'{"key1":"value1","key2":"value2"}'
```

   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 prooject.
    """
```

```
    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'arguments 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 affiché dans la réponse. Vous aurez besoin de créer un modèle.
4. Suivez les étapes décrites dans Création de jeux de données d'entraînement et de test (kit SDK) pour créer les jeux de données d'entraînement et de test pour votre projet.

CreateProject demande d'opération

Le format de la demande d' CreateProject opération est le suivant :

```
{
 "AutoUpdate": "string",
 "Feature": "string",
 "ProjectName": "string",
 "Tags": {
 "string": "string"
 }
}
```

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 ensembles de données à l'aide de la console Amazon Rekognition Custom Labels ou du SDK. AWS Avant de créer un jeu de données, nous vous recommandons de lire Présentation de la fonctionnalité 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 de plus amples informations, veuillez consulter Création de jeux de données d'entraînement et de test avec des images. Si vous utilisez le AWS SDK, consultezCréation 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 des emplacements 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.

Classification d'images

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 Classification d'images.

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 des images à plusieurs étiquettes.

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 Importation d'images depuis un 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 des emplacements 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 les emplacements d'objets, 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

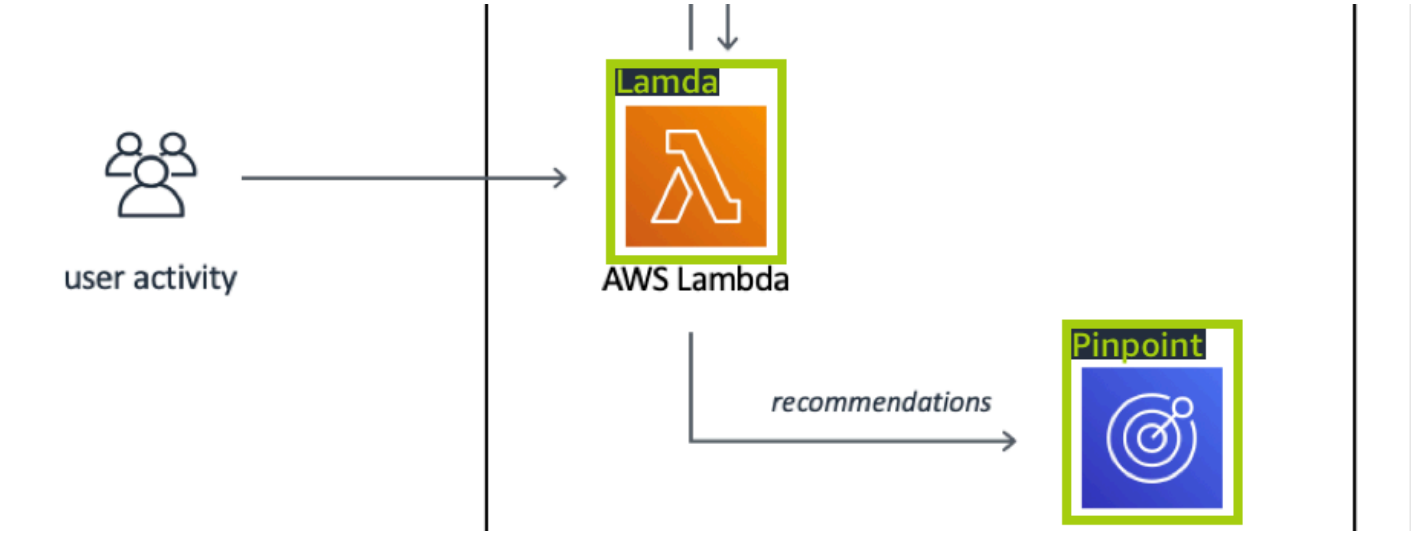
Lorsque vous créez le jeu de données, vous pouvez inclure des informations relatives aux cadres de délimitation pour vos images. Par exemple, vous pouvez importer un fichier manifeste au format SageMaker AI Ground Truth contenant des cadres de délimitation. Vous pouvez également ajouter des cadres de 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 montre 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 des marques montre comment Étiquettes personnalisées Amazon Rekognition utilise des cadres de délimitation étiquetés pour entraîner un modèle à rechercher les emplacements 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 d'emplacement d'objets.

exemple	Images d'entraînement	Images de test
Classification d'images	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 des emplacements 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 Directives et quotas dans Étiquettes personnalisées 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 :

- Importation d'images depuis un compartiment Amazon S3
- Importation d'images depuis un ordinateur local

- Utilisation d'un fichier manifeste pour importer des images
- Copier le contenu d'un ensemble de données existant

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 AI 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 Débogage d'un entraînement de modèle en échec.

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 AWS SDK pour créer des ensembles de données d'entraînement et de test.

L'`CreateDataset`opération vous permet de spécifier éventuellement des balises lors de la création d'un nouvel ensemble de données, dans le but de catégoriser et de gérer vos ressources.

Jeu de données d'entraînement

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

- CreateDatasetÀ utiliser avec un fichier manifeste au format Amazon Sagemaker que vous fournissez. Pour de plus amples informations, veuillez consulter the section called "Création d'un fichier manifeste". Pour obtenir un exemple de code, consultez Création d'un ensemble de données à l'aide d'un fichier manifeste (SDK) SageMaker AI 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 partir d'un jeu de données existant (kit 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 Ajout

d’un jeu de données à un projet. Pour ajouter des images à un jeu de données, consultez Ajout d’autres images (kit 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 AWS SDK pour créer un ensemble de données de test de la manière suivante :

- CreateDatasetÀ utiliser avec un fichier manifeste au format Amazon Sagemaker que vous fournissez. Pour de plus amples informations, veuillez consulter the section called “Création d’un fichier manifeste”. Pour obtenir un exemple de code, consultez Création d'un ensemble de données à l'aide d'un fichier manifeste (SDK) SageMaker AI 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 partir d’un jeu de données existant (kit 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 Ajout d’un jeu de données à un projet. Pour ajouter des images à un jeu de données, consultez Ajout d’autres images (kit 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 Ajout d’un jeu de données à un projet (kit SDK). Pour fractionner le jeu de données d’entraînement, consultez Distribution d’un jeu de données d’entraînement (kit SDK).

Importation d'images depuis un compartiment Amazon S3

Les images sont stockées dans un compartiment Amazon S3. Vous pouvez utiliser le compartiment de console ou un autre compartiment Amazon S3 de votre AWS compte. 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 AWS SDK pour créer un ensemble 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 plus d'informations, consultez Utilisation d'un fichier manifeste pour importer des images.

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-dotsest 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 white-echo-dotsont étiquetées white-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 console) créé pour vous par Amazon Rekognition lorsque vous avez ouvert la console pour la première fois dans la région actuelle. AWS 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 console Amazon S3 AWS Management Console et ouvrez-la 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 de plus amples informations, veuillez consulter Gestion d'un projet Étiquettes personnalisées Amazon Rekognition.
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.

Importation d'images depuis un 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 de plus amples informations, veuillez consulter Importation d'images depuis un compartiment Amazon S3.

> **Note**
>
> Vous ne pouvez pas utiliser le 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 Utilisation d'un fichier manifeste pour importer des images.

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.

Utilisation d'un fichier manifeste pour importer des images

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

L'`CreateDataset`opération est mise à jour pour vous permettre de spécifier éventuellement des balises lors de la création d'un nouvel ensemble de données. Les balises sont des paires clé-valeur que vous pouvez utiliser pour classer et gérer vos ressources.

Rubriques

- Création d'un ensemble de données à l'aide d'un fichier manifeste SageMaker AI Ground Truth (console)
- Création d'un ensemble de données à l'aide d'un fichier manifeste (SDK) SageMaker AI Ground Truth
- Créer une demande de jeu de données
- Étiqueter des images avec une tâche Amazon SageMaker AI Ground Truth
- Création d'un fichier manifeste
- Importation d'étiquettes au niveau de l'image dans des fichiers manifestes

- Localisation d’objets dans les fichiers manifestes
- Règles de validation des fichiers manifestes
- Conversion d’autres formats de jeu de données en fichier manifeste

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

La procédure suivante explique comment créer un ensemble de données à l'aide d'un fichier manifeste au format SageMaker AI 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 AI GroundTruth Job en suivant les instructions de Étiqueter des images avec une tâche Amazon SageMaker AI 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 ensemble de données à l'aide d'un fichier manifeste (SDK) SageMaker AI 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 SageMaker AI Ground Truth, ou créer votre propre fichier manifeste.

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.

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 AI GroundTruth Job en suivant les instructions de Étiqueter des images avec une tâche Amazon SageMaker AI 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
  --tags '{"key1": "value1", "key2": "value2"}'
```

   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 (kit SDK) pour entraîner le modèle.

Créer une demande de jeu de données

Le format de la demande d' CreateDataset opération est le suivant :

```
{
"DatasetSource": {
"DatasetArn": "string",
"GroundTruthManifest": {
"S3Object": {
"Bucket": "string",
"Name": "string",
"Version": "string"
}
}
},
"DatasetType": "string",
"ProjectArn": "string",
"Tags": {
"string": "string"
}
}
```

Étiqueter des images avec une tâche Amazon SageMaker AI Ground Truth

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

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

- 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 AI Ground Truth (Console)

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

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

1. Connectez-vous à la console Amazon S3 AWS Management Console et ouvrez-la 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 bucket de console est créé lorsque vous ouvrez pour la première fois la console Amazon Rekognition Custom Labels dans une région. AWS Pour de plus amples informations, veuillez consulter Gestion d'un projet Étiquettes personnalisées Amazon Rekognition.

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 console SageMaker AI à l'adresse https://console.aws.amazon.com/sagemaker/.
6. Créez une tâche d'étiquetage Ground Truth. Vous aurez besoin d'Amazon S3 URLs pour les 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 SageMaker AI 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 AWS SDK, faites-leCréation d'un ensemble de données à l'aide d'un fichier manifeste (SDK) SageMaker AI Ground Truth.
9. Répétez les étapes 1 à 6 pour créer une tâche SageMaker AI 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 AI Ground Truth. Si vos images sont étiquetées dans un format autre qu'un fichier manifeste SageMaker AI Ground Truth, utilisez les informations suivantes pour créer un fichier manifeste au format SageMaker AI 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 AI 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 d'emplacement d'objets ne sont pas nécessaires) figurant sur une image. Une image peut avoir plusieurs étiquettes au niveau de l'image. Pour plus d'informations, consultez Importation d'étiquettes au niveau de l'image dans des 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

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 AI 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 AI Ground Truth pour votre ensemble de données d'entraînement. Pour plus d'informations, consultez Importation d'étiquettes au niveau de l'image dans des 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 AI 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 SageMaker AI 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 AWS SDK, faites-leCréation d'un ensemble de données à l'aide d'un fichier manifeste (SDK) SageMaker AI Ground Truth.

Importation d'étiquettes au niveau de l'image dans des 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 SageMaker AI 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-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"}}
```

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 AI 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-métadonnées

(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 SageMaker AI 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 AI 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*-métadonnées

(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 AI Ground Truth impose la validation syntaxique. 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 au format Amazon SageMaker AI 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 de plus amples informations, veuillez consulter Utilisation d'un fichier manifeste pour importer des images.

Rubriques

- Transformation d'un ensemble de données COCO en un format de fichier manifeste
- Transformation des fichiers manifestes SageMaker AI Ground Truth à étiquettes multiples
- Création d'un fichier manifeste à partir d'un fichier CSV

Transformation d'un ensemble de données COCO en un format de fichier manifeste

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 Le format du jeu 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.

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

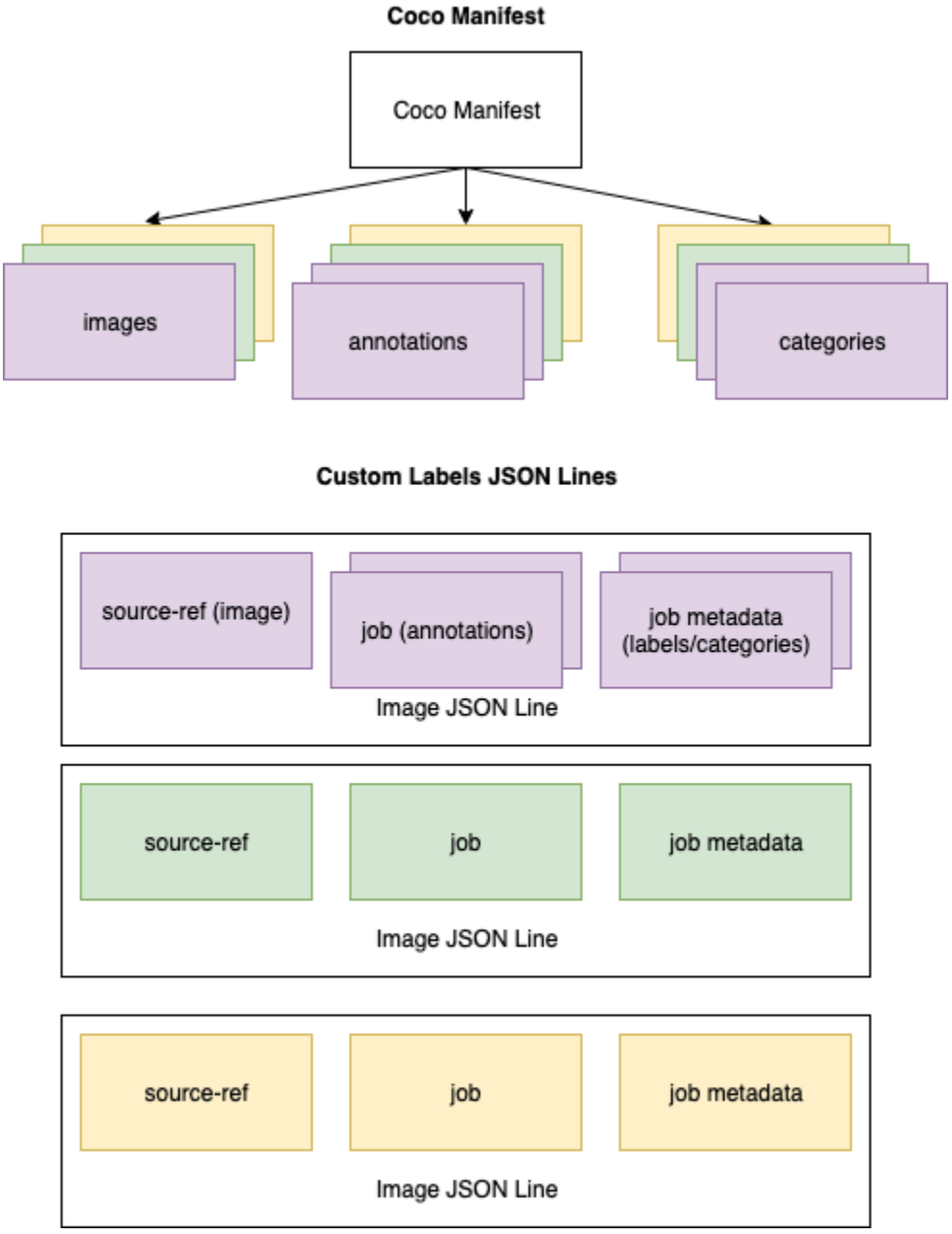
Pour transformer un jeu de données au format COCO, mappez le jeu de données COCO avec un fichier manifeste Étiquettes personnalisées Amazon Rekognition pour la localisation d'objets. Pour de plus amples informations, veuillez consulter Localisation d'objets dans les fichiers manifestes. Pour créer une ligne JSON pour chaque image, le fichier manifeste doit mapper le jeu de données `image` COCO et le champ `category` d'objet IDs. `annotation`

Voici un exemple de fichier manifeste COCO. Pour plus d'informations, consultez Le format du jeu de données 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/xxxxxxxxxxxx.jpg", "flickr_url": "http://farm3.staticflickr.com/2626/
xxxxxxxxxxxx.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/nnnnnnnnnnnn.jpg", "flickr_url": "http://farm1.staticflickr.com/88/
xxxxxxxxxxxx.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"}
    ]
}
```

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. Chaque ligne JSON d'une image possède une référence source, un champ de métadonnées de tâche et un champ de métadonnées de tâche. Les couleurs qui sont les mêmes indiquent des informations relatives à une seule image. Notez que dans le manifeste, une image individuelle peut comporter plusieurs annotations et métadonnées/catégories.



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`-métadonnées

Métadonnées pour l'attribut d'étiquette. Inclut les étiquettes et les identifiants d'étiquettes. Pour plus d'informations, consultez *`bounding-box`*-métadonnées.

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`*-métadonnées.

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.

Le format du jeu de données 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 `categories` 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/xxxxxxxxxxxx.jpg", "flickr_url": "http://farm3.staticflickr.com/2626/
xxxxxxxxxxxx.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/nnnnnnnnnnnn.jpg", "flickr_url": "http://farm1.staticflickr.com/88/
xxxxxxxxxxxx.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/nnnnnnnnnnnn.jpg",
    "flickr_url": "http://farm1.staticflickr.com/88/nnnnnnnnnnnnnnnnnn.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 de plus amples informations, veuillez consulter Configuration des autorisations du kit SDK.

   b. Installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.

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 botocore
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 seperately
        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 SageMaker AI 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 ensemble de données à l'aide d'un fichier manifeste (SDK) SageMaker AI Ground Truth.

Transformation des fichiers manifestes SageMaker AI Ground Truth à étiquettes multiples

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

SageMaker Les fichiers manifestes AI 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 AI 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 d' SageMaker AI Ground Truth

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

Pour télécharger un fichier manifeste de travail d' SageMaker AI Ground Truth

1. Ouvrez la 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 manifeste d' SageMaker IA à étiquettes multiples

La procédure suivante crée un fichier manifeste Amazon Rekognition Custom Labels au format multi-étiquettes à partir d'un fichier manifeste AI 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 manifeste SageMaker AI 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 d' SageMaker AI 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 and 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 SageMaker AI 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 ensemble de données à l'aide d'un fichier manifeste (SDK) SageMaker AI 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 des images à plusieurs étiquettes. Pour plus d'informations, consultez Recherche d'objets, de scènes et de concepts.

> **Note**
>
> Ce script ne crée pas de fichier manifeste adapté à la recherche des emplacements d'objets ni à la recherche 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 "Importation d'étiquettes au niveau de l'image dans des 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 Classification d'images, 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 des images à plusieurs étiquettes (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 dédupliqué. Le fichier image CSV dédupliqué 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 dédupliqué si nécessaire. Réexécutez le script avec le fichier dédupliqué. Si aucun doublon n'est détecté dans le fichier CSV d'entrée, le script supprime le fichier image CSV dédupliqué et l'image dupliquée CSVfile, 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, `s3://my-bucket/flowers/train/camellia1.jpg` . 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`

2. Enregistrez le fichier CSV.
3. 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
    occurence 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 occurence 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 SageMaker AI 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 ensemble de données à l'aide d'un fichier manifeste (SDK) SageMaker AI Ground Truth.

Copier le contenu d'un ensemble de données existant

Si vous avez déjà créé un jeu de données, vous pouvez copier son contenu dans un nouveau jeu de données. Pour créer un ensemble de données à partir d'un ensemble de données existant à l'aide du AWS SDK, consultezCréation d'un jeu de données à partir d'un jeu de données existant (kit 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.

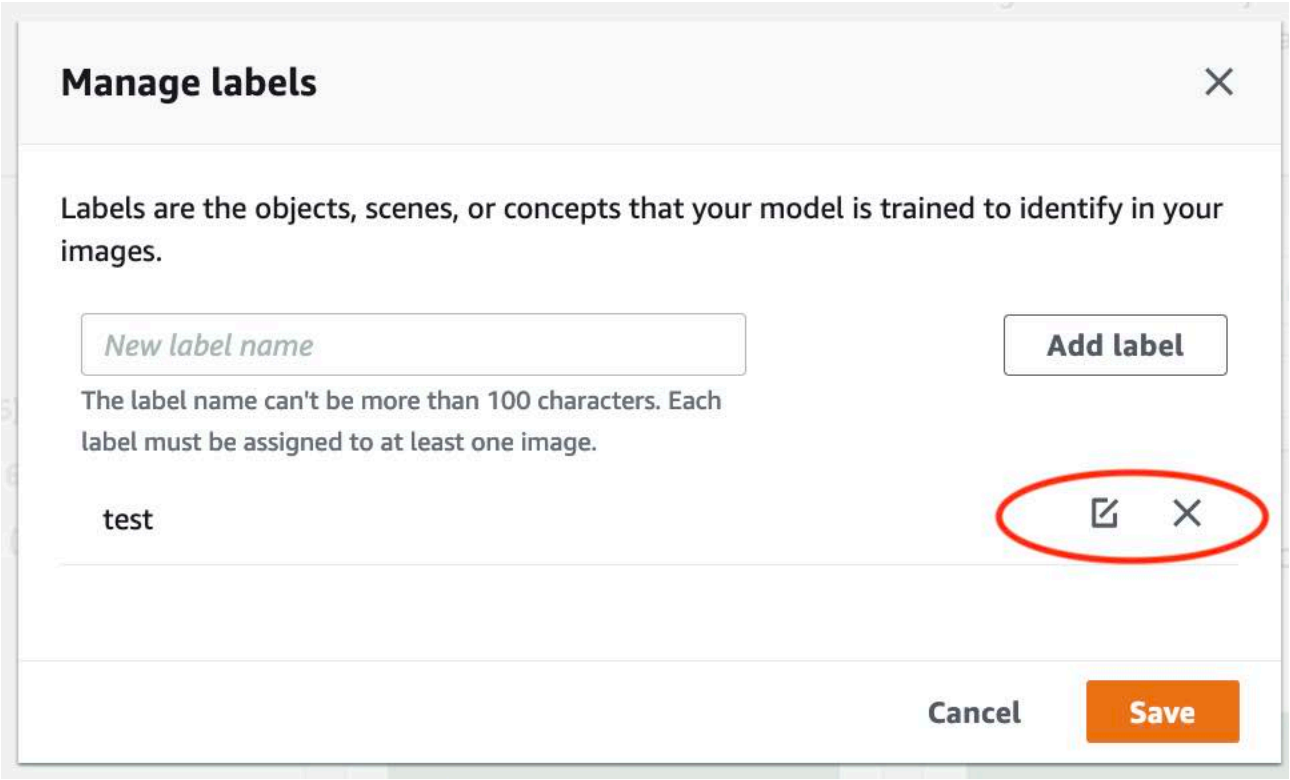
Modification et suppression d'étiquettes (console)

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 Ajout d'autres images (kit 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 Liste des entrées d'un jeu de données (kit SDK).
2. Dans le fichier, supprimez toutes les annotations d'étiquette. Pour plus d'informations, consultez Importation d'étiquettes au niveau de l'image dans des 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 Ajout d'autres images (kit 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 de plus amples informations, veuillez consulter 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 "Importation d'étiquettes au niveau de l'image dans des fichiers manifestes".

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

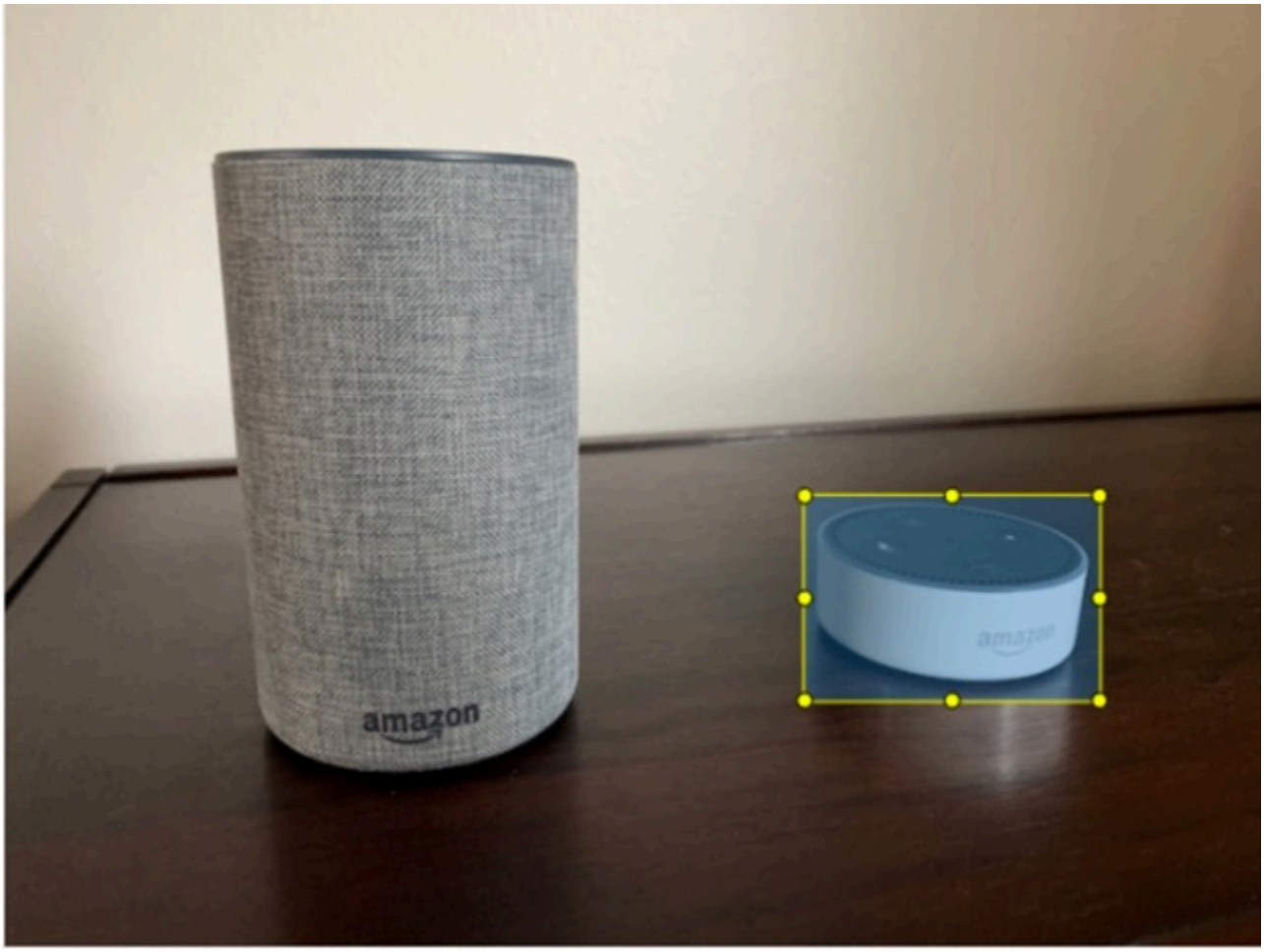
1. Pour obtenir la ligne get JSON correspondant à l'image existante, utilisez `ListDatasetEntries`. Pour le champ `source-ref`, spécifiez l'emplacement de l'image à laquelle vous souhaitez attribuer l'étiquette. Pour plus d'informations, consultez Liste des entrées d'un jeu de données (kit SDK).
2. Mettez à jour la ligne JSON renvoyée à l'étape précédente à l'aide des informations indiquées dans Importation d'étiquettes au niveau de l'image dans des fichiers manifestes.
3. Appelez `UpdateDatasetEntries` pour mettre à jour l'image. Pour plus d'informations, consultez Ajout 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.



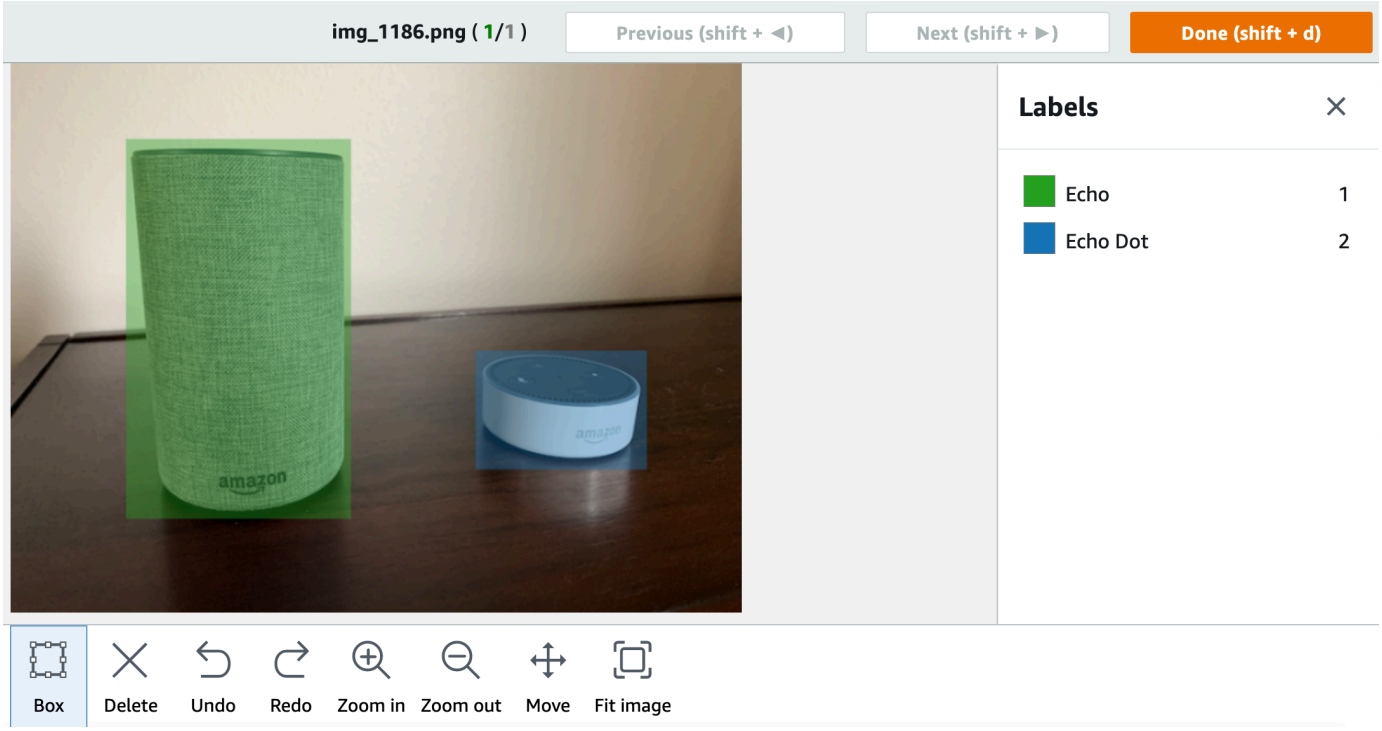
Localisation d'objets à l'aide de cadres de délimitation (console)

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 d'emplacement d'objets 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 Liste des entrées d'un jeu de données (kit 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 Ajout 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 de débogage des jeux de données du terminal
- Débogage des erreurs d'ensembles de données non terminaux

Erreurs de débogage des jeux de données du terminal

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 Description d'un jeu de données (kit 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

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 Directives et quotas dans Étiquettes personnalisées 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

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 d'emplacement d'objets. Pour plus d'informations, consultez Directives et quotas dans Étiquettes personnalisées 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

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

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 définitives de contenu de manifeste.

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

ERROR_MANIFEST_TOO_MANY_LABELS

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'objets : 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

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.

Débogage des erreurs d'ensembles de données non terminaux

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 Liste des entrées d'un jeu de données (kit 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 non définitives de validation de ligne JSON.

- ERROR_NO_LABEL_ATTRIBUTES
- ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT

- ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT
- ERROR_NO_VALID_LABEL_ATTRIBUTES
- ERROR_INVALID_BOUNDING_BOX
- ERROR_INVALID_IMAGE_DIMENSION
- ERROR_BOUNDING_BOX_TOO_SMALL
- ERROR_NO_VALID_ANNOTATIONS
- ERROR_MISSING_BOUNDING_BOX_CONFIDENCE
- ERROR_MISSING_CLASS_MAP_ID
- ERROR_TOO_MANY_BOUNDING_BOXES
- ERROR_UNSUPPORTED_USE_CASE_TYPE
- ERROR_INVALID_LABEL_NAME_LENGTH

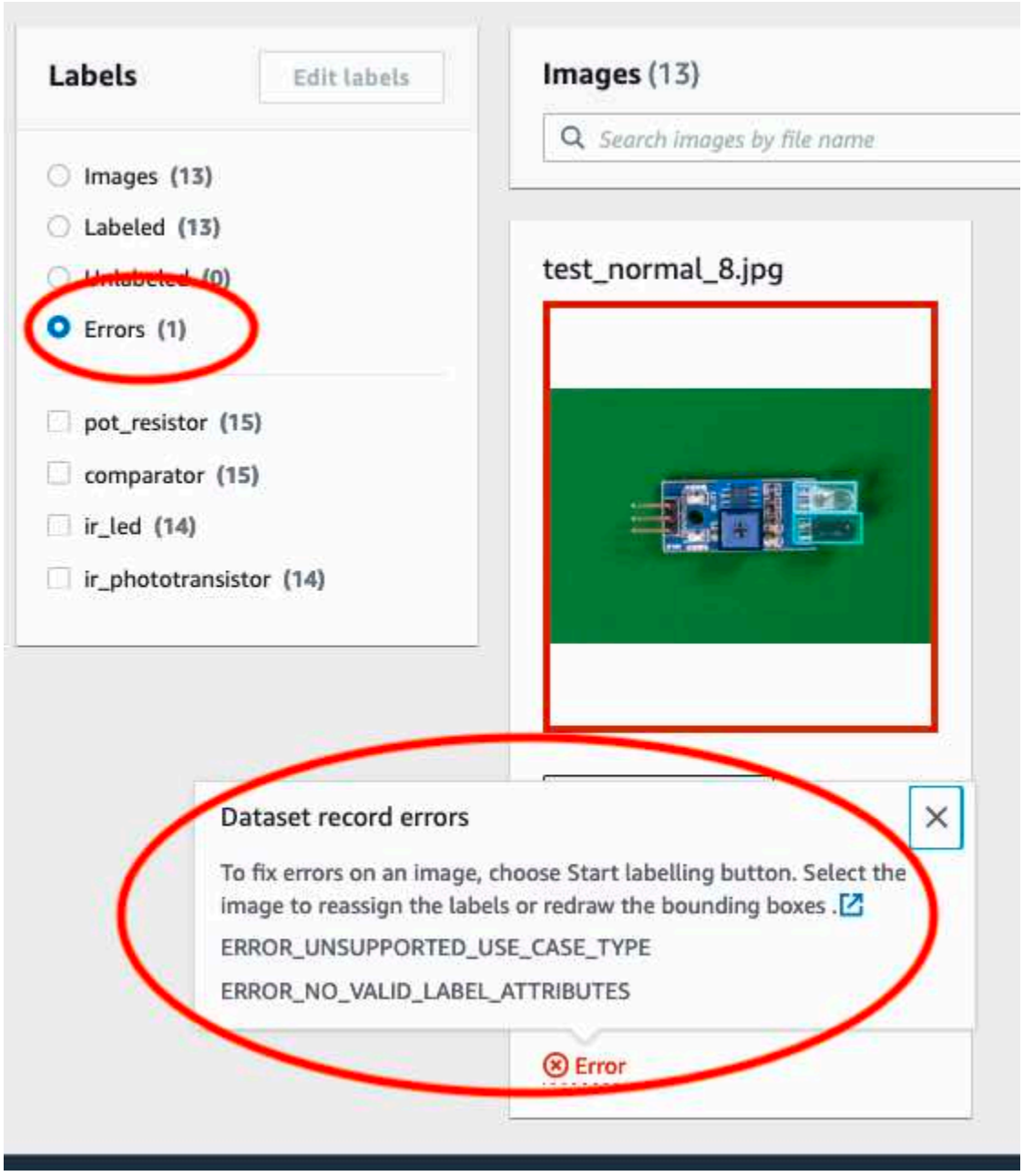
Accès aux erreurs non définitives

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 Liste des entrées d'un jeu de données (kit 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 non définitives de validation de ligne JSON pour corriger l'erreur.



Entraînement d'un modèle Étiquettes personnalisées Amazon Rekognition

Vous pouvez entraîner un modèle à l'aide de la console Étiquettes personnalisées Amazon Rekognition ou de l'API Étiquettes personnalisées Amazon Rekognition. Si l'entraînement du modèle

échoue, utilisez les informations de la section Débogage d'un entraînement de modèle en échec pour trouver la cause de cet échec.

> **Note**
>
> Le temps nécessaire à l'entraînement d'un modèle vous est facturé. L'entraînement dure généralement de 30 minutes à 24 heures. Pour plus d'informations, consultez Heures d'entraînement.

Une nouvelle version du modèle est créée chaque fois que celui-ci est entraîné. Étiquettes personnalisées Amazon Rekognition crée un nom pour le modèle. Ce nom combine le nom du projet et la date de création du modèle.

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

Pour plus d'informations, consultez Concepts d'AWS Key Management Service. Vos images source ne sont pas affectées.

Vous pouvez utiliser le chiffrement côté serveur KMS (SSE-KMS) pour chiffrer les images d'entraînement et de test de votre compartiment Amazon S3, avant qu'elles ne soient copiées par Étiquettes personnalisées Amazon Rekognition. Pour autoriser les étiquettes personnalisées Amazon Rekognition à accéder à vos images AWS , votre compte doit disposer 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).

Une fois que vous avez entraîné un modèle, vous pouvez évaluer ses performances et l'améliorer. Pour plus d'informations, consultez Amélioration d'un modèle entraîné Étiquettes personnalisées Amazon Rekognition.

Pour connaître les autres tâches possibles avec un modèle, telles que le balisage, consultez Gestion d'un modèle Étiquettes personnalisées Amazon Rekognition.

Rubriques

- Entraînement d'un modèle (console)
- Entraînement d'un modèle (kit SDK)

Entraînement d'un modèle (console)

Vous pouvez utiliser la console Étiquettes personnalisées Amazon Rekognition pour entraîner un modèle.

L'entraînement nécessite un projet comprenant un jeu de données d'entraînement et un jeu de données de test. Si votre projet ne dispose pas d'un jeu de données de test, la console Étiquettes personnalisées Amazon Rekognition divise le jeu de données d'entraînement pendant l'entraînement afin d'en créer un pour votre projet. Les images choisies constituent un échantillon représentatif et ne sont pas utilisées dans le jeu de données d'entraînement. Nous vous recommandons de diviser votre jeu de données d'entraînement uniquement si vous ne disposez pas d'un jeu de données de test que vous pouvez utiliser. La division d'un jeu de données d'entraînement réduit le nombre d'images disponibles pour l'entraînement.

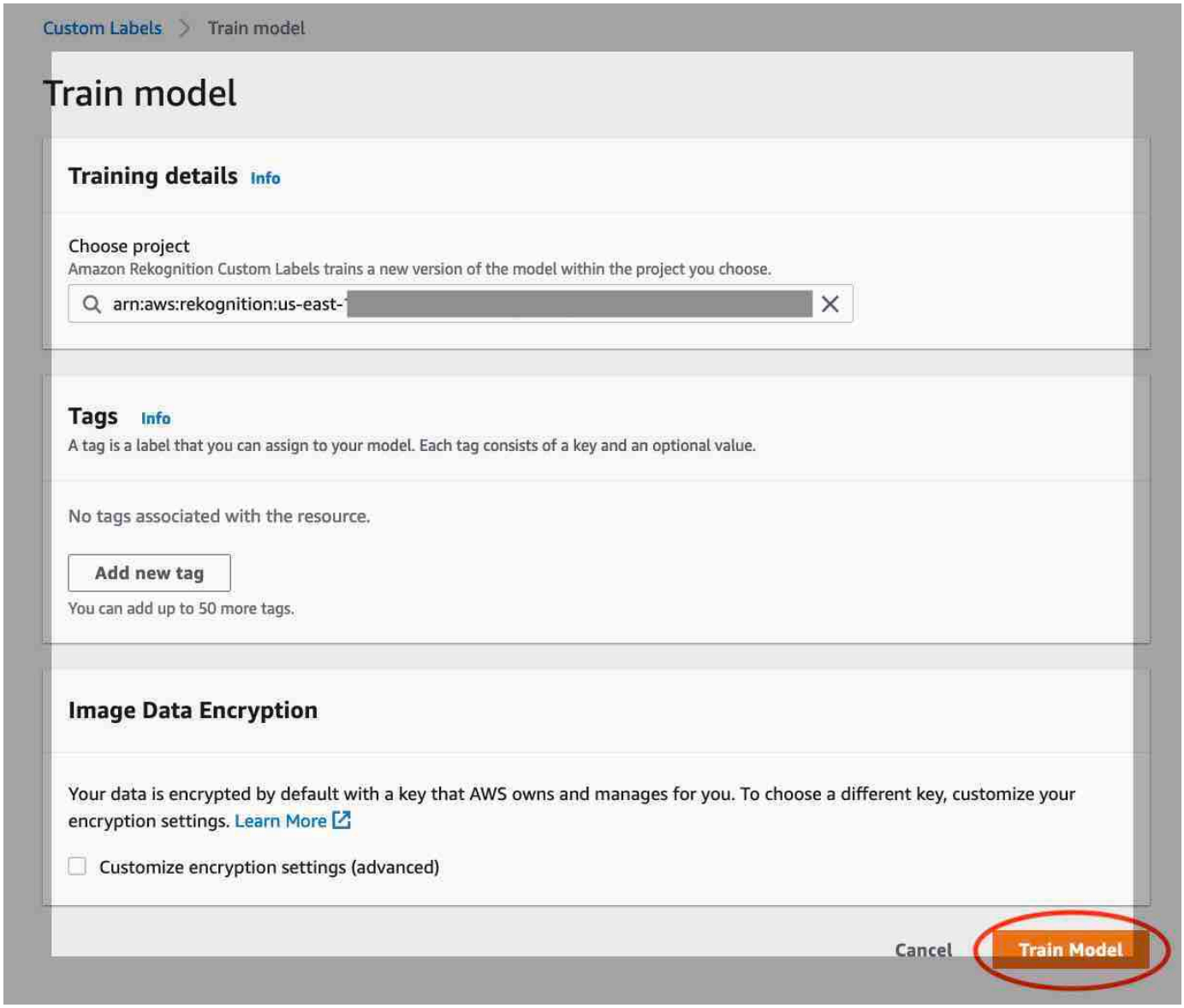
> **Note**
>
> Le temps nécessaire à l'entraînement d'un modèle vous est facturé. Pour plus d'informations, consultez Heures d'entraînement.

Pour entraîner votre modèle (console)

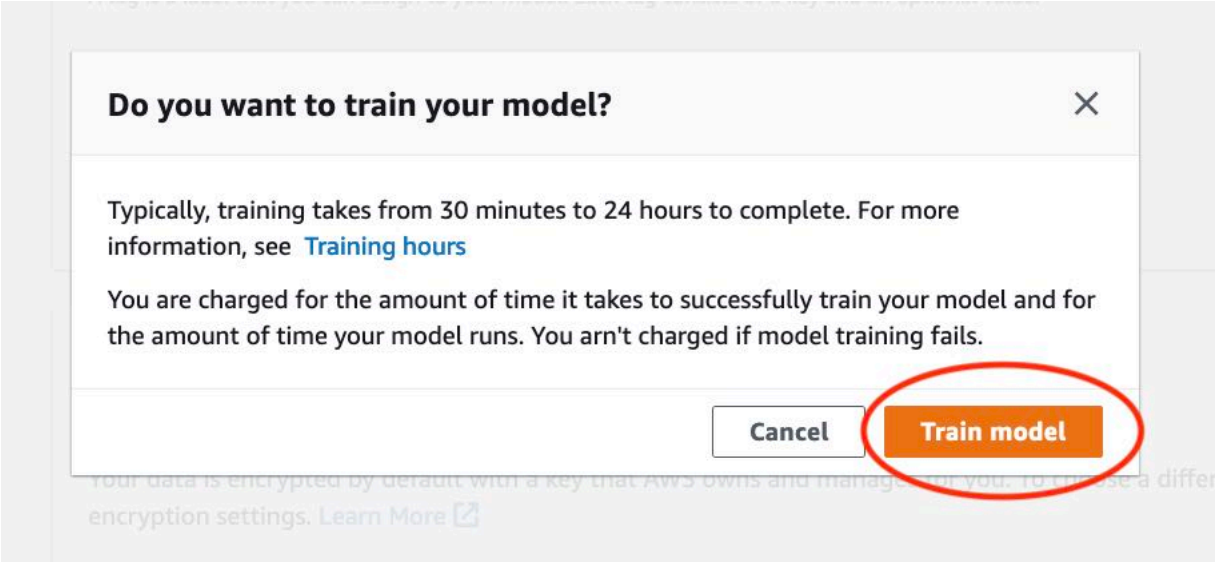
1. Ouvrez la console Amazon Rekognition à l'adresse. https://console.aws.amazon.com/rekognition/
2. Choisissez Utiliser Custom Labels.
3. Dans le volet de navigation de gauche, choisissez Projets.

4. Sur la page Projets, choisissez le projet qui contient le modèle que vous souhaitez entraîner.
5. Sur la page Projet, sélectionnez Entraîner un modèle.

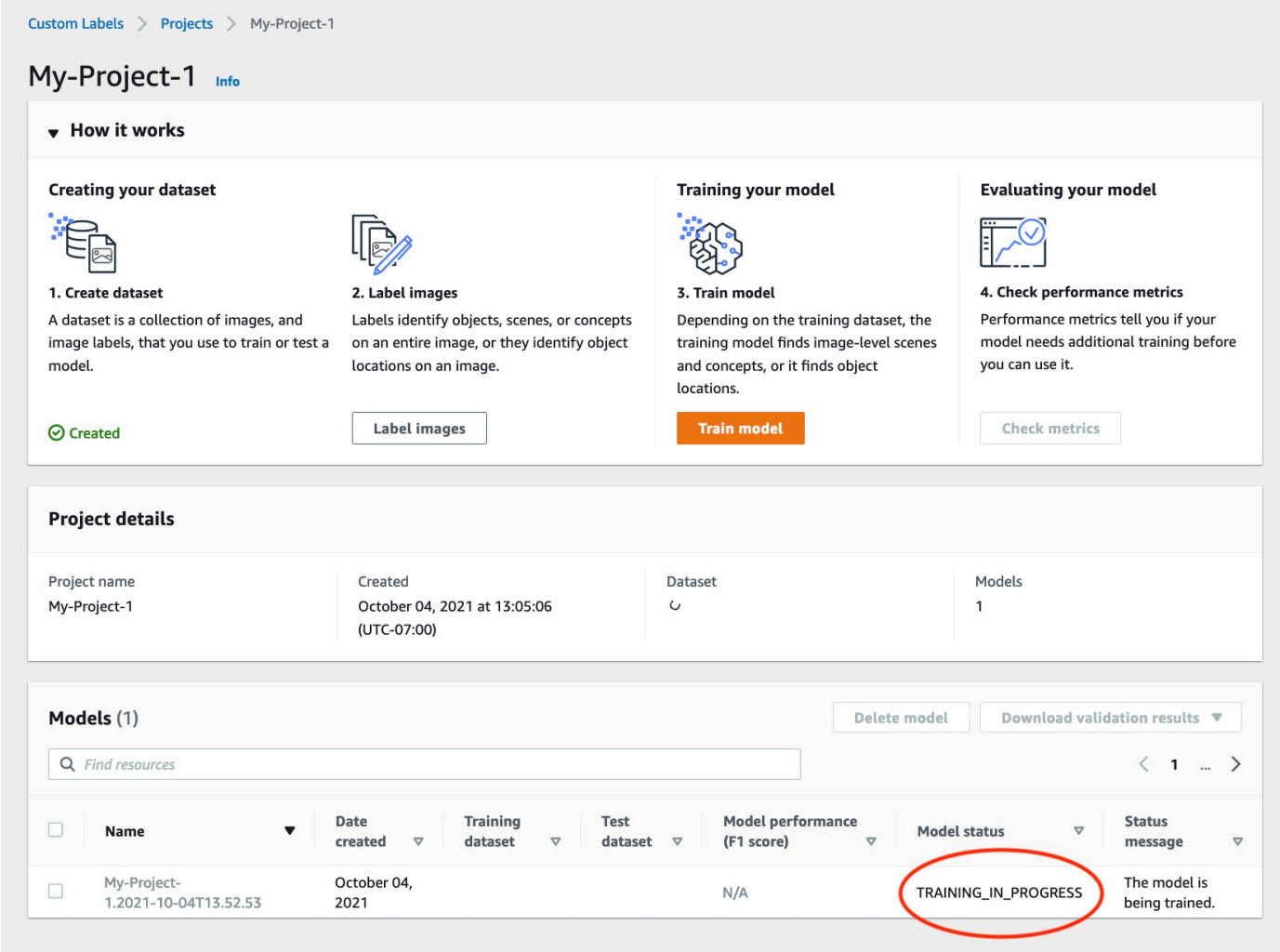
6. (Facultatif) Si vous souhaitez utiliser votre propre clé de chiffrement AWS KMS, procédez comme suit :
   a. Dans Chiffrement des données d'image, choisissez Personnaliser les paramètres de chiffrement (avancé).
   b. Dans encryption.aws_kms_key, saisissez l'Amazon Resource Name (ARN) de votre clé, ou choisissez une clé AWS KMS existante. Pour créer une clé, choisissez Créer une clé AWS KMS.
7. (Facultatif) Si vous souhaitez ajouter des balises à votre modèle, procédez comme suit :
   a. Dans la section Balises, choisissez Ajouter une balise.
   b. Saisissez :
      i. Le nom de la clé dans le champ Clé.
      ii. La valeur de la clé dans le champ Valeur.
   c. Pour ajouter d'autres balises, répétez les étapes 6a et 6b.
   d. (Facultatif) Si vous souhaitez supprimer une balise, choisissez Supprimer en regard de la balise que vous souhaitez supprimer. Si vous supprimez une balise précédemment enregistrée, elle sera supprimée lorsque vous enregistrerez vos modifications.
8. Sur la page Entraîner un modèle, choisissez Entraîner un modèle. L'Amazon Resource Name (ARN) de votre projet doit se trouver dans la zone d'édition Choisir un projet. Dans le cas contraire, saisissez l'ARN de votre projet.



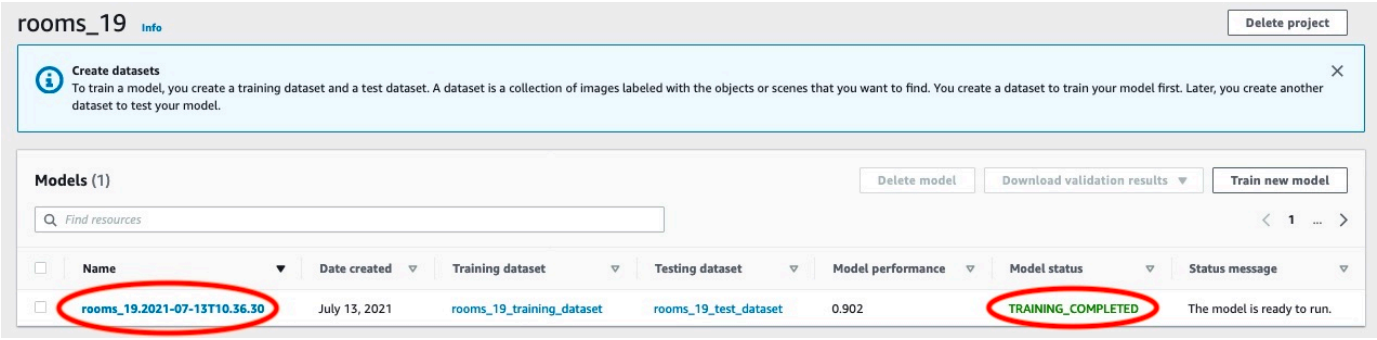
9. Dans la boîte de dialogue Voulez-vous entraîner votre modèle ?, choisissez Entraîner un modèle.



10. Dans la section Modèles de la page du projet, vous pouvez vérifier si l'entraînement est en cours dans la colonne `Model Status`. L'entraînement d'un modèle peut nécessiter un certain temps.



11. Une fois l'entraînement terminé, choisissez le nom du modèle. L'entraînement est terminé lorsque le statut du modèle est TRAINING_COMPLETED. En cas d'échec de l'entraînement, consultez Débogage d'un entraînement de modèle en échec.



12. Étape suivante : Évaluer votre modèle. Pour plus d'informations, consultez Amélioration d'un modèle entraîné Étiquettes personnalisées Amazon Rekognition.

Entraînement d'un modèle (kit SDK)

Vous entraînez un mannequin en appelant CreateProjectVersion. Pour entraîner un modèle, les informations suivantes sont nécessaires :

- Nom : nom unique de la version du modèle.
- ARN du projet : Amazon Resource Name (ARN) du projet qui gère le modèle.
- Emplacement des résultats de l'entraînement : emplacement du compartiment Amazon S3 dans lequel les résultats de l'entraînement sont enregistrés. Vous pouvez utiliser l'emplacement du compartiment Amazon S3 de la console ou choisir un autre emplacement. Nous vous recommandons de choisir un autre emplacement, car cela vous permet de définir des autorisations et d'éviter d'éventuels conflits d'appellation avec les résultats d'entraînement obtenus à l'aide de la console Étiquettes personnalisées Amazon Rekognition.

L'entraînement utilise les jeux de données d'entraînement et de test associés au projet. Pour plus d'informations, consultez Gestion des jeux de données.

> **Note**
>
> Vous pouvez éventuellement spécifier des fichiers manifestes de jeux de données d'entraînement et de test qui sont externes à un projet. Si vous ouvrez la console après avoir entraîné un modèle avec des fichiers manifestes externes, Étiquettes personnalisées Amazon Rekognition crée les jeux de données pour vous en utilisant le dernier jeu de

fichiers manifestes utilisé pour l'entraînement. Vous ne pouvez plus entraîner une version de modèle pour le projet en spécifiant des fichiers manifestes externes. Pour de plus amples informations, veuillez consulter CreatePrjectVersion.

La réponse de `CreateProjectVersion` est un ARN que vous utilisez pour identifier la version du modèle dans les demandes ultérieures. Vous pouvez également utiliser l'ARN pour sécuriser la version du modèle. Pour plus d'informations, consultez Sécurisation des projets Étiquettes personnalisées Amazon Rekognition.

L'entraînement d'une version de modèle peut nécessiter un certain temps. Les exemples Python et Java présentés dans cette section utilisent des programmes d'attente pour attendre la fin de l'entraînement. Un programme d'attente est un utilitaire qui attend qu'un statut particulier survienne. Vous pouvez également obtenir le statut actuel de l'entraînement en appelant `DescribeProjectVersions`. L'entraînement est terminé lorsque la valeur du champ `Status` est `TRAINING_COMPLETED`. Une fois l'entraînement terminé, vous pouvez évaluer la qualité du modèle en passant en revue les résultats de l'évaluation.

Entraînement d'un modèle (kit SDK)

L'exemple suivant explique comment entraîner un modèle en utilisant les jeux de données d'entraînement et de test associés à un projet.

Pour entraîner un modèle (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez l'exemple de code suivant pour entraîner un projet.

   AWS CLI

   L'exemple suivant crée un modèle. Le jeu de données d'entraînement est divisé pour créer le jeu de données de test. Remplacez les éléments suivants :

   - `my_project_arn` par l'Amazon Resource Name (ARN) du projet.
   - `version_name` par un nom de version unique de votre choix.
   - `output_bucket` par le nom du compartiment Amazon S3 dans lequel Étiquettes personnalisées Amazon Rekognition enregistre les résultats de l'entraînement.

- `output_folder` par le nom du dossier dans lequel les résultats de l'entraînement sont enregistrés.
- (Paramètre facultatif) `--kms-key-id` par 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 (ARN) du projet.
- `version_name` : nom de version unique pour le modèle de votre choix.
- `output_bucket` : nom du compartiment Amazon S3 dans lequel Étiquettes personnalisées Amazon Rekognition enregistre les résultats de l'entraînement.
- `output_folder` : nom du dossier dans lequel les résultats de l'entraînement sont enregistrés.

Vous pouvez éventuellement fournir les paramètres de ligne de commande suivants pour associer une balise à votre modèle :

- `tag` : nom de la balise que vous souhaitez associer au modèle.
- `tag_value` : 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 (ARN) du projet.
- `version_name` : nom de version unique pour le modèle de votre choix.
- `output_bucket` : nom du compartiment Amazon S3 dans lequel Étiquettes personnalisées Amazon Rekognition enregistre les résultats de l'entraînement.

- `output_folder` : nom du dossier dans lequel les résultats de l'entraînement 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. En cas d'échec de l'entraînement, consultez Débogage d'un entraînement de modèle en échec.

Débogage d'un entraînement de modèle en échec

Il est possible que vous rencontriez des erreurs lors de l'entraînement d'un modèle. Amazon Rekognition Custom Labels signale des erreurs d'entraînement dans la console et dans la réponse de. DescribeProjectVersions

Les erreurs sont soit définitives (l'entraînement ne peut pas continuer), soit non définitives (l'entraînement peut continuer). Concernant les erreurs liées au contenu des jeux de données d'entraînement et de test, vous pouvez télécharger les résultats de validation (un récapitulatif du manifeste et des manifestes de validation des entraînements 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 dans les fichiers manifestes (erreurs définitives survenant avant que le contenu du fichier manifeste soit validé).

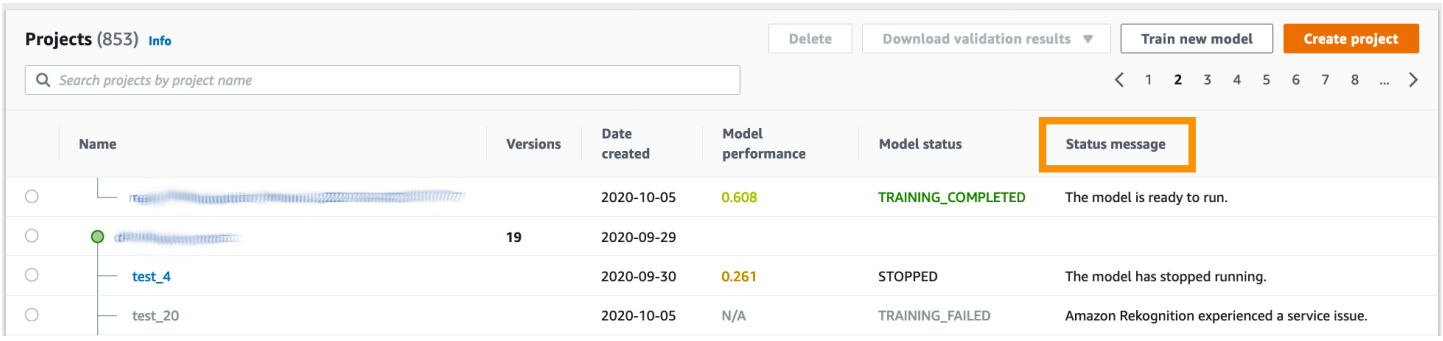
> **Note**
>
> Un manifeste est un fichier utilisé pour stocker le contenu d'un jeu de données.

Vous pouvez corriger certaines erreurs à l'aide de la console Étiquettes personnalisées Amazon Rekognition. D'autres erreurs peuvent vous obliger à mettre à jour les fichiers manifestes d'entraînement ou de test. Vous aurez peut-être besoin d'apporter d'autres modifications, par exemple aux autorisations IAM. Pour plus d'informations, consultez la documentation relative aux erreurs individuelles.

Erreurs définitives

Les erreurs définitives interrompent l'entraînement d'un modèle. Il existe 3 catégories d'erreurs définitives d'entraînement : les erreurs de service, les erreurs de fichier manifeste et les erreurs de contenu de manifeste.

Dans la console, la fonctionnalité Étiquettes personnalisées Amazon Rekognition affiche les erreurs définitives d'un modèle dans la colonne Message d'état de la page des projets. Le tableau de bord de gestion de projet affiche la liste des projets avec le nom, les versions, la date de création, les performances du modèle et un message d'état indiquant l'état du modèle, tel que la formation terminée ou échouée



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 valeur `Status` est `TRAINING_FAILED` et le champ `StatusMessage` contient l'erreur.

Erreurs de service

Des erreurs définitives de service se produisent quand Amazon Rekognition rencontre un problème de service et ne peut pas poursuivre l'entraînement. Par exemple, la défaillance d'un autre service dont dépend la fonctionnalité Étiquettes personnalisées Amazon Rekognition. La fonctionnalité Étiquettes personnalisées Amazon Rekognition signale les erreurs de service dans la console quand Amazon Rekognition rencontre un problème de service. Si vous utilisez le AWS SDK, les erreurs de service survenant pendant l'entraînement sont signalées comme une `InternalServerError` exception par CreateProjectVersionet DescribeProjectVersions.

En cas d'erreur de service, réessayez d'entraîner le modèle. Si l'entraînement continue d'échouer, contactez AWS Support et incluez toute information signalée avec l'erreur de service.

Liste des erreurs du fichier manifeste du terminal

Les erreurs de fichier manifeste sont des erreurs définitives présentes dans les jeux de données d'entraînement et de test, qui se produisent au niveau du fichier ou sur plusieurs fichiers. Elles sont détectées avant que le contenu des jeux de données d'entraînement et de test soit validé. Les erreurs de fichier manifeste empêchent le signalement des erreurs de validation non définitives. Par exemple, un fichier manifeste vide génère une erreur The manifest file is empty. Le fichier étant vide, aucune erreur de validation de ligne JSON non définitive ne peut être signalée. Le récapitulatif 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 de fichier manifeste.

- Le contenu ou l'extension 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.

Liste des erreurs de contenu du manifeste du terminal

Les erreurs de contenu de manifeste sont des erreurs définitives 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 permettre le fractionnement automatique. s'affiche, 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 `DescribeProjectVersions`, l'erreur est signalée dans le récapitulatif du manifeste avec toute autre erreur définitive de contenu. Pour plus d'informations, consultez Présentation du récapitulatif du manifeste.

Les erreurs non définitives de ligne JSON sont également signalées dans des manifestes distincts de résultats de validation des entraînements et des tests. Les erreurs non définitives de ligne JSON détectées par la fonctionnalité Étiquettes personnalisées Amazon Rekognition ne sont pas nécessairement liées aux erreurs de contenu de manifeste qui interrompent l'entraînement. Pour plus d'informations, consultez Présentation des manifestes de résultats de validation des entraînements et des tests.

Vous devez corriger les erreurs de fichier manifeste avant de pouvoir entraîner votre modèle.

Les messages d'erreur relatifs aux erreurs de contenu de 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 S3 d'images.
- Le fichier manifeste ne contient pas suffisamment d'images étiquetées par étiquette pour permettre le fractionnement automatique.
- Le fichier manifeste contient trop peu d'étiquettes.
- Le fichier manifeste contient trop d'étiquettes.
- Moins de {} % de chevauchement d'étiquettes entre les fichiers manifestes d'entraînement et de test.
- Le fichier manifeste contient trop peu d'étiquettes utilisables.
- Moins de {} % de chevauchement d'étiquettes utilisables entre les fichiers manifestes d'entraînement et de test.
- Échec de la copie d'images à partir du compartiment S3.

Liste des erreurs de validation de ligne JSON non terminales

Les erreurs de validation de ligne JSON sont des erreurs non définitives qui n'obligent pas la fonctionnalité Étiquettes personnalisées Amazon Rekognition à arrêter l'entraînement d'un modèle.

Les erreurs de validation de ligne JSON ne s'affichent pas dans la console.

Dans les jeux de données d'entraînement et de test, une ligne JSON représente les informations d'entraînement ou de test d'une seule image. Les erreurs de validation dans une ligne JSON, comme une image non valide, sont signalées dans les manifestes de validation des entraînements et des tests. La fonctionnalité Étiquettes personnalisées Amazon Rekognition termine l'entraînement en utilisant les autres lignes JSON valides du manifeste. Pour plus d'informations, consultez Présentation des manifestes de résultats de validation des entraînements et des tests. Pour plus d'informations sur les règles de validation, consultez Règles de validation des fichiers manifestes.

> **Note**
>
> L'entraînement échoue quand il existe trop d'erreurs de ligne JSON.

Nous vous recommandons de corriger également les erreurs non définitives de ligne JSON, car elles peuvent potentiellement provoquer de futures erreurs ou avoir un impact sur l'entraînement de votre modèle.

La fonctionnalité Étiquettes personnalisées Amazon Rekognition peut générer les erreurs non définitives de validation de ligne JSON suivantes.

- La clé source-ref est manquante.
- Le format de la valeur source-ref 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.
- La valeur de confiance d'un ou de plusieurs cadres de délimitation est 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 d'accès S3 et/ou les propriétés de l'image.
- Le cadre de délimitation comporte des valeurs hors cadre.
- La hauteur et la largeur du cadre de délimitation sont trop petites.
- Il existe plus de cadres de délimitation que le maximum autorisé.
- Aucune annotation valide n'a été trouvée.

Présentation du récapitulatif du manifeste

Le récapitulatif du manifeste comprend les informations suivantes.

- Informations sur les Liste des erreurs de contenu du manifeste du terminal rencontrées lors de la validation.
- Informations sur l'emplacement des Liste des erreurs de validation de ligne JSON non terminales dans les jeux de données d'entraînement et de test.
- Statistiques concernant les erreurs, comme le nombre total de lignes JSON non valides trouvées dans les jeux de données d'entraînement et de test.

Le récapitulatif du manifeste est créé pendant l'entraînement quand il n'y a pas d'Liste des erreurs du fichier manifeste du terminal. Pour connaître l'emplacement du fichier récapitulatif du manifeste (manifest_summary.json), consultez Obtention des résultats de validation.

> **Note**
>
> Les erreurs de service et les erreurs de fichier manifeste ne sont pas signalées dans le récapitulatif du manifeste. Pour plus d'informations, consultez Erreurs définitives.

Pour plus d'informations sur les erreurs spécifiques de contenu du manifeste, consultez Erreurs définitives de contenu de manifeste.

Format du fichier récapitulatif du manifeste

Un fichier manifeste comporte 2 sections, `statistics` et `errors`.

statistics

`statistics` contient des informations sur les erreurs détectées dans les jeux 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 le jeu de données de test.

Les objets du tableau `errors` contiennent le code d'erreur et le message relatifs aux erreurs de contenu du manifeste.

Le tableau `error_line_indices` contient le numéro de chaque ligne JSON du manifeste d'entraînement ou de test présentant une erreur. Pour plus d'informations, consultez Correction des erreurs d'entraînement.

errors

Erreurs couvrant à la fois le jeu de données d'entraînement et le jeu de données de test. Par exemple, une erreur ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP se produit quand il n'y a pas assez d'étiquettes utilisables chevauchant les jeux 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 récapitulatif partiel du manifeste qui montre une erreur définitive de contenu (ERROR_TOO_MANY_INVALID_ROWS_IN_IN_MANIFEST). Le tableau `error_json_line_indices` contient les numéros des lignes JSON comportant des erreurs non définitives dans le manifeste de validation d'entraînement 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
            ]
        }
    }
}
```

Présentation des manifestes de résultats de validation des entraînements et des tests

Pendant un entraînement, la fonctionnalité Étiquettes personnalisées Amazon Rekognition crée des manifestes de résultats de validation pour répertorier les erreurs non définitives de ligne JSON. Les manifestes de résultats de validation sont des copies des jeux de données d'entraînement et de test comportant les informations sur les erreurs. Vous pouvez accéder aux manifestes de validation une

fois l'entraînement terminé. Pour plus d'informations, consultez Obtention des résultats de validation. La fonctionnalité Étiquettes personnalisées Amazon Rekognition crée également un récapitulatif du manifeste qui inclut des informations générales sur les erreurs de ligne JSON, comme leur emplacement et leur nombre. Pour plus d'informations, consultez Présentation du récapitulatif du manifeste.

> **Note**
>
> Les résultats de validation (Manifestes de résultats de validation de l'entraînement et des tests et Récapitulatif du manifeste) ne sont créés que s'il n'y en a pas d'Liste des erreurs du fichier manifeste du terminal.

Un manifeste contient des lignes JSON pour chaque image du jeu de données. Dans les manifestes de résultats de validation, les informations concernant les erreurs de ligne JSON sont ajoutées aux lignes JSON concernées.

Une erreur de ligne JSON est une erreur non définitive liée à une seule image. Une erreur de validation non définitive peut invalider la totalité ou une partie d'une ligne JSON. Par exemple, si l'image référencée dans une ligne JSON n'est pas au format PNG ou JPG, une erreur ERROR_INVALID_IMAGE se produit et l'intégralité de la ligne JSON est exclue de l'entraînement. L'entraînement se poursuit avec les 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 de 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 délimitation valides. Les informations d'erreur de ligne JSON sont renvoyées pour le cadre de sélection non valide (ERROR_INVALID_BOUNDING_BOX). Dans cet exemple, les informations d'erreur sont ajoutées à l'objet `annotation` où l'erreur se produit.

Les erreurs d'avertissement, comme WARNING_NO_ANNOTATIONS, ne sont pas utilisées pour l'entraînement et sont considérées comme lignes JSON ignorées (`ignored_json_lines`) dans le récapitulatif du manifeste. Pour plus d'informations, consultez Présentation du récapitulatif du manifeste. De plus, les lignes JSON ignorées ne sont pas prises en compte dans le seuil d'erreur de 20 % applicable à l'entraînement et aux tests.

Pour plus d'informations sur les erreurs non définitives spécifiques à la validation des données, consultez Erreurs non définitives de validation de ligne JSON.

> **Note**
>
> S'il existe trop d'erreurs de validation des données, l'entraînement est arrêté et une erreur définitive ERROR_TOO_MANY_INVALID_ROWS_IN_IN_MANIFEST est signalée dans le récapitulatif du manifeste.

Pour plus d'informations sur la correction des erreurs de ligne JSON, consultez Correction des erreurs d'entraînement.

Format d'une erreur de ligne JSON

La fonctionnalité Étiquettes personnalisées Amazon Rekognition ajoute les informations d'erreur non définitive de validation aux lignes JSON de format de localisation d'objet et de niveau image. Pour plus d'informations, consultez the section called "Création d'un fichier manifeste".

Erreurs de niveau image

L'exemple suivant montre les tableaux `Error` d'une ligne JSON de niveau image. Il existe deux types d'erreurs. Erreurs liées aux métadonnées des attributs d'étiquette (dans cet exemple, sport-metadata) et erreurs liées à l'image. Une erreur inclut un code d'erreur (code), un message d'erreur (message). Pour plus d'informations, consultez Importation d'étiquettes au niveau de l'image dans des 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'objet

L'exemple suivant montre les tableaux d'erreurs dans une ligne JSON de localisation d'objet. La ligne JSON contient un tableau d'informations `Errors` pour les champs des sections de ligne JSON suivantes. Chaque objet `Error` inclut le code d'erreur et le message d'erreur.

- attribut d'étiquette : erreurs concernant les champs d'attributs d'étiquettes. Voir `bounding-box` dans l'exemple.
- annotations : les erreurs d'annotation (cadres de délimitation) sont stockées dans le tableau `annotations` à l'intérieur de l'attribut d'étiquette.
- attribut d'étiquette-métadonnées : erreurs concernant les métadonnées d'attributs d'étiquettes. Voir `bounding-box-metadata` dans l'exemple.
- image : erreurs non liées aux champs de métadonnées d'attribut d'étiquette, d'annotation et d'attribut d'étiquette.

Pour plus d'informations, consultez 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'objet suivante (formatée pour plus de lisibilité) affiche une erreur ERROR_BOUNDING_BOX_TOO_SMALL. 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"
    }
}
```

Obtention des résultats de validation

Les résultats de validation contiennent des informations pour les Liste des erreurs de contenu du manifeste du terminal et les Liste des erreurs de validation de ligne JSON non terminales. Il existe trois fichiers de résultats de validation.

- training_manifest_with_validation.json : une copie du fichier manifeste du jeu de données d'entraînement à laquelle sont ajoutées les informations d'erreur de ligne JSON.
- testing_manifest_with_validation.json : une copie du fichier manifeste du jeu de données de test à laquelle sont ajoutées les informations d'erreur de ligne JSON.
- manifest_summary.json : un récapitulatif des erreurs de contenu du manifeste et des erreurs de ligne JSON détectées dans les jeux de données d'entraînement et de test. Pour plus d'informations, consultez Présentation du récapitulatif du manifeste.

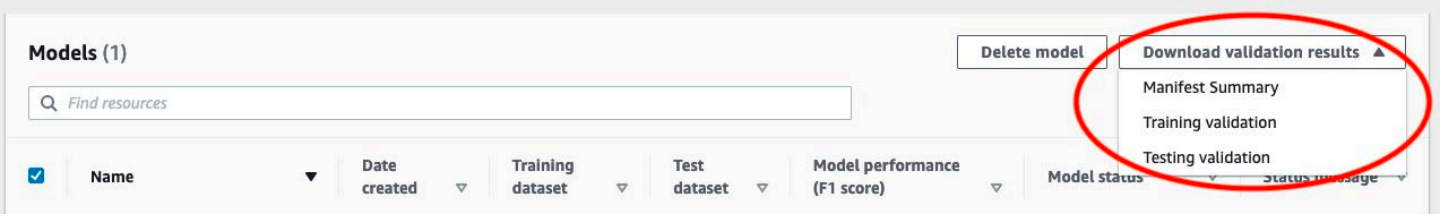
Pour plus d'informations sur le contenu des manifestes de validation des entraînements et des tests, consultez Débogage d'un entraînement de modèle en échec.

> **Note**
>
> - Les résultats de validation ne sont créés que si des Liste des erreurs du fichier manifeste du terminal ne sont pas générées pendant l'entraînement.
> - Si une erreur de service survient après la validation du manifeste de formation et de test, les résultats de validation sont créés, mais la réponse de DescribeProjectVersionsn'inclut pas les emplacements des fichiers de résultats de validation.

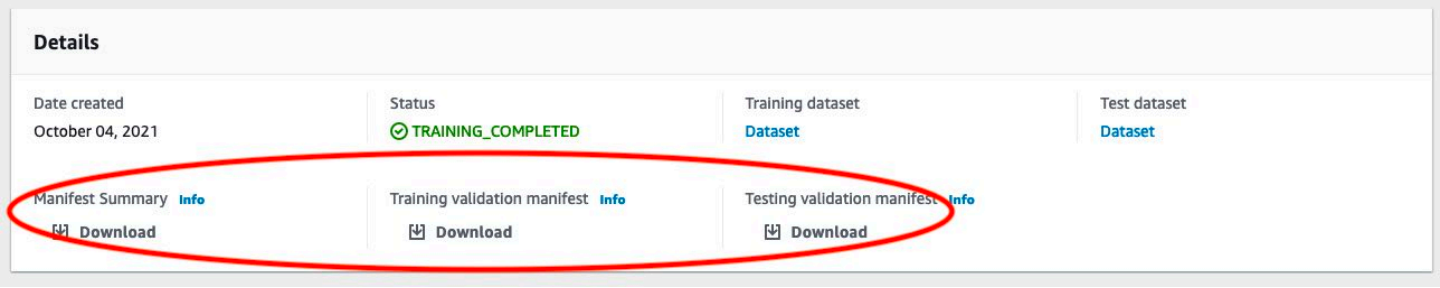
Une fois la formation terminée ou échouée, vous pouvez télécharger les résultats de validation à l'aide de la console Amazon Rekognition Custom Labels ou obtenir l'emplacement du compartiment Amazon S3 en appelant l'API. DescribeProjectVersions

Obtention des 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 indiqué dans le schéma suivant. Le panneau Modèles affiche les résultats de l'entraînement et de la validation des modèles avec la possibilité de télécharger les résultats de validation.



Vous pouvez également télécharger les résultats de validation depuis la page des détails d'un modèle. La page de détails présente les détails de l'ensemble de données avec le statut, les ensembles de données d'entraînement et de test, ainsi que des liens de téléchargement pour le résumé du manifeste, le manifeste de validation de la formation et le manifeste de validation des tests.



Pour de plus amples informations, veuillez consulter Entraînement d'un modèle (console).

Obtention des résultats de validation (kit SDK)

Une fois l'entraînement du modèle terminé, la fonctionnalité Étiquettes personnalisées Amazon Rekognition stocke les résultats de validation dans le compartiment Amazon S3 spécifié pendant l'entraînement. Vous pouvez obtenir l'emplacement du compartiment S3 en appelant l'DescribeProjectVersionsAPI une fois la formation terminée. Pour entraîner un modèle, consultez Entraînement d'un modèle (kit SDK).

Un ValidationDataobjet est renvoyé pour l'ensemble de données d'entraînement (TrainingDataResult) et l'ensemble de données de test (TestingDataResult). Le récapitulatif du manifeste est renvoyé dans `ManifestSummary`.

Après avoir obtenu l'emplacement du compartiment Amazon S3, vous pouvez télécharger les résultats de validation. Pour plus d'informations, consultez Pour télécharger un objet à partir d'un compartiment S3. Vous pouvez également utiliser l'opération GetObject.

Pour obtenir des données de validation (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez l'exemple suivant pour obtenir l'emplacement des résultats de validation.

   Python

   Remplacez `project_arn` par l'Amazon Resource Name (ARN) du projet qui contient le modèle. Pour plus d'informations, consultez Gestion d'un projet Étiquettes personnalisées Amazon Rekognition. Remplacez `version_name` par le nom de version du modèle. Pour plus d'informations, consultez Entraînement d'un modèle (kit 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. En sortie de programme, notez le champ `Validation` situé dans les objets `TestingDataResult` et `TrainingDataResult`. Le récapitulatif du manifeste est dans `ManifestSummary`.

Correction des erreurs d'entraînement

Vous utilisez le récapitulatif du manifeste pour identifier les Liste des erreurs de contenu du manifeste du terminal et les Liste des erreurs de validation de ligne JSON non terminales rencontrées pendant l'entraînement. Vous devez corriger les erreurs de contenu du manifeste. Nous recommandons de corriger également les erreurs non définitives de ligne JSON. Pour plus d'informations sur des erreurs spécifiques, consultez Erreurs non définitives de validation de ligne JSON et Erreurs définitives de contenu de manifeste.

Vous pouvez apporter des corrections au jeu de données d'entraînement ou de test utilisé pour l'entraînement. Autrement, vous pouvez aussi apporter les corrections dans les fichiers manifestes de validation d'entraînement et de test, et les utiliser pour entraîner le modèle.

Une fois les corrections effectuées, vous devez importer les manifestes mis à jour et réentraîner le modèle. Pour plus d'informations, consultez Création d'un fichier manifeste.

La procédure suivante montre comment utiliser le récapitulatif du manifeste pour corriger les erreurs définitives de contenu de manifeste. 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 Étiquettes personnalisées Amazon Rekognition

1. Téléchargez les fichiers de résultats de validation. Les noms de fichiers sont training_manifest_with_validation.json, testing_manifest_with_validation.json et manifest_summary.json. Pour plus d'informations, consultez Obtention des résultats de validation.
2. Ouvrez le fichier récapitulatif du manifeste (manifest_summary.json).
3. Corrigez les erreurs éventuelles dans le récapitulatif du manifeste. Pour plus d'informations, consultez Présentation du récapitulatif du manifeste.
4. Dans le récapitulatif du manifeste, parcourez le tableau `error_line_indices` dans `training`, puis corrigez les erreurs dans le fichier `training_manifest_with_validation.json` aux numéros de ligne JSON correspondants. Pour plus d'informations, consultez the section called "Présentation des manifestes de résultats de validation des entraînements et des tests".
5. Parcourez le tableau `error_line_indices` dans `testing`, puis corrigez les erreurs dans le fichier `testing_manifest_with_validation.json` aux numéros de ligne JSON correspondants.
6. Formez de nouveau le modèle en utilisant les fichiers manifestes de validation comme jeux de données d'entraînement et de test. Pour de plus amples 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 des données d'apprentissage ou de validation des tests, utilisez l'emplacement des fichiers manifestes de données de validation dans les paramètres TrainingDataet TestingDatad'entrée pour CreateProjectVersion. Pour de plus amples informations, veuillez consulter Entraînement d'un modèle (kit 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 toute autre erreur de ligne JSON.

- MISSING_SOURCE_REF
- ERROR_INVALID_SOURCE_REF_FORMAT
- 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 définitives de fichier manifeste

Cette rubrique décrit les Liste des erreurs du fichier manifeste du terminal. Aucun code d'erreur n'est associé aux erreurs de fichier manifeste. Aucun manifeste de résultats de validation n'est créé lorsqu'une erreur définitive de fichier manifeste se produit. Pour plus d'informations, consultez Présentation du récapitulatif du manifeste. Les erreurs définitives de manifeste empêchent le signalement des Erreurs non définitives de validation de ligne JSON.

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 Le contenu ou l'extension 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 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.

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 utilisée pour l'entraînement et les tests.

Pour corriger l'erreur Le fichier manifeste est vide.

1. Vérifiez quels manifestes d'entraînement ou de test sont vides.

2. Ajoutez des lignes JSON au fichier manifeste vide. Pour plus d'informations, consultez Création d'un fichier manifeste. Autrement, vous pouvez aussi créer un jeu de données avec la console. Pour plus d'informations, consultez 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 (en octets) du fichier manifeste d'entraînement ou de test est trop importante. Pour plus d'informations, consultez Directives et quotas dans Étiquettes personnalisées 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 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 dans les fichiers manifestes dont la taille est trop importante. Pour plus d'informations, consultez Création d'un fichier manifeste.

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 Les autorisations du compartiment S3 sont incorrectes.

- 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.

Impossible d'écrire dans le compartiment S3 de sortie.

Le service n'est pas en mesure de générer les fichiers de sortie d'entraînement.

Pour corriger l'erreur Impossible d'écrire dans le compartiment S3 de sortie.

- Vérifiez que les informations du compartiment Amazon S3 contenues dans le paramètre OutputConfigd'entrée CreateProjectVersionsont correctes.

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

Erreurs définitives de contenu de manifeste

Cette rubrique décrit les Liste des erreurs de contenu du manifeste du terminal signalées dans le récapitulatif du manifeste. Le récapitulatif du manifeste inclut un code d'erreur et un message pour chaque erreur détectée. Pour plus d'informations, consultez Présentation du récapitulatif du manifeste. Les erreurs définitives de contenu de manifeste n'empêchent pas le signalement des Liste des erreurs de validation de ligne JSON non terminales.

ERROR_TOO_MANY_INVALID_ROWS_IN_IN_MANIFEST

Message d'erreur

Le fichier manifeste contient trop de lignes non valides.

En savoir plus

Une erreur `ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST` se produit quand trop de lignes JSON contiennent du contenu non valide.

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger une erreur `ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST`.

Pour corriger l'erreur ERROR_TOO_MANY_INVALID_ROWS_IN_IN_MANIFEST

1. Vérifiez le manifeste pour détecter les erreurs de ligne JSON. Pour plus d'informations, consultez Présentation des manifestes de résultats de validation des entraînements et des tests.
2. Corrigez les lignes JSON présentant des erreurs. Pour plus d'informations, consultez Erreurs non définitives de validation de ligne JSON.

ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS

Message d'erreur

Le fichier manifeste contient des images provenant de plusieurs compartiments S3.

En savoir plus

Un manifeste ne peut référencer que des images stockées dans un seul compartiment. Chaque ligne JSON stocke l'emplacement Amazon S3 de l'emplacement d'une image sous forme de valeur `source-ref`. Dans l'exemple suivant, le nom du compartiment est my-bucket.

```
"source-ref": "s3://my-bucket/images/sunrise.png"
```

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

Pour corriger l'erreur **`ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS`**

- Assurez-vous que toutes vos images se trouvent dans le même compartiment Amazon S3 et que la valeur `source-ref` de chaque ligne JSON fait référence au compartiment dans lequel vos images sont stockées. Autrement, vous pouvez aussi choisir un compartiment Amazon S3 préféré, et supprimer les lignes JSON où `source-ref` ne fait pas référence à ce dernier.

ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET

Message d'erreur

Les autorisations du compartiment S3 d'images ne sont pas valides.

En savoir plus

Les autorisations du compartiment Amazon S3 contenant les images sont incorrectes.

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

Pour corriger l'erreur **`ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET`**

- Vérifiez les autorisations du compartiment contenant les images. La valeur `source-ref` d'une image contient l'emplacement du compartiment.

ERROR_INVALID_IMAGES_S3_BUCKET_OWNER

Message d'erreur

Identifiant de propriétaire non valide pour le compartiment S3 d'images.

En savoir plus

Le propriétaire du compartiment contenant les images d'entraînement ou de test est différent de celui du compartiment contenant le manifeste d'entraînement ou de test. Vous pouvez utiliser la commande suivante pour rechercher le propriétaire d'un compartiment.

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

L'élément `OWNER ID` doit correspondre à celui des compartiments stockant les images et les fichiers manifestes.

Pour corriger l'erreur ERROR_INVALID_IMAGES_S3_BUCKET_OWNER

1. Choisissez le propriétaire souhaité des compartiments d'entraînement, de test, de sortie et d'images. Celui-ci doit être autorisé à utiliser la fonctionnalité Étiquettes personnalisées Amazon Rekognition.
2. Pour chaque compartiment n'appartenant pas actuellement au propriétaire souhaité, créez un 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 copier tous les objets d'un compartiment Amazon S3 vers un autre ?.

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT

Message d'erreur

Le fichier manifeste ne contient pas suffisamment d'images étiquetées par étiquette pour permettre le fractionnement 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. L'erreur ERROR_INSUFFICIENT_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 Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

Pour corriger l'erreur ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT

- Ajoutez d'autres images étiquetées à votre jeu de données d'entraînement. Il est possible d'ajouter des images dans la console Étiquettes personnalisées Amazon Rekognition en ajoutant des images au jeu de données d'entraînement, ou des lignes JSON au manifeste d'entraînement. Pour plus d'informations, consultez Gestion des jeux de données.

ERROR_MANIFEST_TOO_FEW_LABELS

Message d'erreur

Le fichier manifeste contient trop peu d'étiquettes.

En savoir plus

Les jeux de données d'entraînement et de test exigent un nombre minimum d'étiquettes. Ce minimum dépend du fait que le jeu de données forme ou teste un modèle pour détecter les étiquettes de niveau image (classification), ou que le modèle détecte les emplacements d'objets. Si le jeu de données d'entraînement est fractionné pour créer un jeu de données de test, le nombre d'étiquettes du jeu de données est déterminé après le fractionnement du jeu de données d'entraînement. Pour plus d'informations, consultez Directives et quotas dans Étiquettes personnalisées Amazon Rekognition.

Pour corriger l'erreur ERROR_MANIFEST_TOO_FEW_LABELS (Console)

1. Ajoutez de nouvelles étiquettes au jeu de données. Pour plus d'informations, consultez Gestion des étiquettes.
2. Ajoutez les nouvelles étiquettes aux images du jeu de données. Si votre modèle détecte des étiquettes de niveau image, consultez Attribution d'étiquettes au niveau de l'image à une image.

Si votre modèle détecte les emplacements d'objets, consultez the section called "Étiquetage des objets à l'aide de cadres de délimitation".

Pour corriger l'erreur ERROR_MANIFEST_TOO_FEW_LABELS (Ligne JSON)

- Ajoutez des lignes JSON pour les nouvelles images comportant de nouvelles étiquettes. Pour plus d'informations, consultez Création d'un fichier manifeste. Quand votre modèle détecte des étiquettes de niveau image, vous ajoutez de nouveaux noms d'étiquettes au champ `class-name`. 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 les emplacements d'objets, ajoutez de nouvelles étiquettes à `class-map`, comme illustré dans 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 des cadres de délimitation. Pour plus d'informations, consultez Localisation d'objets dans les fichiers manifestes.

ERROR_MANIFEST_TOO_MANY_LABELS

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'objets : 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_LABEL_OVERLAP

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 d'étiquette du jeu de données de test et ceux du jeu de données d'entraînement.

Pour corriger l'erreur ERROR_INSUFFICIENT_LABEL_OVERLAP (Console)

- Supprimez les étiquettes du jeu de données d'entraînement. Autrement, vous pouvez aussi ajouter des étiquettes plus courantes à votre jeu de données de test. 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_INSUFFICIENT_LABEL_OVERLAP en supprimant les étiquettes du jeu de données d'entraînement (Ligne JSON)

- Manifestes avec lignes JSON de niveau image : si l'image possède une seule étiquette, supprimez la ligne JSON de l'image 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. Procédez ainsi pour chaque ligne JSON du manifeste contenant l'étiquette que vous souhaitez supprimer.

  Manifestes avec lignes JSON d'emplacement d'objets : 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.

Pour corriger l'erreur ERROR_INSUFFICIENT_LABEL_OVERLAP en ajoutant des étiquettes communes au jeu de données de test (Ligne JSON)

- Ajoutez au jeu de données de test des lignes JSON incluant des images étiquetées avec des étiquettes déjà présentes dans le jeu de données d'entraînement. Pour plus d'informations, consultez Création d'un fichier manifeste.

ERROR_MANIFEST_TOO_FEW_USABLE_LABELS

Message d'erreur

Le fichier manifeste contient trop peu d'étiquettes utilisables.

En savoir plus

Un manifeste d'entraînement peut contenir des lignes JSON au format d'étiquette de niveau image et au format d'emplacement d'objets. En fonction du type de lignes JSON trouvé dans le manifeste d'entraînement, la fonctionnalité Étiquettes personnalisées Amazon Rekognition choisit de créer un modèle qui détecte les étiquettes de niveau image, ou un modèle qui détecte les emplacements d'objets. La fonctionnalité Étiquettes personnalisées Amazon Rekognition filtre les enregistrements JSON valides pour les lignes JSON qui ne correspondent pas au format

choisi. L'erreur 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.

Un minimum d'une étiquette est obligatoire pour entraîner un modèle détectant les étiquettes de niveau image. Un minimum de deux étiquettes est obligatoire pour entraîner un modèle détectant les emplacements d'objets.

Pour corriger l'erreur ERROR_MANIFEST_TOO_FEW_USABLE_LABELS (Console)

1. Vérifiez le champ `use_case` dans le récapitulatif du manifeste.
2. Ajoutez d'autres étiquettes au jeu de données d'entraînement pour le cas d'utilisation (niveau image ou localisation d'objet) correspondant à la valeur de `use_case`. 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_FEW_USABLE_LABELS (Ligne JSON)

1. Vérifiez le champ `use_case` dans le récapitulatif du manifeste.
2. Ajoutez d'autres étiquettes au jeu de données d'entraînement pour le cas d'utilisation (niveau image ou localisation d'objet) correspondant à la valeur de `use_case`. Pour plus d'informations, consultez Création d'un fichier manifeste.

ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP

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 de niveau image et au format d'emplacement d'objets. En fonction des formats trouvés dans le manifeste d'entraînement, la fonctionnalité Étiquettes personnalisées Amazon Rekognition choisit de créer un modèle qui détecte les étiquettes de niveau image, ou un modèle qui détecte les emplacements d'objets. La fonctionnalité Étiquettes personnalisées Amazon Rekognition les enregistrements JSON valides pour les lignes JSON qui ne correspondent pas au 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 d'entraînement et de test utilisées.

Pour corriger l'erreur ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP (Console)

- Supprimez les étiquettes du jeu de données d'entraînement. Autrement, vous pouvez aussi ajouter des étiquettes plus courantes à votre jeu de données de test. 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_INSUFFICIENT_USABLE_LABEL_OVERLAP en supprimant les étiquettes du jeu de données d'entraînement (Ligne JSON)

- Jeux de données utilisés pour détecter les étiquettes de niveau image : si l'image possède 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, consultez Ajout de plusieurs étiquettes au niveau de l'image à une image. Procédez ainsi pour chaque ligne JSON du manifeste contenant l'étiquette que vous souhaitez supprimer.

  Jeux de données utilisés pour détecter les emplacements d'objets : 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.

Pour corriger l'erreur ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP en ajoutant des étiquettes communes au jeu de données de test (Ligne JSON)

- Ajoutez au jeu de données de test des lignes JSON incluant des images étiquetées avec des étiquettes déjà présentes dans le jeu de données d'entraînement. Pour plus d'informations, consultez Création d'un fichier manifeste.

ERROR_FAILED_IMAGES_S3_COPY

Message d'erreur

Échec de la copie d'images à partir du compartiment S3.

En savoir plus

Le service n'a pu copier aucune des images du jeu de données.

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

Pour corriger l'erreur ERROR_FAILED_IMAGES_S3_COPY

1. Vérifiez les autorisations associées à vos images.
2. Si vous en utilisez AWS KMS, vérifiez la politique relative aux compartiments. Pour de plus amples informations, veuillez consulter Déchiffrer des fichiers chiffrés avec AWS Key Management Service.

Le fichier manifeste contient trop d'erreurs définitives.

Il existe trop de lignes JSON contenant des erreurs définitives de contenu.

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 définitives de contenu de manifeste.

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

Erreurs non définitives de validation de ligne JSON

Cette rubrique répertorie les erreurs non définitives de ligne JSON signalées par la fonctionnalité Étiquettes personnalisées Amazon Rekognition lors de l'entraînement. Les erreurs sont signalées dans le manifeste de validation des entraînements et des tests. Pour plus d'informations, consultez Présentation des manifestes de résultats de validation des entraînements et des tests. Vous pouvez

corriger une erreur non définitive de ligne JSON en mettant à jour la ligne JSON dans le fichier manifeste d'entraînement 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 définitives, 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, consultez Création d'un fichier manifeste. Pour plus d'informations sur la correction des erreurs de validation, consultez Correction des erreurs d'entraînement. Il est possible de corriger certaines erreurs à l'aide de la console Étiquettes personnalisées Amazon Rekognition.

ERROR_MISSING_SOURCE_REF

Message d'erreur

La clé source-ref est manquante.

En savoir plus

Le champ `source-ref` de ligne JSON indique l'emplacement Amazon S3 d'une image. Cette erreur se produit lorsque la clé `source-ref` est manquante ou mal orthographiée. Cette erreur se produit généralement dans les fichiers manifestes créés manuellement. Pour plus d'informations, consultez Création d'un fichier manifeste.

Pour corriger l'erreur **`ERROR_MISSING_SOURCE_REF`**

1. Vérifiez que la clé `source-ref` est présente et qu'elle est correctement orthographiée. Une clé `source-ref` et une valeur complètes ressemblent à ceci : `"source-ref": "s3://bucket/path/image"`.
2. Mettez à jour la clé `source-ref` dans la ligne JSON. Autrement, vous pouvez aussi supprimer la ligne JSON du fichier manifeste.

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

ERROR_INVALID_SOURCE_REF_FORMAT

Message d'erreur

Le format de la valeur source-ref n'est pas valide.

En savoir plus

La clé `source-ref` 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, consultez Création d'un fichier manifeste.

Pour corriger l'erreur **`ERROR_INVALID_SOURCE_REF_FORMAT`**

1. Vérifiez que le schéma suit bien le modèle `"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 Étiquettes personnalisées Amazon Rekognition pour corriger l'erreur `ERROR_INVALID_SOURCE_REF_FORMAT`.

ERROR_NO_LABEL_ATTRIBUTES

Message d'erreur

Aucun attribut d'étiquette n'a été trouvé.

En savoir plus

L'attribut d'étiquette ou le nom de clé `-metadata` de l'attribut d'étiquette (ou les deux) n'est pas valide ou est manquant. Dans l'exemple suivant, l'erreur `ERROR_NO_LABEL_ATTRIBUTES` se produit chaque fois que la clé `bounding-box` ou `bounding-box-metadata` (ou les deux) est manquante. Pour plus d'informations, consultez 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 erreur `ERROR_NO_LABEL_ATTRIBUTES` se produit généralement dans un fichier manifeste créé manuellement. Pour plus d'informations, consultez Création d'un fichier manifeste.

Pour corriger l'erreur **`ERROR_NO_LABEL_ATTRIBUTES`**

1. Vérifiez que l'identifiant d'attribut d'étiquette et les clés `-metadata` d'identifiant d'attribut d'étiquette sont présents, et que les noms de clés sont correctement orthographiés.
2. Mettez à jour ou supprimez la ligne JSON dans le fichier manifeste.

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger l'erreur `ERROR_NO_LABEL_ATTRIBUTES`.

ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT

Message d'erreur

Le format de l'attribut d'étiquette {} n'est pas valide.

En savoir plus

Le schéma de la clé de l'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 corriger l'erreur **`ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT`**

1. Vérifiez que la section de ligne JSON Line est correcte pour la clé de l'attribut d'étiquette. Dans l'exemple d'emplacement d'objets suivant, les objets `image_size` et `annotations` doivent être corrects. La clé de l'attribut d'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 Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT

Message d'erreur

Le format des métadonnées de l'attribut d'étiquette n'est pas valide.

En savoir plus

Le schéma de la clé des métadonnées de l'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 corriger l'erreur **`ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT`**

1. Vérifiez que le schéma de ligne JSON pour la clé des métadonnées de l'attribut d'étiquette est similaire à celui de l'exemple suivant. La clé des 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 Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

ERROR_NO_VALID_LABEL_ATTRIBUTES

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. La fonctionnalité Étiquettes personnalisées Amazon Rekognition vérifie à la fois l'attribut d'étiquette et son identifiant. 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 manifeste SageMaker AI pris en charge, Amazon Rekognition Custom Labels la marque comme non valide `ERROR_NO_VALID_LABEL_ATTRIBUTES` et une erreur est signalée. Actuellement, la console Étiquettes personnalisées Amazon Rekognition prend en charge les formats de tâche de classification et de cadre de sélection. Pour plus d'informations, consultez Création d'un fichier manifeste.

Pour corriger l'erreur **`ERROR_NO_VALID_LABEL_ATTRIBUTES`**

1. Vérifiez que la ligne JSON de la clé d'attribut d'étiquette et des métadonnées de l'attribut d'étiquette est correcte.
2. Mettez à jour ou supprimez la ligne JSON dans le fichier manifeste. Pour plus d'informations, consultez the section called "Création d'un fichier manifeste".

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

ERROR_MISSING_BOUNDING_BOX_CONFIDENCE

Message d'erreur

La valeur de confiance d'un ou de plusieurs cadres de délimitation est manquante.

En savoir plus

La clé de confiance est absente pour un ou plusieurs cadres de délimitation d'emplacement d'objets. La clé de confiance pour un cadre de sélection se trouve dans les métadonnées d'attribut d'étiquette, comme illustré dans l'exemple suivant. Une erreur ERROR_MISSING_BOUNDING_BOX_CONFIDENCE se produit généralement dans les fichiers

manifestes créés manuellement. Pour plus d'informations, consultez the section called "Localisation d'objets dans les fichiers manifestes".

```
"bounding-box-metadata": {
 "objects": [{
  "confidence": 1
 }, {
  "confidence": 1
 }],
```

Pour corriger l'erreur **`ERROR_MISSING_BOUNDING_BOX_CONFIDENCE`**

1. Vérifiez dans l'attribut d'étiquette qu'il y a le même nombre de clés de confiance dans le tableau `objects` que d'objets dans le tableau `annotations`.
2. Mettez à jour ou supprimez la ligne JSON dans le fichier manifeste.

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

ERROR_MISSING_CLASS_MAP_ID

Message d'erreur

Un ou plusieurs identifiants de classe sont absents de la carte des classes.

En savoir plus

L'élément `class_id` d'un objet d'annotation (cadre de délimitation) n'a pas d'entrée correspondante dans la carte des classes de métadonnées d'attribut d'étiquette (`class-map`). Pour plus d'informations, consultez 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 l'erreur ERROR_MISSING_CLASS_MAP_ID

1. Vérifiez que la valeur `class_id` de chaque objet d'annotation (cadre de délimitation) correspond à une valeur dans le tableau `class-map`, comme indiqué dans l'exemple suivant. Les tableaux `annotations` et `class_map` 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 Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

ERROR_INVALID_JSON_LINE

Message d'erreur

Le format de la ligne JSON n'est pas valide.

En savoir plus

Un caractère inattendu a été détecté dans la ligne JSON. La ligne JSON est remplacée par une nouvelle ligne JSON contenant uniquement les informations d'erreur. Une erreur ERROR_INVALID_JSON_LINE se produit généralement dans les fichiers manifestes créés manuellement. Pour plus d'informations, consultez the section called "Localisation d'objets dans les fichiers manifestes".

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

Pour corriger l'erreur **`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 qu'aucun des caractères obligatoires `;` ou `,` ne manque.
3. Mettez à jour ou supprimez la ligne JSON dans le fichier manifeste.

ERROR_INVALID_IMAGE

Message d'erreur

L'image n'est pas valide. Vérifiez le chemin d'accès S3 et/ou les propriétés de l'image.

En savoir plus

Le fichier référencé par `source-ref` n'est pas une image valide. Les causes potentielles incluent les proportions, la taille et le format de l'image.

Pour plus d'informations, consultez Directives et quotas.

Pour corriger l'erreur **`ERROR_INVALID_IMAGE`**

1. Vérifiez les points suivants.

- Les proportions de l'image sont inférieures à 20:1.
- La taille de l'image est supérieure à 15 Mo
- L'image est au format PNG ou JPEG.
- Le chemin vers l'image indiqué dans `source-ref` est correct.
- La dimension minimale de l'image est supérieure à 64 x 64 pixels.
- La dimension maximale de l'image est inférieure à 4 096 x 4 096 pixels.

2. Mettez à jour ou supprimez la ligne JSON dans le fichier manifeste.

Vous ne pouvez pas utiliser la console Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

ERROR_INVALID_IMAGE_DIMENSION

Message d'erreur

Les dimensions de l'image ne sont pas conformes à celles autorisées.

En savoir plus

L'image référencée par `source-ref` n'est pas conforme aux dimensions autorisées pour les images. La dimension minimale est 64 pixels. La dimension maximale est 4 096 pixels. L'erreur `ERROR_INVALID_IMAGE_DIMENSION` est signalée pour les images comportant des cadres de délimitation.

Pour plus d'informations, consultez Directives et quotas.

Pour corriger l'erreur **`ERROR_INVALID_IMAGE_DIMENSION`** (Console)

1. Dans le compartiment Amazon S3, mettez à jour l'image avec des dimensions que la fonctionnalité Étiquettes personnalisées Amazon Rekognition peut traiter.
2. Dans la console Étiquettes personnalisées Amazon Rekognition, procédez comme suit :

   a. Supprimez de l'image les cadres de délimitation existants.
   b. Ajoutez de nouveau les cadres de délimitation à l'image.
   c. Enregistrez vos modifications.

   Pour plus d'informations, consultez Étiquetage des objets à l'aide de cadres de délimitation.

Pour corriger l'erreur **`ERROR_INVALID_IMAGE_DIMENSION`** (kit SDK)

1. Dans le compartiment Amazon S3, mettez à jour l'image avec des dimensions que la fonctionnalité Étiquettes personnalisées Amazon Rekognition peut traiter.
2. Obtenez la ligne JSON existante pour l'image en appelant ListDatasetEntries. Pour le paramètre d'entrée `SourceRefContains`, spécifiez l'emplacement Amazon S3 et le nom de fichier de l'image.
3. Appelez UpdateDatasetEntriesset 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 délimitation pour qu'elles correspondent aux dimensions requises du cadre de délimitation de 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"
  }
}
```

ERROR_INVALID_BOUNDING_BOX

Message d'erreur

Le cadre de délimitation comporte des valeurs hors cadre.

En savoir plus

Les informations du cadre de délimitation spécifient une image qui est hors du cadre de l'image ou qui contient des valeurs négatives.

Pour plus d'informations, consultez Directives et quotas.

Pour corriger l'erreur **`ERROR_INVALID_BOUNDING_BOX`**

1. Vérifiez les valeurs des cadres de délimitation du tableau `annotations`.

```
 "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 Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

ERROR_NO_VALID_ANNOTATIONS

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 valides sur les cadres de délimitation.

Pour corriger l'erreur **`ERROR_NO_VALID_ANNOTATIONS`**

1. Mettez à jour le tableau `annotations` pour inclure des objets de cadre de délimitation valides. Vérifiez également que les informations du cadre de délimitation (`confidence` et `class_map`) correspondant dans les métadonnées des attributs d'étiquette sont correctes. Pour plus d'informations, consultez 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 Étiquettes personnalisées Amazon Rekognition pour corriger cette erreur.

ERROR_BOUNDING_BOX_TOO_SMALL

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 l’entraînement, la fonctionnalité Étiquettes personnalisées Amazon Rekognition redimensionne une image si l’une de ses dimensions est supérieure à 1 280 pixels (les images sources ne sont pas affectées). La hauteur et la largeur obtenues pour le cadre de délimitation doivent être supérieures à 1 x 1 pixel. L’emplacement d’un cadre de délimitation est stocké dans

le tableau `annotations` d'une ligne JSON d'emplacement d'objets. Pour plus d'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 l'erreur ERROR_BOUNDING_BOX_TOO_SMALL

- Choisissez l'une des options suivantes :
  - Augmentez la taille des cadres de délimitation qui sont trop petits.
  - Supprimez les cadres de délimitation trop petits. Pour plus d'informations sur la suppression d'un cadre de délimitation, consultez ERROR_TOO_MANY_BOUNDING_BOXES.
  - Supprimez l'image (Ligne JSON) du manifeste.

ERROR_TOO_MANY_BOUNDING_BOXES

Message d'erreur

Il existe plus de cadres de délimitation que le maximum autorisé.

En savoir plus

Le nombre de cadres de délimitation est supérieur à la limite autorisée (50). Vous pouvez supprimer les cadres de délimitation en trop dans la console Étiquettes personnalisées Amazon Rekognition, ou vous pouvez les supprimer de la ligne JSON.

Pour corriger l'erreur **`ERROR_TOO_MANY_BOUNDING_BOXES`** (Console)

1. Choisissez les cadres de délimitation à supprimer.
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 le projet qui contient le jeu de données que vous souhaitez utiliser.
6. Dans la section Ensembles de données, choisissez le jeu de données que vous souhaitez utiliser.
7. Sur la page de la galerie de jeux de données, choisissez Commencer l'étiquetage pour passer en mode d'étiquetage.
8. Choisissez l'image dont vous souhaitez supprimer les cadres de délimitation.
9. Choisissez Dessiner un cadre de délimitation.
10. Dans l'outil de dessin, choisissez le cadre de délimitation que vous souhaitez supprimer.
11. Appuyez sur la touche Suppr de votre clavier pour supprimer le cadre de sélection.
12. Répétez les deux étapes précédentes jusqu'à ce que vous ayez supprimé suffisamment de cadres de délimitation.
13. Choisissez Terminé.
14. Choisissez Enregistrer les Modifications pour enregistrer vos Modifications.
15. Choisissez Quitter pour quitter le mode d'étiquetage.

Pour corriger l'erreur ERROR_TOO_MANY_BOUNDING_BOXES (Ligne JSON)

1. Ouvrez le fichier manifeste et accédez à la ligne JSON où se produit l'erreur ERROR_TOO_MANY_BOUNDING_BOXES.

2. Supprimez les éléments suivants pour chaque cadre de délimitation que vous souhaitez supprimer.

   - Supprimez l'objet `annotation` obligatoire du tableau `annotations`.
   - Supprimez l'objet `confidence` correspondant du tableau `objects` dans les métadonnées de l'attribut d'étiquette.
   - Si elle n'est plus utilisée par d'autres cadres de délimitation, supprimez l'étiquette de l'entrée `class-map`.

   Utilisez l'exemple suivant pour identifier les éléments à supprimer.

```
{
 "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"
 }
}
```

WARNING_UNANNOTATED_RECORD

Message d'avertissement

L'enregistrement n'est pas annoté.

En savoir plus

Une image ajoutée à un jeu de données à l'aide de la console Étiquettes personnalisées Amazon Rekognition n'est 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 l'erreur WARNING_UNANNOTATED_RECORD

- Étiquetez l'image via la console Étiquettes personnalisées Amazon Rekognition. Pour obtenir des instructions, consultez Attribution d'étiquettes au niveau de l'image à une image.

WARNING_NO_ANNOTATIONS

Message d'avertissement

Aucune annotation n'est fournie.

En savoir plus

Une ligne JSON au format de localisation d'objet ne contient aucune information de cadre de délimitation, bien qu'elle ait été 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, consultez Présentation des manifestes de résultats de validation des entraînements 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 l'erreur WARNING_NO_ANNOTATIONS

- Choisissez l'une des options suivantes :
  - Ajoutez les informations du cadre de délimitation (`annotations`) à la ligne JSON. Pour plus d'informations, consultez Localisation d'objets dans les fichiers manifestes.
  - Supprimez l'image (Ligne JSON) du manifeste.

WARNING_NO_ATTRIBUTE_ANNOTATIONS

Message d'avertissement

Aucune annotation d'attribut n'est fournie.

En savoir plus

Une ligne JSON au format de localisation d'objet ne contient aucune information d'annotation de cadre de délimitation, bien qu'elle ait été 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, consultez Présentation des manifestes de résultats de validation des entraînements 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 l'erreur WARNING_NO_ATTRIBUTE_ANNOTATIONS

- Choisissez l'une des options suivantes :
  - Ajoutez un ou plusieurs objets `annotation` de cadre de délimitation à la ligne JSON. Pour plus d'informations, consultez 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 non valide.

ERROR_UNSUPPORTED_USE_CASE_TYPE

Message d'avertissement

En savoir plus

La valeur du champ `type` n'est pas `groundtruth/image-classification` ni `groundtruth/object-detection`. Pour plus d'informations, consultez 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 l'erreur ERROR_UNSUPPORTED_USE_CASE_TYPE

- Choisissez l'une des options suivantes :
  - Remplacez la valeur du champ `type` par `groundtruth/image-classification` ou `groundtruth/object-detection`, selon le type de modèle que vous souhaitez créer. Pour plus d'informations, consultez Création d'un fichier manifeste.
  - Supprimez l'image (Ligne JSON) du manifeste.

ERROR_INVALID_LABEL_NAME_LENGTH

En savoir plus

Le nom d'une étiquette est trop long. 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 entraîné Étiquettes personnalisées Amazon Rekognition

Lorsque l’entraînement est terminé, vous évaluez les performances du modèle. Pour vous aider, Étiquettes personnalisées Amazon Rekognition fournit les métriques récapitulatives et les métriques d’évaluation de chaque étiquette. Pour plus d’informations sur les métriques disponibles, consultez Métriques d’évaluation du modèle. Pour améliorer votre modèle à l’aide des métriques, consultez Amélioration d’un modèle Étiquettes personnalisées Amazon Rekognition.

Si vous êtes satisfait de la précision du modèle, vous pouvez commencer à l’utiliser. Pour plus d’informations, consultez Exécution d’un modèle Étiquettes personnalisées Amazon Rekognition entraîné.

Rubriques

Métriques d’évaluation du modèle

Une fois le modèle formé, Étiquettes personnalisées Amazon Rekognition renvoie les métriques des tests du modèle. Vous pouvez les utiliser pour évaluer les performances du modèle. Cette rubrique décrit les métriques mises à votre disposition et explique comment déterminer si le modèle entraîné fonctionne correctement.

La console Étiquettes personnalisées Amazon Rekognition fournit les métriques suivantes comme résumé des résultats de l’entraînement et comme métriques de chaque étiquette :

- Précision
- Rappel
- F1

Chaque métrique fournie constitue une métrique couramment utilisée pour évaluer les performances d'un modèle de machine learning. Étiquettes personnalisées Amazon Rekognition renvoie les métriques des résultats des tests pour l'ensemble du jeu de données, ainsi que les métriques de chaque étiquette personnalisée. Vous pouvez aussi examiner les performances du modèle entraîné personnalisé pour chaque image du jeu de données des tests. Pour plus d'informations, consultez Accès aux métriques d'évaluation (console).

Évaluation des performances du modèle

Au cours des tests, Étiquettes personnalisées Amazon Rekognition prédit si une image de test contient une étiquette personnalisée. Le score de confiance est une valeur quantifiant la certitude de la prédiction du modèle.

Si le score de confiance d'une étiquette personnalisée dépasse la valeur seuil, la sortie du modèle inclut l'étiquette. Les prédictions peuvent être classées de la manière suivante :

- Vrai positif : le modèle Étiquettes personnalisées Amazon Rekognition prédit correctement la présence de l'étiquette personnalisée dans l'image de test : l'étiquette prédite est également une étiquette de vérité sur le terrain pour l'image. Par exemple, Étiquettes personnalisées Amazon Rekognition renvoie correctement une étiquette de ballon de football lorsqu'un tel ballon est présent dans une image.
- Faux positif : le modèle Étiquettes personnalisées Amazon Rekognition prédit incorrectement la présence d'une étiquette personnalisée dans une image de test : l'étiquette prédite n'est pas une étiquette de vérité sur le terrain pour l'image. Par exemple, Étiquettes personnalisées Amazon Rekognition renvoie une étiquette de ballon de football, mais il n'existe aucune étiquette de ballon de football dans la vérité sur le terrain pour l'image.
- Faux négatif : le modèle Étiquettes personnalisées Amazon Rekognition ne prédit pas la présence d'une étiquette personnalisée dans l'image, mais la vérité sur le terrain de l'image inclut l'étiquette. Par exemple, Étiquettes personnalisées Amazon Rekognition ne renvoie pas d'étiquette personnalisée « ballon de football » pour une image en contenant un.
- Vrai négatif : le modèle Étiquettes personnalisées Amazon Rekognition prédit correctement qu'une étiquette personnalisée n'est pas présente dans l'image de test. Par exemple, Étiquettes personnalisées Amazon Rekognition ne renvoie pas d'étiquette personnalisée « ballon de football » pour une image n'en contenant pas un.

La console permet d'accéder aux valeurs vrai positif, faux positif et faux négatif pour chaque image du jeu de données des tests. Pour plus d'informations, consultez Accès aux métriques d'évaluation (console).

Les résultats de prédiction permettent de calculer les métriques suivantes de chaque étiquette, ainsi qu'un cumul pour la totalité du jeu de données des tests. Les mêmes définitions s'appliquent aux prédictions faites par le modèle au niveau du cadre de délimitation, à la différence que toutes les métriques sont calculées sur chaque cadre de délimitation (prédiction ou vérité sur le terrain) de chaque image de test.

Intersection sur l'union (IoU, Intersection over Union) et détection d'objets

Intersection sur l'union mesure le pourcentage de chevauchement entre deux cadres de délimitation d'objet sur leur surface combinée. La plage est comprise entre 0 (chevauchement le plus faible) et 1 (chevauchement complet). Pendant le test, un cadre de délimitation prédit est correct si l'Intersection sur l'union entre le cadre de délimitation de la vérité sur le terrain et le cadre de délimitation prédit est au moins de 0,5.

Seuil supposé

Étiquettes personnalisées Amazon Rekognition calcule automatiquement une valeur seuil supposée (0-1) pour chacune des étiquettes personnalisées. Vous ne pouvez pas définir la valeur de seuil supposée pour une étiquette personnalisée. Le seuil supposé de chaque étiquette est la valeur au-dessus de laquelle une prédiction est considérée comme « vrai » ou « faux positif ». Il est défini en fonction du jeu de données des tests. Le seuil supposé est calculé sur la base du meilleur score F1 obtenu sur le jeu de données des tests lors de l'entraînement du modèle.

Vous pouvez obtenir la valeur du seuil supposé d'une étiquette à partir des résultats d'entraînement du modèle. Pour plus d'informations, consultez Accès aux métriques d'évaluation (console).

Les modifications des valeurs de seuil supposées sont généralement utilisées pour améliorer la précision et le rappel d'un modèle. Pour plus d'informations, consultez Amélioration d'un modèle Étiquettes personnalisées Amazon Rekognition. Comme il est impossible de définir le seuil supposé du modèle d'une étiquette, vous pouvez obtenir les mêmes résultats en analysant une image avec `DetectCustomLabels` et en spécifiant un paramètre d'entrée `MinConfidence`. Pour plus d'informations, consultez Analyse d'une image avec un modèle entraîné.

Précision

Étiquettes personnalisées Amazon Rekognition fournit les métriques de précision pour chaque étiquette et une métrique de précision moyenne pour la totalité du jeu de données des tests.

La précision est le rapport entre les prédictions correctes (vrais positifs) et toutes les prédictions du modèle (vrais et faux positifs) au seuil supposé d'une étiquette donnée. Si le seuil augmente, il se peut que le modèle propose moins de prédictions. En général, toutefois, le ratio entre les vrais positifs et les faux positifs est plus élevé que le seuil inférieur. Les valeurs de précision possibles sont comprises entre 0 et 1 et les valeurs plus élevées indiquent une précision supérieure.

Par exemple, lorsque le modèle prédit la présence d'un ballon de football dans une image, à quelle fréquence la prédiction est-elle correcte ? Supposons qu'il y ait une image avec 8 ballons de football et 5 rochers. 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 sur l'image avec 8 prédictions correctes et 5 incorrectes, la précision obtenue est inférieure.

Pour plus d'informations, consultez Précision et rappel.

Rappel

Étiquettes personnalisées Amazon Rekognition fournit les métriques de rappel moyen de chaque étiquette et une métrique de rappel moyen pour la totalité du jeu de données des tests.

Le rappel est la fraction des étiquettes du jeu de données des tests 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 lorsqu'elle est réellement présente dans les images du jeu de données des tests. La plage de rappel est comprise entre 0 et 1. Les valeurs plus élevées indiquent un rappel supérieur.

Par exemple, si une image contient 8 ballons de football, combien d'entre eux sont correctement détectés ? Dans l'exemple où une image comporte 8 ballons de football et 5 rochers, si le modèle détecte 5 ballons de football, la valeur de rappel est 0,62. Si, après un réentraînement, le nouveau modèle détecte 9 ballons de football, dont les 8 déjà présents sur l'image, la valeur de rappel est 1,0.

Pour plus d'informations, consultez Précision et rappel.

F1

Étiquettes personnalisées Amazon Rekognition utilise la métrique Score F1 pour mesurer les performances moyennes du modèle de chaque étiquette et les performances moyennes du modèle de la totalité du jeu de données des tests.

La performance du modèle est une mesure de cumul qui prend en compte à la fois la précision et le rappel de toutes les étiquettes (par exemple, le score F1 ou la précision moyenne). Le score de performance 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 représente 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 en termes de précision et de rappel. Si le modèle ne fonctionne pas correctement avec, par exemple, une précision faible 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 que le rappel est faible (0,20), le score F1 est de 0,33. Dans les deux cas, le score F1 est faible et indique des problèmes liés au modèle.

Pour plus d'informations, consultez Score F1.

Utilisation des métriques

Pour un modèle donné que vous avez entraîné et en fonction de l'application, vous pouvez faire un compromis entre la précision et le rappel en définissant le paramètre d'entrée `MinConfidence` sur `DetectCustomLabels`. À une valeur `MinConfidence` plus élevée, vous obtenez généralement une plus grande précision (plus de prédictions correctes sur les ballons de football), mais un rappel plus bas (le nombre manquant de vrais ballons de football sera plus grand). À une valeur `MinConfidence` plus basse, vous obtenez un rappel plus élevé (nombre plus élevé de ballons de football correctement prédits), mais une précision moindre (un plus grand nombre de prédictions seront fausses). Pour plus d'informations, consultez Analyse d'une image avec un modèle entraîné.

Les métriques informent également sur les mesures à prendre pour améliorer les performances du modèle si nécessaire. Pour plus d'informations, consultez Amélioration d'un modèle Étiquettes personnalisées Amazon Rekognition.

> **Note**
>
> `DetectCustomLabels` renvoie les prédictions comprises entre 0 et 100, ce qui correspond à la plage métrique de 0 à 1.

Accès aux métriques d'évaluation (console)

Au cours des tests, les performances du modèle sont évaluées par rapport au jeu de données des tests. Les étiquettes du jeu de données des tests sont considérées comme « vérités sur le terrain », car elles correspondent à ce que représente l'image réelle. Pendant les tests, le modèle fait des prédictions à l'aide du jeu de données des tests. Les étiquettes prédites sont comparées aux étiquettes de vérité sur le terrain et les résultats sont disponibles sur la page d'évaluation de la console.

La console Étiquettes personnalisées Amazon Rekognition affiche les métriques récapitulatives de la totalité du modèle et les métriques des étiquettes individuelles. Les métriques disponibles dans la console sont le rappel de précision, le score F1, la confiance et le seuil de confiance. Pour plus d'informations, consultez Amélioration d'un modèle entraîné Étiquettes personnalisées Amazon Rekognition.

Vous pouvez utiliser la console pour vous concentrer sur des métriques spécifiques. Par exemple, pour étudier les problèmes de précision d'une étiquette, vous pouvez filtrer les résultats de l'entraînement par étiquette et par faux positifs. Pour plus d'informations, consultez Métriques d'évaluation du modèle.

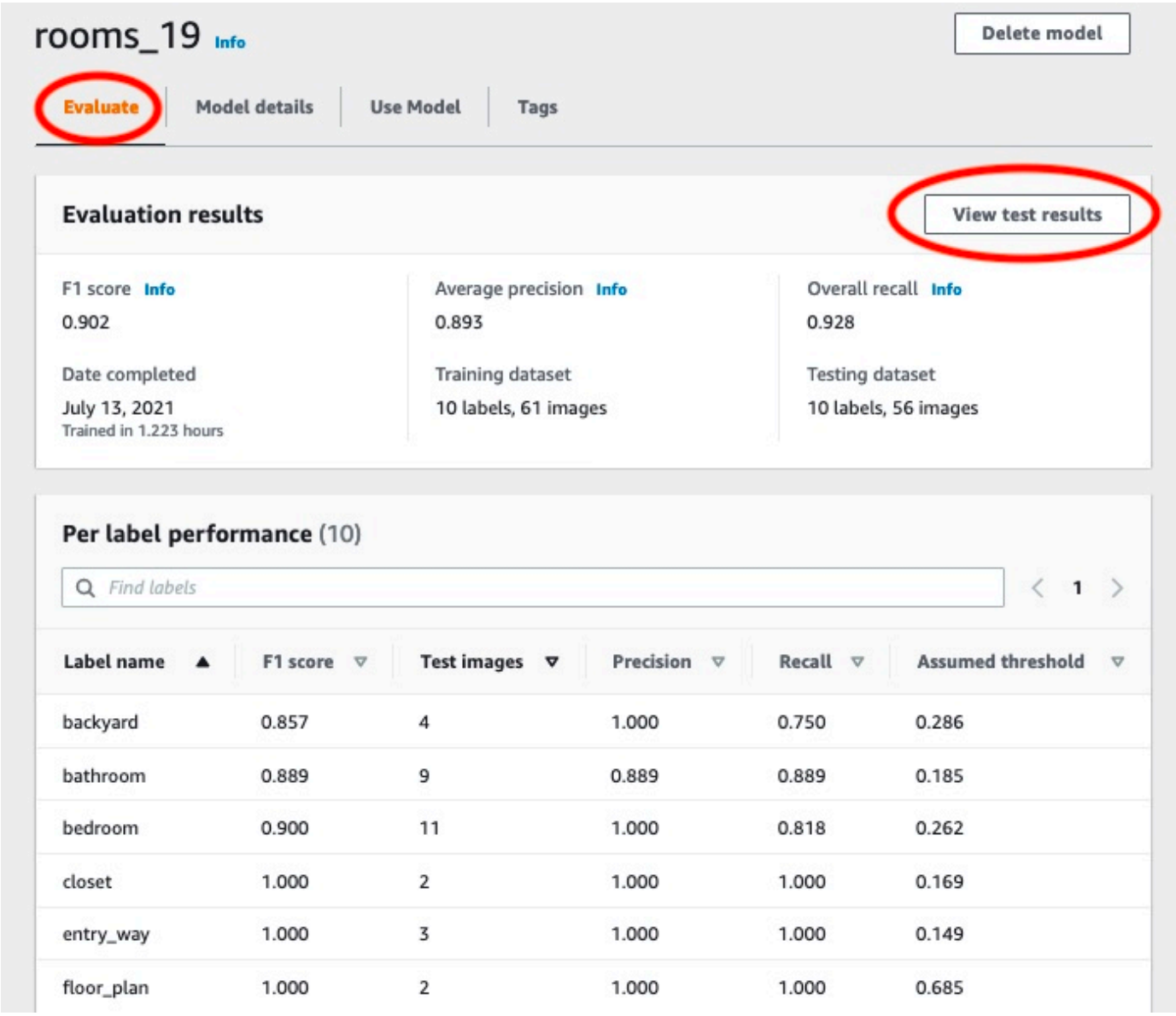
Après l'entraînement, le jeu de données d'entraînement est en lecture seule. Si vous décidez d'améliorer le modèle, vous pouvez copier le jeu de données d'entraînement dans un nouveau jeu de données. Vous utilisez la copie du jeu de données pour entraîner une nouvelle version du modèle.

Dans cette étape, vous utilisez la console pour accéder aux résultats de l'entraînement dans la console.

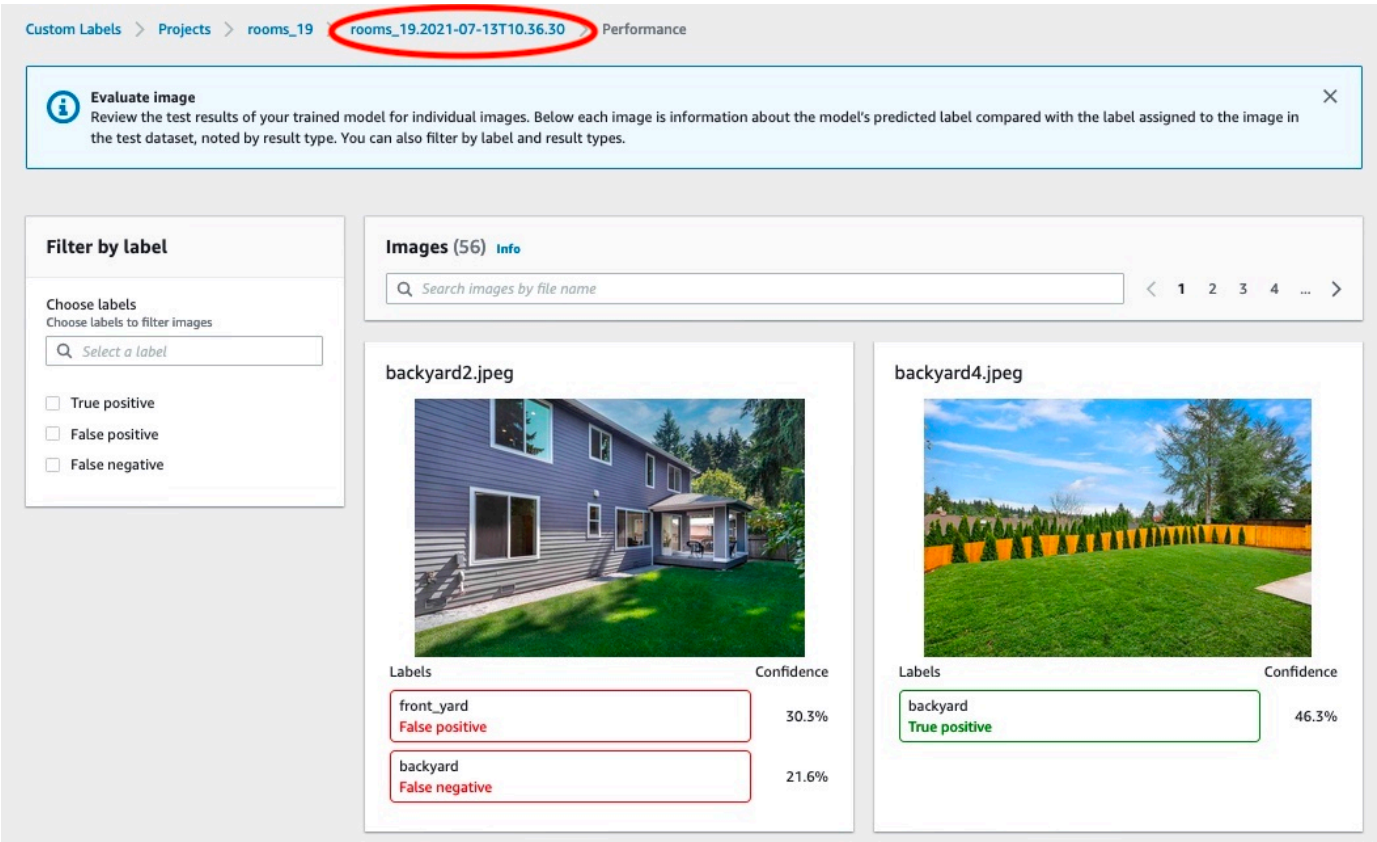
Pour accéder aux métriques d'évaluation (console)

1. Ouvrez la console Amazon Rekognition à l'adresse. https://console.aws.amazon.com/rekognition/
2. Choisissez Utiliser Custom Labels.
3. Choisissez Démarrer.

4. Dans le volet de navigation de gauche, choisissez Projets.
5. Dans la page Projets, choisissez le projet contenant le modèle entraîné à évaluer.
6. Dans Modèles, choisissez le modèle à évaluer.
7. Cliquez sur l'onglet Évaluation pour afficher les résultats de l'évaluation. Pour plus d'informations sur l'évaluation d'un modèle, consultez Amélioration d'un modèle entraîné Étiquettes personnalisées Amazon Rekognition.
8. Choisissez Afficher les résultats des tests pour visualiser les résultats des images des tests. Pour de plus amples informations, veuillez consulter Métriques d'évaluation du modèle. La capture d'écran suivante du résumé de l'évaluation du modèle montre le score F1, la précision moyenne et le rappel global de 6 étiquettes avec les résultats des tests et les mesures de performance. Des détails sur l'utilisation du modèle entraîné sont également fournis.



9. Après avoir consulté les résultats des tests, choisissez le nom du projet pour revenir à la page du modèle. La page des résultats des tests affiche des images avec des étiquettes prédites et des scores de confiance pour un modèle d'apprentissage automatique formé sur les catégories d'images de la cour arrière et de la cour avant. Deux exemples d'images sont affichés.



10. Utilisez les métriques pour évaluer les performances du modèle. Pour plus d'informations, consultez Amélioration d'un modèle Étiquettes personnalisées Amazon Rekognition.

Accès aux métriques d'évaluation d'Étiquettes personnalisées Amazon Rekognition (kit SDK)

L'DescribeProjectVersionsopération permet d'accéder à des métriques autres que celles fournies dans la console.

De même que la console, `DescribeProjectVersions` permet d'accéder aux métriques suivantes sous la forme d'informations récapitulatives pour les résultats des tests et de résultats des tests pour chaque étiquette :

- Précision

- Rappel
- F1

Vous obtenez le seuil moyen de toutes les étiquettes et le seuil des étiquettes individuelles.

`DescribeProjectVersions` donne également accès aux métriques 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, consultez Affichage de la matrice de confusion d'un modèle.
- Précision moyenne médiane (mAP, Mean Average Precision) pour la détection d'images.
- Rappel moyen médian (mAR, Mean Average Recall) pour la détection d'images.

`DescribeProjectVersions` donne également accès aux valeurs vrai positif, faux positif, faux négatif et vrai négatif. Pour plus d'informations, consultez Métriques d'évaluation du modèle.

La métrique Score F1 cumulé est renvoyée directement par `DescribeProjectVersions`. Les autres métriques sont accessibles à partir des fichiers Accès au fichier récapitulatif du modèle et Interprétation de l'instantané du manifeste d'évaluation stockés dans un compartiment Amazon S3. Pour de plus amples informations, veuillez consulter Accès au fichier récapitulatif et à l'instantané du manifeste d'évaluation (kit SDK).

Rubriques

- Accès au fichier récapitulatif du modèle
- Interprétation de l'instantané du manifeste d'évaluation
- Accès au fichier récapitulatif et à l'instantané du manifeste d'évaluation (kit SDK)
- Affichage de la matrice de confusion d'un modèle
- Référence : fichier récapitulatif des résultats de l'entraînement

Accès au fichier récapitulatif du modèle

Le fichier récapitulatif contient les informations sur les résultats de l'évaluation du modèle dans son ensemble, ainsi que les métriques de chaque étiquette. Les métriques sont Précision, Rappel et Score F1. La valeur de seuil du modèle est également fournie. L'emplacement du fichier récapitulatif est accessible depuis l'objet `EvaluationResult` renvoyé par `DescribeProjectVersions`. Pour plus d'informations, consultez Référence : fichier récapitulatif des résultats de l'entraînement.

Voici 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:nnnnnnnnn: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
    }
  ]
}
```

Interprétation de l'instantané du manifeste d'évaluation

L’instantané du manifeste d’évaluation contient les informations détaillées sur les résultats des tests. L’instantané inclut l’indice de confiance pour chaque prédiction. Il inclut aussi 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 représentent un instantané, car ne sont incluses que les images utilisées pour les tests et l’entraînement. Les images qui ne peuvent pas être vérifiées, telles que les images au mauvais format, ne sont pas incluses dans le manifeste. L’emplacement de l’instantané des tests est accessible depuis l’objet `TestingDataResult` renvoyé par `DescribeProjectVersions`. L’emplacement de l’instantané de l’entraînement est accessible depuis l’objet `TrainingDataResult` renvoyé par `DescribeProjectVersions`.

L'instantané est au format de sortie du manifeste SageMaker AI 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 additionnels.

```
"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 du champ `rekognition-custom-labels-evaluation-details` au sein de l'instantané du manifeste.
- is-true-positive... — La classification binaire de la prédiction basée sur la comparaison entre le score de confiance et le seuil minimum de l'étiquette.
- is-present-in-ground-truth — Vrai si la prédiction faite par le modèle est présente dans les informations de vérité du sol utilisées pour l'entraînement, sinon fausse. La valeur ne dépend pas du fait que le score de confiance dépasse ou non le seuil minimal calculé par le modèle.
- ground-truth-labeling-jobs— Une liste des champs de vérité de base figurant dans la ligne du manifeste qui sont utilisés pour la formation.

Pour plus d'informations sur le format du manifeste SageMaker AI Ground Truth, consultez Output.

Voici un exemple d'instantané de manifeste de tests affichant les métriques de classification d'images et de détection d'objets.

```
// For image classification
{
  "source-ref": "s3://amzn-s3-demo-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://amzn-s3-demo-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 (kit SDK)

Pour obtenir les résultats de l'entraînement, tu appelles DescribeProjectVersions. Pour obtenir un exemple de code, consultez Description d'un modèle (kit SDK).

L'emplacement des métriques est renvoyé dans la réponse `ProjectVersionDescription` de `DescribeProjectVersions`.

- `EvaluationResult` : emplacement du fichier récapitulatif.
- `TestingDataResult` : emplacement de l'instantané du manifeste d'évaluation utilisé pour les tests.

Le score F1 et l'emplacement du fichier récapitulatif 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 d'entrée `--output-config` que vous avez spécifié dans Entraînement d'un modèle (kit SDK).

> **Note**
>
> La quantité de temps d'entraînement, en secondes, pour laquelle vous êtes facturé est renvoyée dans `BillableTrainingTimeInSeconds`.

Pour plus d'informations sur les métriques renvoyées par Étiquettes personnalisées Amazon Rekognition, consultez Accès aux métriques d'évaluation d'Étiquettes personnalisées Amazon Rekognition (kit SDK).

Affichage de la matrice de confusion d'un modèle

Une matrice de confusion permet de voir les étiquettes que le modèle confond avec d'autres étiquettes du même modèle. En utilisant une matrice de confusion, vous pouvez concentrer vos améliorations sur le modèle.

Lors de l'évaluation du modèle, Étiquettes personnalisées Amazon Rekognition crée une matrice de confusion à l'aide des images de test afin de détecter les étiquettes mal identifiées (confuses). Étiquettes personnalisées Amazon Rekognition crée uniquement une matrice de confusion pour les modèles de classification. La matrice de classification est accessible depuis le fichier récapitulatif créé par Étiquettes personnalisées Amazon Rekognition lors de l'entraînement du modèle. Il n'est pas possible d'afficher la matrice de confusion dans la console Étiquettes personnalisées Amazon Rekognition.

Rubriques

- Utilisation d'une matrice de confusion
- Obtention de la matrice de confusion d'un modèle

Utilisation d'une matrice de confusion

Le tableau suivant constitue la matrice de confusion de l'exemple de projet de classification d'images de pièces (projet Rooms). Les en-têtes de colonne représentent les étiquettes (étiquettes de vérité sur le terrain) attribuées aux images de test. Les en-têtes de ligne représentent les étiquettes que

le modèle prédit pour les images de test. Chaque cellule représente le pourcentage de prédictions pour une étiquette (ligne) d'être l'étiquette de vérité sur le terrain (colonne). Par exemple, 67 % des prédictions sur les salles de bains ont été correctement étiquetées comme salles de bains. En revanche, 33 % des salles de bains ont été incorrectement étiquetées comme cuisines. Un modèle très performant possède des valeurs de cellules élevées lorsque l'étiquette prédite correspond à l'étiquette de vérité sur le terrain. Vous pouvez le voir grâce à la diagonale allant de la première à la dernière des étiquettes prédites et des étiquettes de vérité sur le terrain. Si la valeur d'une cellule est égale à 0, aucune prédiction n'a été faite pour l'étiquette prédite qui doit être l'étiquette de vérité sur le terrain de la cellule.

> **Note**
>
> Comme les modèles ne sont pas déterministes, les valeurs des cellules de la matrice de confusion obtenues lors de l'entraînement du projet Rooms peuvent différer de celles indiquées dans le tableau ci-dessous.

La matrice de confusion identifie les zones sur lesquelles il est nécessaire de se concentrer. Par exemple, la matrice de confusion montre que 50 % du temps, le modèle confond les placards et les chambres à coucher. Dans ce cas, vous devez ajouter de nouvelles images de placards et de chambres à coucher au jeu de données d'entraînement. Vérifiez également que les images existantes de placard et de chambre à coucher sont correctement étiquetées. Le modèle devrait ainsi mieux faire la différence entre les deux étiquettes. Pour ajouter de nouvelles images à un jeu de données, consultez Ajout d'autres images à un jeu de données.

Bien que la matrice de confusion soit utile, il importe de prendre en compte d'autres métriques. Par exemple, 100 % des prédictions ont correctement identifié l'étiquette floor_plan, ce qui constitue une excellente performance. Cependant, le jeu de données des tests ne contient que 2 images avec l'étiquette floor_plan. Il contient également 11 images avec l'étiquette living_space. Ce déséquilibre est également présent dans le jeu de données d'entraînement (13 images living_space et 2 images closet). Pour obtenir une évaluation plus précise, équilibrez les jeux de données d'entraînement et de test en ajoutant davantage d'images d'étiquettes sous-représentées (plans de niveau dans cet exemple). Pour obtenir le nombre d'images de test par étiquette, consultez Accès aux métriques d'évaluation (console).

Le tableau suivant est un exemple de matrice de confusion, comparant l'étiquette prévue (sur l'axe Y) à l'étiquette de vérité fondamentale :

Étiquette prévue	backyard (jardin arrière)	bathroom (salle de bains)	bedroom (chambre à coucher)	closet (placard)	entry_way (entrée)	floor_plan (plan d'étage)	front_yard (jardin avant)	kitchen (cuisine)	living_space (pièce de vie)	patio
backyard (jardin arrière)	75%	0 %	0 %	0 %	0 %	0 %	33 %	0 %	0 %	0 %
bathroom (salle de bains)	0 %	67 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
bedroom (chambre à coucher)	0 %	0 %	82 %	50%	0 %	0 %	0 %	0 %	9 %	0 %
closet (placard)	0 %	0 %	0 %	50%	0 %	0 %	0 %	0 %	0 %	0 %
entry_way (entrée)	0 %	0 %	0 %	0 %	33 %	0 %	0 %	0 %	0 %	0 %
floor_plan (plan d'étage)	0 %	0 %	0 %	0 %	0 %	100 %	0 %	0 %	0 %	0 %
front_yard (jardin avant)	25%	0 %	0 %	0 %	0 %	0 %	67 %	0 %	0 %	0 %
kitchen (cuisine)	0 %	33 %	0 %	0 %	0 %	0 %	0 %	88 %	0 %	0 %

Étiquette prévue	backyard (jardin arrière)	bathroom (salle de bains)	bedroom (chambre à coucher)	closet (placard)	entry_way (entrée)	floor_plan (plan d'étage)	front_yard (jardin avant)	kitchen (cuisine)	living_space (pièce de vie)	patio
living_space (pièce de vie)	0 %	0 %	18 %	0 %	67 %	0 %	0 %	12 %	91 %	33 %
patio	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	67 %

Obtention de la matrice de confusion d'un modèle

Le code suivant utilise les DescribeProjectVersionsopérations DescribeProjectset 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 (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez le code suivant pour afficher la matrice de confusion d'un modèle. Fournissez les arguments de ligne de commande suivants :

   - `project_name` : nom du projet que vous souhaitez utiliser. Vous pouvez obtenir le nom du projet sur la page des projets de la console Étiquettes personnalisées Amazon Rekognition.
   - `version_name` : version du modèle que vous souhaitez utiliser. Vous pouvez obtenir le nom de la version sur la page des informations détaillées du projet de la console Étiquettes personnalisées Amazon Rekognition.

```
# 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 l'entraînement

Le résumé des résultats de l'entraînement contient les métriques que vous pouvez utiliser pour évaluer votre modèle. Le fichier récapitulatif est également utilisé pour afficher les métriques dans la page de la console relative aux résultats de l'entraînement. 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, consultez Accès au fichier récapitulatif et à l'instantané du manifeste d'évaluation (kit SDK).

Fichier récapitulatif

Le format JSON suivant est le format du fichier récapitulatif.

EvaluationDetails (article 3)

Informations générales sur la tâche d'entraînement. Ces informations incluent l'ARN du projet auquel appartient le modèle (`ProjectVersionArn`), la date et l'heure de fin de l'entraînement, la version du modèle évalué (`EvaluationEndTimestamp`) et la liste des étiquettes détectées pendant l'entraînement (`Labels`). Figurent aussi le nombre d'images utilisées pour l'entraînement (`NumberOfTrainingImages`) et l'évaluation (`NumberOfTestingImages`).

AggregatedEvaluationResults (article 1)

Vous pouvez utiliser `AggregatedEvaluationResults` pour évaluer les performances globales du modèle entraîné lorsqu'il est utilisé avec le jeu de données des tests. Des métriques de cumul sont incluses pour les métriques `Precision`, `Recall` et `F1Score`. Pour la détection d'objets (emplacement de l'objet sur une image), les métriques `AverageRecall` (mAR) et `AveragePrecision` (mAP) sont renvoyées. Pour la classification (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 métriques incluses sont `Precision`, `Recall`, `F1Score` et `Threshold` (utilisées pour la classification).

Le format du fichier est le suivant : `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 Étiquettes personnalisées Amazon Rekognition

Les performances des modèles de machine learning dépendent largement de facteurs tels que la complexité et la variabilité des étiquettes personnalisées (objets et scènes spécifiques qui vous intéressent), la qualité et la capacité représentative du jeu de données d'entraînement que vous fournissez, ainsi que les infrastructures du modèle et les méthodes de machine learning utilisées pour entraîner le modèle.

Étiquettes personnalisées Amazon Rekognition simplifie le processus et aucune expertise en machine learning 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.

Données

En général, vous pouvez améliorer la qualité de votre modèle avec de plus grandes quantités de données de meilleure qualité. Utilisez des images d'entraînement qui montrent clairement l'objet ou la scène et qui ne sont pas encombrées d'objets inutiles. Pour les cadres de délimitation autour des objets, utilisez les images d'entraînement montrant l'objet entièrement visible et non masqué par d'autres objets.

Assurez-vous que les jeux de données d'entraînement et de test correspondent au type d'images sur lesquelles l'inférence va être exécutée. Pour les objets, tels que les logos, pour lesquels vous n'avez que quelques exemples d'entraînement, vous devez fournir des cadres de délimitation autour du logo dans les images de test. Ces images représentent ou illustrent les scénarios dans lesquels vous souhaitez localiser l'objet.

Pour ajouter de nouvelles images à un jeu de données d'entraînement ou de test, consultez Ajout d'autres images à un jeu de données.

Réduction des faux positifs (plus grande précision)

- Tout d'abord, vérifiez si l'augmentation du seuil supposé permet de conserver les prédictions correctes, tout en diminuant les faux positifs. À un moment donné, les gains diminuent 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 d'entrée `MinConfidence` telle que `DetectCustomLabels`. Pour plus d'informations, consultez Analyse d'une image avec un modèle entraîné.
- Il se peut qu'une ou plusieurs des étiquettes personnalisées qui vous intéressent (A) soient régulièrement confondues avec la même classe d'objets (mais pas avec une étiquette qui vous intéresse) (B). Pour vous aider, ajoutez B comme étiquette de classe d'objet à votre jeu de données d'entraînement (avec les images sur lesquelles vous avez obtenu un 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 Ajout d'autres images à un jeu de données.
- Vous constaterez peut-être que le modèle confond deux de vos étiquettes personnalisées (A et B) : il est prédit que l'image de test portant l'étiquette A portera l'étiquette B et inversement. Dans ce cas, vérifiez d'abord qu'il n'y a pas d'images mal étiquetées dans vos jeux de données d'entraînement et de test. Utilisez la galerie de jeux de données pour gérer les étiquettes attribuées à un jeu de données. Pour plus d'informations, consultez Gestion des étiquettes. De plus, l'ajout d'images d'entraînement liées à ce type de confusion aide un modèle réentraîné à mieux faire la distinction entre A et B. Pour ajouter des images à un jeu de données d'entraînement, consultez Ajout d'autres images à un jeu de données.

Réduction des faux négatifs (meilleur rappel)

- Utilisez une valeur inférieure pour le seuil supposé. Vous ne pouvez pas définir le seuil supposé d'une étiquette, mais vous pouvez obtenir le même résultat en spécifiant une valeur moindre pour le paramètre d'entrée `MinConfidence` telle que `DetectCustomLabels`. Pour plus d'informations, consultez Analyse d'une image avec un modèle entraîné.
- Utilisez de meilleurs exemples pour modéliser la diversité de l'objet et les images dans lesquelles il apparaît.

- Scindez l’étiquette en deux catégories plus faciles à apprendre. Par exemple, au lieu de bons et de mauvais biscuits, vous préférerez peut-être de bons biscuits, des biscuits brûlés ou des biscuits brisés 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 AWS SDK. La console inclut également des exemples d’opérations du kit SDK que vous pouvez utiliser.

Rubriques

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, démarrez 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
`DesiredInferenceUnits`	Le nombre d'unités d'inférence par rapport auxquelles la fonctionnalité Étiquettes personnalisées Amazon Rekognition est augmentée ou réduite.
`InServiceInferenceUnits`	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 Description d'un modèle (kit 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 DetectCustomLabelssont 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 à l'aide de la console ou à l'aide de 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 de plus amples informations, veuillez consulter 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 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'StartProjectVersionAPI 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 ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
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 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'StopProjectVersionAPI 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 ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
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. Unexepected 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 AWS région, 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 CloudTraillac.

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

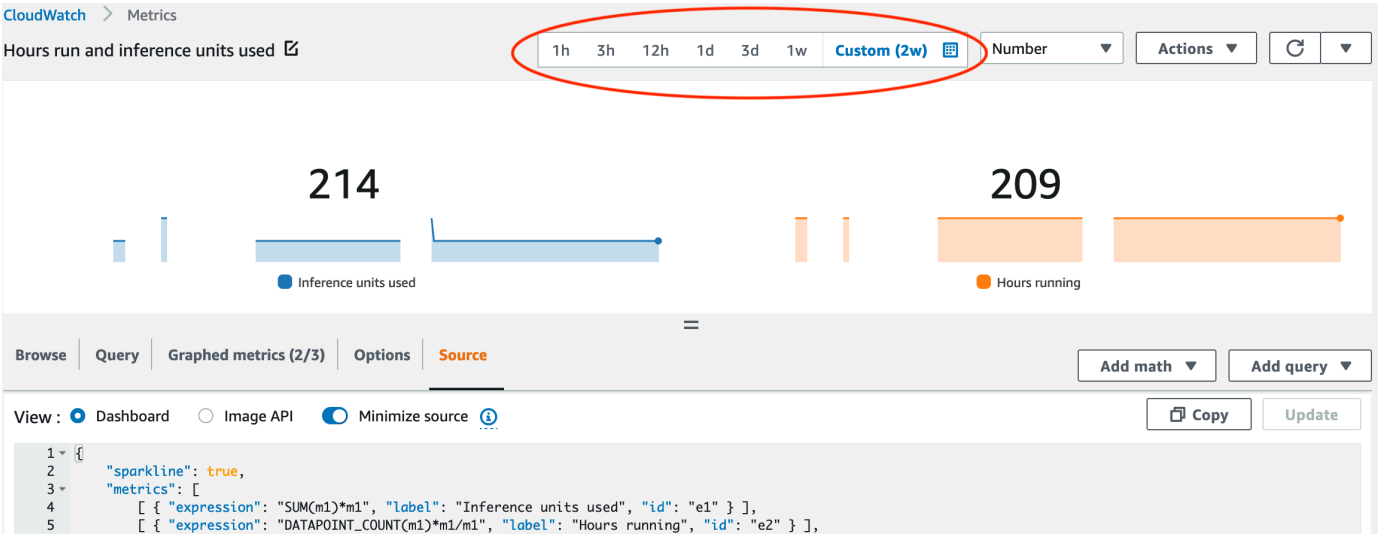
1. Connectez-vous à la CloudWatch console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans le volet de navigation de gauche, choisissez Toutes les métriques sous Métriques.
3. Dans le volet de contenu, choisissez l’onglet Source.
4. Assurez-vous que 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 AWS région 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. La capture d'écran de la console ci-dessous montre les unités d'inférence utilisées et les heures réparties sur des périodes, avec une durée personnalisée de 2 semaines définie, avec les valeurs les plus élevées de 214 unités d'inférence et 209 heures de fonctionnement.



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 d’évaluation du modèle.

Pour augmenter la précision de votre modèle, définissez une valeur plus élevée pour `MinConfidence`. Pour plus d'informations, consultez Réduction des faux positifs (plus grande précision).

Pour augmenter le rappel de votre modèle, utilisez une valeur inférieure pour `MinConfidence`. Pour plus d'informations, consultez Réduction des faux négatifs (meilleur rappel).

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 entraîné Étiquettes personnalisées Amazon Rekognition.

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 Étiquettes personnalisées Amazon Rekognition. 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 d'étiquettes personnalisées.

Pour plus d'informations sur la sécurisation de `DetectCustomLabels`, consultez Sécurisation DetectCustomLabels.

Pour détecter des étiquettes personnalisées (API)

1. Si vous ne l'avez pas déjà fait :

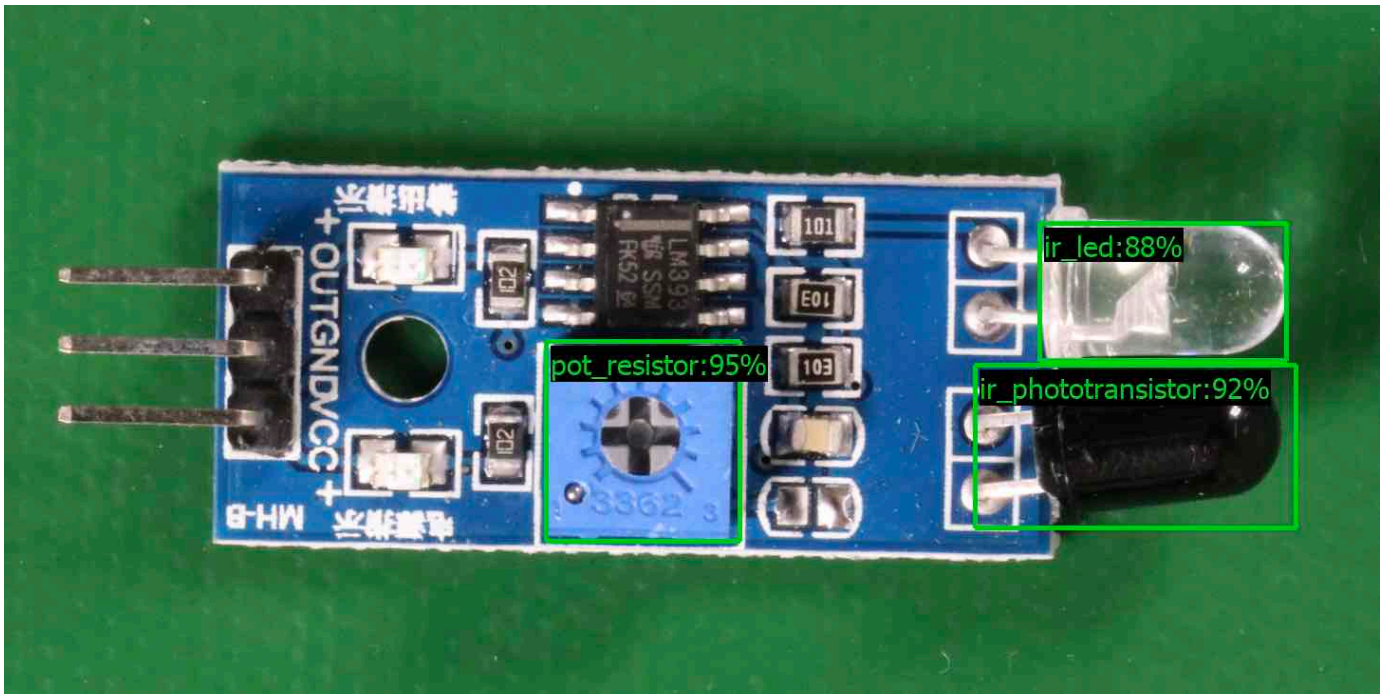
   a. Veillez à disposer des autorisations `DetectCustomLabels` et `AmazonS3ReadOnlyAccess`. Pour de plus amples informations, veuillez consulter Configuration des autorisations du kit SDK.

   b. Installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.

2. Entraînez et déployez votre modèle. Pour plus d'informations, consultez Création d'un modèle 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 Sécurisation DetectCustomLabels.
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. Les images suivantes contiennent des cadres de délimitation et des étiquettes pour un circuit imprimé doté d'un potentiomètre, d'un phototransistor infrarouge et de composants LED.



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`*`bucket name`*— Le compartiment Amazon S3 que vous avez utilisé à l'étape 4.

Notez que cet exemple suppose que votre version de Pillow est >= 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 Étiquettes personnalisées Amazon Rekognition

Cette section présente les ressources Amazon Rekognition Custom Labels que vous utilisez pour entraîner et gérer un modèle. Des informations générales sur l'utilisation du AWS SDK pour entraîner et utiliser un modèle sont également incluses.

Les étiquettes personnalisées Amazon Rekognition s'appuient sur trois ressources différentes pour détecter vos étiquettes personnalisées : les projets, les ensembles de données et les modèles.

- Projets : utilisés pour regrouper d'autres ressources telles que des ensembles de données, des versions de modèles et des évaluations de modèles.
- Ensembles de données : définit les images et les métadonnées associées à utiliser dans les modèles de formation et de test. Vous pouvez créer un ensemble de données en utilisant un fichier manifeste au format SageMaker AI ou en copiant un ensemble de données Amazon Rekognition Custom Labels existant.
- Modèles : modèle mathématique qui prédit réellement la présence d'objets, de scènes et de concepts dans les images en identifiant des modèles dans les images utilisées pour entraîner le modèle.

Rubriques

Gestion d’un projet Étiquettes personnalisées Amazon Rekognition

Dans Étiquettes personnalisées Amazon Rekognition, vous utilisez un projet pour gérer les modèles que vous créez pour un cas d’utilisation spécifique. Un projet gère les jeux de données, l’entraînement des modèles, les versions des modèles, l’évaluation des modèles et l’exécution des modèles de votre projet.

Rubriques

- Suppression d’un projet Étiquettes personnalisées Amazon Rekognition
- Description d’un projet (kit SDK)
- Création d’un projet avec AWS CloudFormation

Suppression d’un projet Étiquettes personnalisées Amazon Rekognition

Vous pouvez supprimer un projet à l'aide de la console Amazon Rekognition ou en appelant l'API. DeleteProject Pour supprimer un projet, vous devez d’abord supprimer chaque modèle associé. Il est impossible de rétablir un projet ou un modèle supprimé.

Rubriques

- Suppression d’un projet Étiquettes personnalisées Amazon Rekognition (console)
- Suppression d’un projet Étiquettes personnalisées Amazon Rekognition (kit SDK)

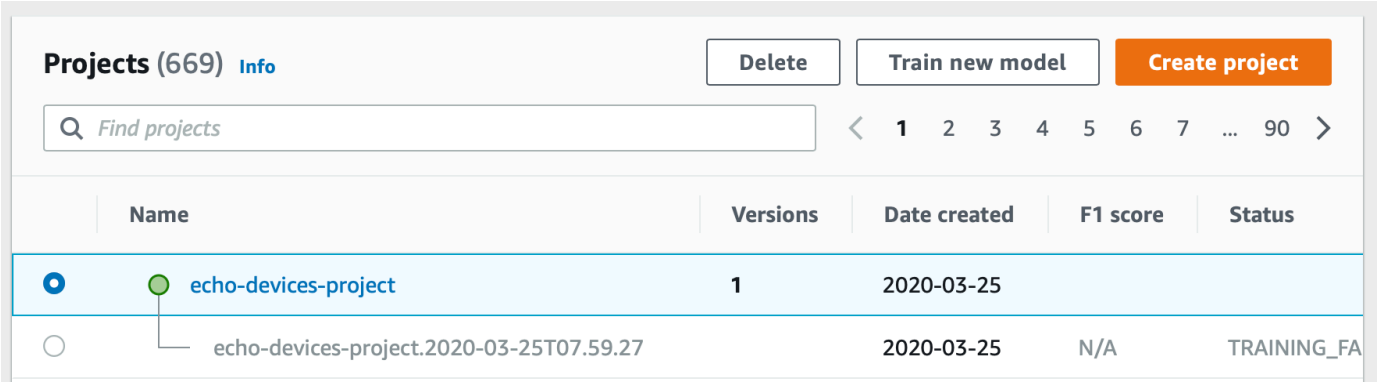
Suppression d’un projet Étiquettes personnalisées Amazon Rekognition (console)

Vous pouvez supprimer un projet depuis la page des projets ou supprimer un projet depuis la page détaillée d’un projet. La procédure suivante vous montre comment supprimer un projet en utilisant la page des projets.

La console Étiquettes personnalisées Amazon Rekognition supprime pour vous les modèles et les jeux de données associés 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 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’entraînement, attendez la fin de l’entraînement pour supprimer le projet.

Pour supprimer un projet (console)

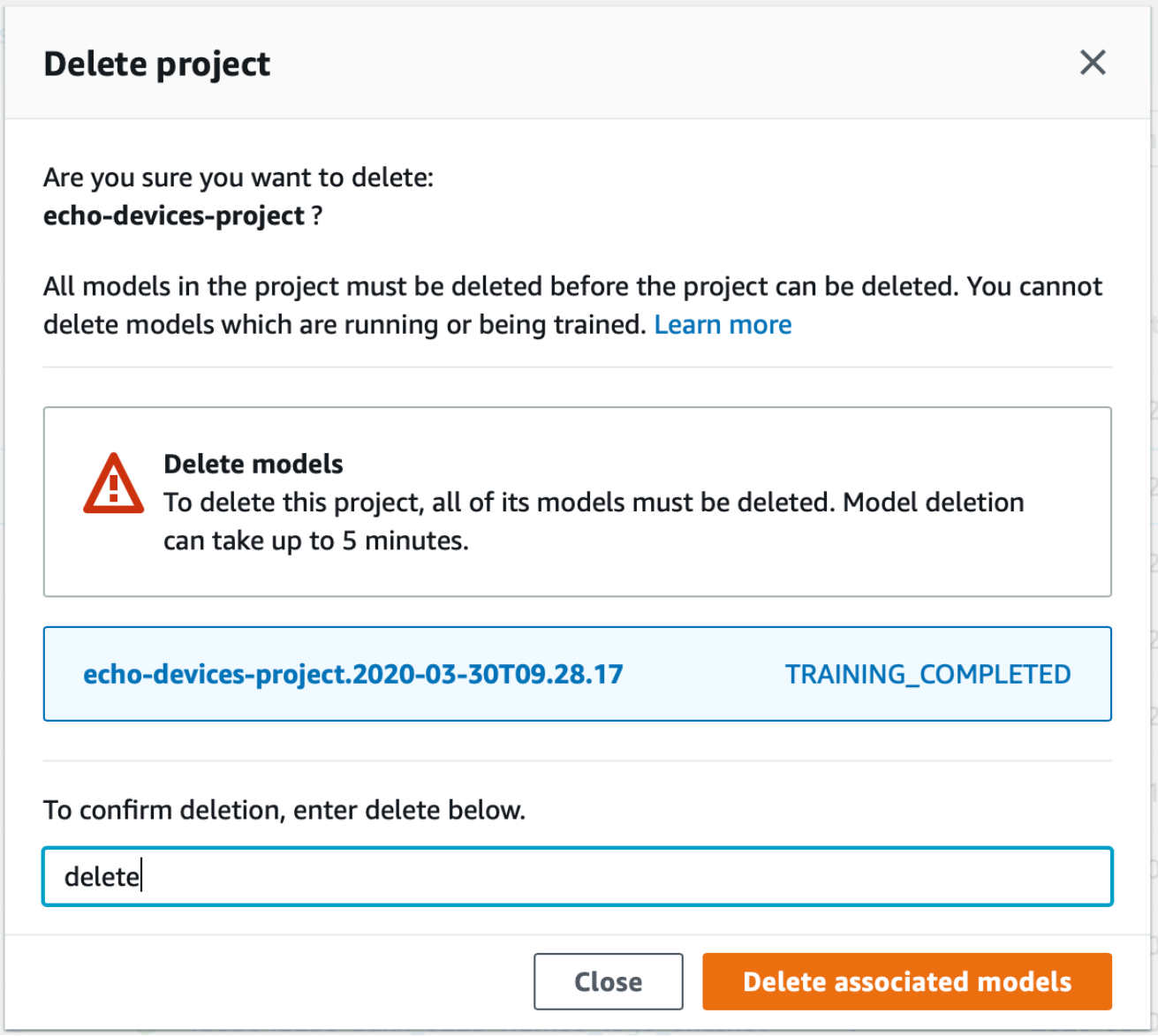
1. Ouvrez la console Amazon Rekognition à l'adresse. https://console.aws.amazon.com/rekognition/
2. Choisissez Utiliser Custom Labels.
3. Choisissez Démarrer.
4. Dans le volet de navigation de gauche, choisissez Projets.
5. Sur la page Projets, sélectionnez la case d’option pour le projet que vous souhaitez supprimer. La liste des projets affiche echo-devices-project, avec 1 version créée le 25/03/2020, et les options pour supprimer, entraîner un nouveau modèle ou créer un projet.



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 aucun modèle ou jeu de données n'est associé au projet :
   a. Entrez delete pour confirmer que vous souhaitez supprimer le ou les modèles et jeux de données.
   b. Choisissez Supprimer les modèles associés, Supprimer les jeux de données associés ou Supprimer les ensembles 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.

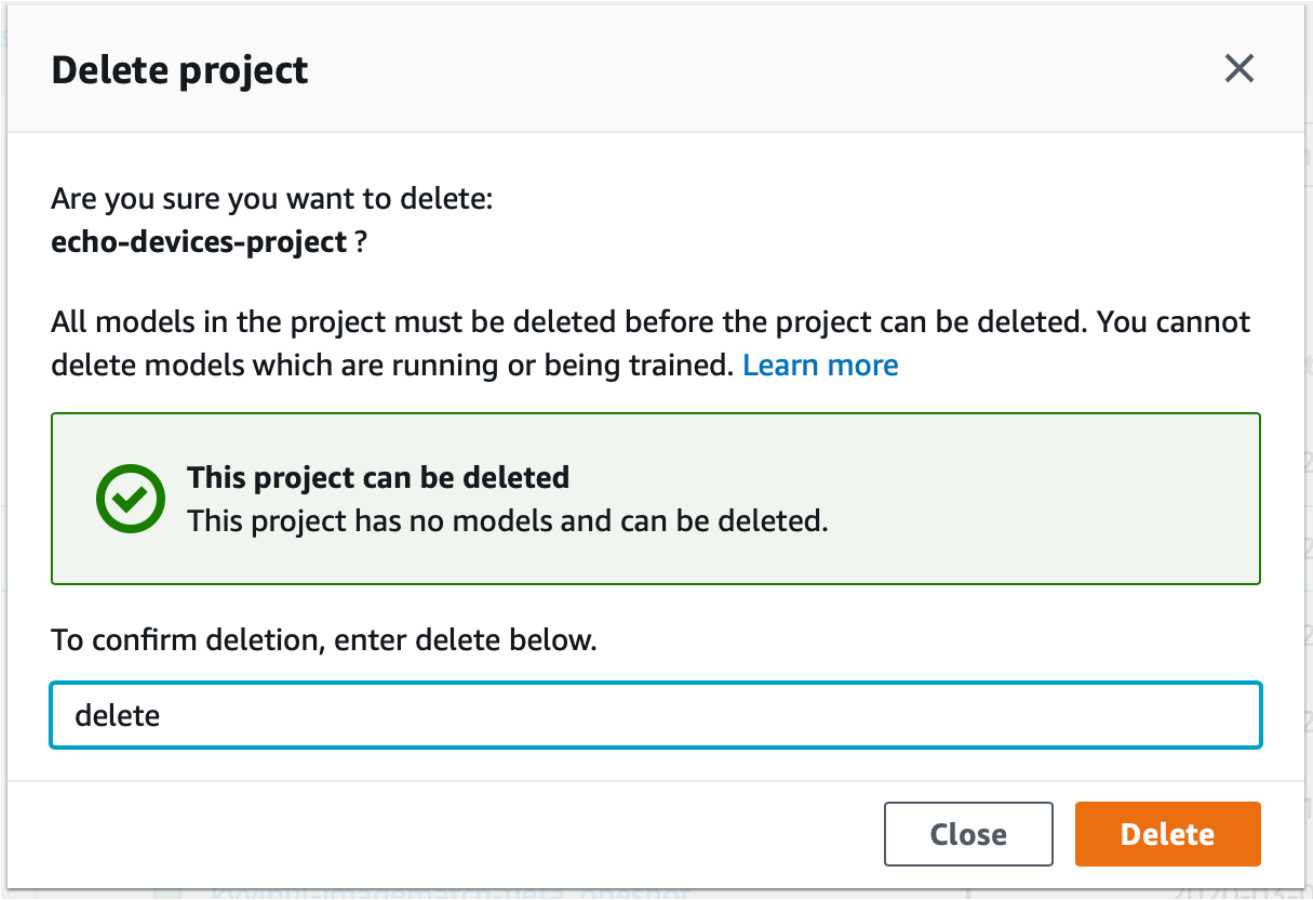
> **Note**
>
> La console ne peut pas supprimer les modèles en cours d'entraînement ou d'exécution. Réessayez après avoir arrêté tous les modèles en cours d'exécution répertoriés et attendez que ceux répertoriés comme des modèles d'entraînement soient terminés.
> Si vous choisissez de Fermer la boîte de dialogue pendant la suppression du modèle, les modèles sont quand même supprimés. Vous pouvez supprimer le projet par la suite en répétant cette procédure.

Le panneau de suppression d'un modèle vous donne des instructions explicites pour supprimer les modèles associés.



c. Entrez delete pour confirmer que vous voulez supprimer le projet.

d. Choisissez Supprimer pour supprimer le projet.



Suppression d’un projet Étiquettes personnalisées Amazon Rekognition (kit SDK)

Vous supprimez un projet Amazon Rekognition Custom Labels en DeleteProjectappelant et en fournissant le nom de ressource Amazon (ARN) du projet que vous souhaitez supprimer. Pour obtenir ARNs les projets sur votre AWS compte, appelez DescribeProjects. La réponse inclut un ensemble d'ProjectDescriptionobjets. L’ARN du projet est le champ `ProjectArn`. 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 supprimer un projet, vous devez commencer par supprimer tous les modèles et jeux de données dans le projet. Pour plus d’informations, consultez Suppression d’un modèle Étiquettes personnalisées Amazon Rekognition (kit SDK) et Suppression d’un jeu de données.

La suppression du projet peut prendre quelques instants. Pendant ce temps, le statut du projet est `DELETING`. Le projet est supprimé si un appel ultérieur DescribeProjectsn'inclut pas le projet que vous avez supprimé.

Pour supprimer un projet (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez le code suivant pour supprimer un projet.

   AWS CLI

   Remplacez la valeur de `project-arn` 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 occurences.
    :param input_string: The string you want to search
    : n: the occurence 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);
        }
    }

}
```

Description d’un projet (kit SDK)

Vous pouvez utiliser l’API `DescribeProjects` pour obtenir des informations sur vos projets.

Pour décrire un projet (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
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 : le 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 avec AWS CloudFormation

Amazon Rekognition Custom Labels est AWS CloudFormation intégré à un service qui vous aide à modéliser et à AWS configurer vos ressources afin que vous puissiez passer moins de temps à créer et à gérer vos ressources et votre infrastructure. Vous créez un modèle qui décrit toutes les AWS ressources que vous souhaitez, et vous vous AWS CloudFormation occupez de leur provisionnement et de leur configuration.

Vous pouvez l'utiliser AWS CloudFormation pour approvisionner et configurer les projets Amazon Rekognition Custom Labels.

Lorsque vous l'utilisez AWS CloudFormation, vous pouvez réutiliser votre modèle pour configurer vos projets Amazon Rekognition Custom Labels de manière cohérente et répétée. Décrivez simplement vos projets une seule fois, puis fournissez les mêmes projets à plusieurs reprises dans plusieurs AWS comptes et régions.

Étiquettes et modèles personnalisés Amazon Rekognition AWS CloudFormation

Pour allouer et configurer des projets pour Étiquettes personnalisées Amazon Rekognition et les services connexes, vous devez maîtriser les modèles AWS CloudFormation. Les modèles sont des fichiers texte formatés en JSON ou YAML. Ces modèles décrivent les ressources que vous souhaitez mettre à disposition dans vos AWS CloudFormation piles. Si vous n'êtes pas familiarisé avec JSON ou YAML, vous pouvez utiliser AWS CloudFormation Designer pour vous aider à démarrer avec les AWS CloudFormation modèles. Pour plus d’informations, consultez Qu’est-ce que AWS CloudFormation Designer ? dans le AWS CloudFormation Guide de l’utilisateur.

Pour obtenir des informations de référence sur des projets Étiquettes personnalisées Amazon Rekognition, y compris des exemples de modèles JSON et YAML, consultez Référence du type de ressource Rekognition.

En savoir plus sur AWS CloudFormation

Pour en savoir plus AWS CloudFormation, consultez les ressources suivantes :

- AWS CloudFormation
- AWS CloudFormation Guide de l'utilisateur

- AWS CloudFormation API Reference
- AWS CloudFormation Guide de l'utilisateur de l'interface de ligne de commande

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 expliquent comment gérer un ensemble de données à l'aide de la console Amazon Rekognition Custom Labels et du SDK. AWS

Rubriques

- Ajout d’un jeu de données à un projet
- Ajout d’autres images à un jeu de données
- Création d’un jeu de données à partir d’un jeu de données existant (kit SDK)
- Description d’un jeu de données (kit SDK)
- Liste des entrées d’un jeu de données (kit SDK)
- Distribution d’un jeu de données d’entraînement (kit SDK)
- Suppression d’un jeu de données

Ajout d’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 d’abord le jeu de données existant. Pour plus d’informations, consultez Suppression d’un jeu de données. Ajoutez ensuite le nouveau jeu de données.

Rubriques

- Ajout d’un jeu de données à un projet (console)
- Ajout d’un jeu de données à un projet (kit SDK)

Ajout d’un jeu de données à un projet (console)

Vous pouvez ajouter un jeu de données d’entraînement ou de test à un projet à l’aide de la console Étiquettes personnalisées Amazon Rekognition.

Pour ajouter un jeu de données à un 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, choisissez Projets. La vue Projets s'affiche.
4. Choisissez le projet auquel vous souhaitez ajouter un jeu de données.
5. Dans le volet de navigation de gauche, sous le nom du projet, choisissez Ensembles 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, consultez 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 jeu de données existant (entraînement ou test), la page des détails du projet s'affiche. Procédez comme suit :

   a. Sur la page des détails du projet, choisissez 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, consultez 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, consultez Ajout d'autres images (console).
9. Ajoutez des étiquettes à votre jeu de données. Pour plus d'informations, consultez 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 délimitation, consultez Étiquetage des objets à l'aide de cadres de délimitation. Pour plus d'informations, consultez Utilisation des jeux de données.

Ajout d'un jeu de données à un projet (kit SDK)

Vous pouvez ajouter un jeu de données d'entraînement ou de test à un projet existant des façons suivantes :

- Créez un jeu de données à l'aide d'un fichier manifeste. Pour plus d'informations, consultez Création d'un ensemble de données à l'aide d'un fichier manifeste (SDK) SageMaker AI Ground Truth.
- Créez un jeu de données vide et remplissez-le ensuite. L'exemple suivant montre comment créer un jeu de données vide. Pour ajouter des entrées après avoir créé un jeu de données vide, consultez Ajout d'autres images à un jeu de données.

Pour ajouter un jeu de données à un projet (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez les exemples suivants pour ajouter des lignes JSON à un jeu 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 jeu 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 jeu de données. Fournissez l'options de ligne de commande suivantes :

   - `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 (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 jeu de données. Fournissez l'options de ligne de commande suivantes :

- `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 (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 datast 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, consultez Ajout d'autres images (kit SDK).

Ajout d'autres images à un jeu de données

Vous pouvez ajouter d'autres images à vos jeux de données en utilisant la console Étiquettes personnalisées Amazon Rekognition ou en appelant l'API `UpdateDatasetEntries`.

Rubriques

- Ajout d'autres images (console)
- Ajout d'autres images (kit SDK)

Ajout d'autres images (console)

Lorsque vous utilisez la console Étiquettes personnalisées Amazon Rekognition, 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/
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, choisissez Projets. La vue Projets s'affiche.
4. Choisissez le projet que vous voulez utiliser.

5. Dans le volet 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 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. Vous pouvez charger jusqu'à 30 images à la fois.
8. Choisissez Charger des images.
9. Sélectionnez Enregistrer les modifications.
10. Étiquetez les images. Pour plus d'informations, consultez Étiquetage des images.

Ajout d'autres images (kit SDK)

`UpdateDatasetEntries` met à jour ou ajoute des lignes JSON à un fichier manifeste. Vous transmettez les lignes JSON sous forme d'objet de données codé byte64 dans le champ `GroundTruth`. Si vous utilisez un AWS SDK pour appeler`UpdateDatasetEntries`, le SDK code les données pour vous. Chaque ligne JSON contient des informations sur une seule image (étiquettes attribuées, informations sur les cadres de délimitation, par exemple). 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, consultez Création d'un fichier manifeste.

Utilisez le champ `source-ref` comme clé pour identifier les images que vous souhaitez mettre à jour. Si le jeu de données ne contient pas de valeur de champ `source-ref` correspondante, la ligne JSON est ajoutée en tant que nouvelle image.

Pour ajouter d'autres images à un jeu de données (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez les exemples suivants pour ajouter des lignes JSON à un jeu 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},{\"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 : l'ARN du jeu de données que vous souhaitez mettre à jour.
   - updates_file : le 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 yuo 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 : l'ARN du jeu de données que vous souhaitez mettre à jour.
- update_file : le 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
*/
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 partir d'un jeu de données existant (kit SDK)

La procédure suivante explique comment créer un ensemble de données à partir d'un ensemble de données existant à l'aide de l'CreateDatasetopération.

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez l'exemple de code suivant pour créer un jeu de données en copiant un autre jeu de données.

   AWS CLI

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

   - `project_arn` : l'ARN du projet que vous souhaitez ajouter au jeu de données.
   - `dataset_type` : le type de jeu de données (`TRAIN` ou `TEST`) que vous souhaitez créer dans le projet.
   - `dataset_arn` : 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 jeu de données à partir d'un jeu de données existant et affiche son ARN.

   Pour exécuter le programme, fournissez les arguments de ligne de commande suivants :

   - `project_arn` : l'ARN du projet que vous souhaitez utiliser.
   - `dataset_type` : le type de jeu de données du projet que vous souhaitez créer (`train` ou `test`).

- `dataset_arn` : l'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 jeu de données à partir d'un jeu de données existant et affiche son ARN.

Pour exécuter le programme, fournissez les arguments de ligne de commande suivants :

- `project_arn` : l'ARN du projet que vous souhaitez utiliser.
- `dataset_type` : le type de jeu de données du projet que vous souhaitez créer (`train` ou `test`).
- `dataset_arn` : l'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);
        }

    }
```

```
}
```

Description d'un jeu de données (kit SDK)

Vous pouvez utiliser l'API `DescribeDataset` pour obtenir des informations sur un jeu de données.

Pour décrire un jeu de données (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez l'exemple de code suivant pour décrire un jeu de données.

   AWS CLI

   Remplacez la valeur de `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 : l'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['StatusMessageCode']}")
        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 : l'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);
        }

    }

}
```

Liste des entrées d'un jeu de données (kit SDK)

Vous pouvez utiliser l'API `ListDatasetEntries` pour répertorier les lignes JSON pour chaque image d'un jeu de données. Pour plus d'informations, consultez Création d'un fichier manifeste.

Pour répertorier les entrées d'un jeu de données (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez l'exemple de code suivant pour répertorier les entrées d'un jeu de données.

   AWS CLI

   Remplacez la valeur de `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 : l'ARN du jeu de données que vous souhaitez répertorier.
- show_errors_only : spécifiez `true` si vous souhaitez voir uniquement les erreurs, ou `false` dans le cas contraire.

```
# 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 : l'ARN du jeu de données que vous souhaitez répertorier.
- show_errors_only : spécifiez `true` si vous souhaitez voir uniquement les erreurs, ou `false` dans le cas contraire.

```
/*
   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 (kit SDK)

La fonctionnalité Étiquettes personnalisées Amazon Rekognition nécessite un jeu de données d’entraînement et un jeu de données de test pour entraîner votre modèle.

Si vous utilisez l'API, vous pouvez utiliser l'DistributeDatasetEntriesAPI pour distribuer 20 % de l'ensemble de données d'apprentissage dans un ensemble de données de test vide. La distribution du jeu de données d’entraînement peut être utile si vous ne disposez que d’un seul fichier manifeste.

Utilisez le fichier manifeste unique pour créer votre jeu de données d'entraînement. Créez ensuite un jeu de données de test vide et utilisez `DistributeDatasetEntries` pour remplir le jeu de données de test.

> **Note**
>
> Si vous utilisez la console Étiquettes personnalisées Amazon Rekognition et que vous commencez par un seul projet de jeu de données, la fonctionnalité Étiquettes personnalisées Amazon Rekognition divise (distribue) le jeu de données d'entraînement, pendant l'entraînement, pour créer un jeu de données de test. 20 % des entrées du jeu de données d'entraînement sont déplacées vers le jeu de données de test.

Pour distribuer un jeu de données d'entraînement (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Créez un projet. Pour plus d'informations, consultez Création d'un projet Étiquettes personnalisées Amazon Rekognition (kit SDK).
3. Créez votre jeu de données d'entraînement. Pour en savoir plus sur les jeux de données, consultez Création de jeux de données d'entraînement et de test.
4. Créez un jeu de données de test vide.
5. Utilisez l'exemple de code suivant pour distribuer 20 % des entrées du jeu de données d'entraînement dans le jeu de données de test. Vous pouvez obtenir les Amazon Resource Names (ARN) pour les ensembles de données d'un projet en appelant DescribeProjects. Pour obtenir un exemple de code, consultez Description d'un projet (kit SDK).

   AWS CLI

   Remplacez la valeur de `training_dataset-arn` et `test_dataset_arn` par les ARN 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 : l'ARN du jeu de données d'entraînement à partir duquel vous distribuez les entrées.
- test_dataset_arn : l'ARN du jeu de données de test vers lequel vous distribuez les 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 : l'ARN du jeu de données d'entraînement à partir duquel vous distribuez les entrées.
- test_dataset_arn : l'ARN du jeu de données de test vers lequel vous distribuez les 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);
        }

    }

}
```

Suppression d’un jeu de données

Vous pouvez supprimer les jeux de données d’entraînement et de test d’un projet.

Rubriques

- Suppression d’un jeu de données (console)
- Suppression d’un jeu de données Étiquettes personnalisées Amazon Rekognition (kit SDK)

Suppression d’un jeu de données (console)

Utilisez la procédure suivante pour supprimer un jeu de données. Ensuite, si le projet possède un jeu de données restant (entraînement ou test), 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 créer un nouveau jeu de données d'entraînement pour le projet avant de pouvoir entraîner un modèle. Pour plus d'informations, consultez 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 afin de créer un nouveau jeu 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 maintenir la qualité, nous vous recommandons de créer un nouveau jeu de données de test avant d'entraîner un modèle. Pour plus d'informations, consultez Ajout d'un jeu de données à un projet.

Supprimer un jeu de données

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, choisissez Projets. La vue Projets s'affiche.
4. Choisissez le projet qui contient le jeu de données que vous souhaitez supprimer.
5. Dans le volet 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 un jeu de données d'entraînement.
8. Pour supprimer le jeu de données de test, choisissez Supprimer un jeu de données de test.
9. Dans la boîte de dialogue Supprimer un jeu de données d'entraînement ou de test, entrez delete pour confirmer que vous souhaitez supprimer le jeu de données.
10. Choisissez Supprimer un jeu de données d'entraînement ou de test pour supprimer le jeu de données.

Suppression d'un jeu de données Étiquettes personnalisées Amazon Rekognition (kit SDK)

Vous supprimez un ensemble de données Amazon Rekognition Custom Labels en DeleteDatasetappelant et en fournissant le nom de ressource Amazon (ARN) du jeu de données que vous souhaitez supprimer. Pour obtenir les ensembles ARNs de données de formation et de test

d'un projet, appelez DescribeProjects. La réponse inclut un ensemble d'ProjectDescriptionobjets. Les types d'ensembles de données ARNs (`DatasetArn`) et de jeux de données (`DatasetType`) figurent dans la `Datasets` liste.

Si vous supprimez le jeu de données d'entraînement, vous devez créer un nouveau jeu de données d'entraînement pour le projet avant de pouvoir entraîner un modèle. Si vous supprimez le jeu de données de test, vous devez créer un nouveau jeu de données de test avant de pouvoir entraîner le modèle. Pour plus d'informations, consultez Ajout d'un jeu de données à un projet (kit SDK).

Pour supprimer un jeu de données (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez le code suivant pour supprimer un jeu de données.

   AWS CLI

   Remplacez la valeur de `dataset-arn` par 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 : l'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 : l'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 Étiquettes personnalisées Amazon Rekognition

Un modèle Étiquettes personnalisées Amazon Rekognition est un modèle mathématique qui prédit la présence d’objets, de scènes et de concepts dans de nouvelles images. Pour ce faire, il recherche des modèles dans les images utilisées pour entraîner le modèle. Cette section explique comment entraîner un modèle, évaluer ses performances et apporter des améliorations. Elle explique 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 Étiquettes personnalisées Amazon Rekognition
- Balisage d’un modèle
- Description d’un modèle (kit SDK)
- Copie d’un modèle Étiquettes personnalisées Amazon Rekognition (kit SDK)

Suppression d’un modèle Étiquettes personnalisées Amazon Rekognition

Vous pouvez supprimer un modèle à l'aide de la console Amazon Rekognition Custom Labels ou à l'aide de l'API. DeleteProjectVersion Vous ne pouvez pas supprimer un modèle s’il fonctionne ou s’il est en cours d’entraînement. Pour arrêter un modèle en cours d'exécution, utilisez l'StopProjectVersionAPI. Pour de plus amples 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 Étiquettes personnalisées Amazon Rekognition (console)
- Suppression d’un modèle Étiquettes personnalisées Amazon Rekognition (kit SDK)

Suppression d’un modèle Étiquettes personnalisées Amazon Rekognition (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 de détails d’un modèle.

Pour supprimer 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. Choisissez le projet qui contient le modèle que vous souhaitez supprimer. La page de détails du projet s’ouvre.
6. Dans la section Modèles, sélectionnez les modèles que vous voulez supprimer.

   > **Note**
   >
   > Si le modèle ne peut pas être sélectionné, il est en cours d’exécution ou d’entraînement et ne peut pas être supprimé. Vérifiez le champ Statut 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 Supprimer pour supprimer le modèle. La suppression du modèle peut prendre un certain temps.

   > **Note**
   >
   > Si vous choisissez de Fermer la boîte de dialogue pendant la suppression du modèle, les modèles sont quand même supprimés.

Suppression d’un modèle Étiquettes personnalisées Amazon Rekognition (kit SDK)

Vous supprimez un modèle d'étiquettes personnalisées Amazon Rekognition en DeleteProjectVersionappelant et en fournissant le nom de ressource Amazon (ARN) du modèle

que vous souhaitez supprimer. Vous pouvez obtenir l'ARN du modèle depuis la section Utiliser votre modèle de la page de détails du modèle dans la console Étiquettes personnalisées Amazon Rekognition. Vous pouvez également appeler DescribeProjectVersionset fournir les documents suivants.

- ARN du projet (`ProjectArn`) auquel le modèle est associé.
- Nom de la version (`VersionNames`) du modèle.

L'ARN du modèle est le `ProjectVersionArn` champ de l'ProjectVersionDescriptionobjet, à partir de la `DescribeProjectVersions` réponse.

Vous ne pouvez pas supprimer un modèle s'il est en cours d'exécution ou d'entraînement. Pour déterminer si le modèle est en cours d'exécution ou en cours d'entraînement, appelez DescribeProjectVersionset vérifiez le `Status` champ de l'ProjectVersionDescriptionobjet du modèle. Pour arrêter un modèle en cours d'exécution, utilisez l'StopProjectVersionAPI. Pour de plus amples informations, veuillez consulter Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition (kit SDK). Vous devez attendre la fin de l'entraînement d'un modèle pour pouvoir le supprimer.

Pour supprimer un modèle (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez le code suivant pour supprimer un modèle.

   AWS CLI

   Remplacez la valeur de `project-version-arn` 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` — ARN du projet qui contient le modèle que vous souhaitez supprimer.
   - `model_arn` — 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 occurences.
    :param input_string: The string you want to search
    : n: the occurence 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` — ARN du projet qui contient le modèle que vous souhaitez supprimer.
- `model_arn` — 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);
        }

    }

}
```

Balisage d’un modèle

Vous pouvez identifier, organiser, rechercher et filtrer les collections Amazon Rekognition à l’aide de balises. Chaque balise est une étiquette composée d’une clé définie par l’utilisateur et d’une valeur. Par exemple, pour déterminer la facturation de vos modèles, balisez-les avec une clé `Cost center` et ajoutez le numéro de centre de coûts approprié sous forme de valeur. Pour plus d’informations, consultez Balisage des ressources AWS.

Utilisez des balises pour :

- Suivre la facturation d'un modèle à l'aide des balises de répartition des coûts. Pour plus d'informations, consultez Utilisation des balises de répartition des coûts.
- Contrôler l'accès à un modèle à l'aide de Gestion des identités et des accès (IAM). Pour plus d'informations, consultez Contrôle de l'accès aux ressources AWS à l'aide de balises de ressources.
- Automatiser la gestion des modèles. Par exemple, vous pouvez exécuter des scripts automatisés de départ ou d'arrêt qui désactivent les modèles de développement en dehors des heures ouvrables afin de réduire les coûts. Pour de plus amples informations, veuillez consulter Exécution d'un modèle Étiquettes personnalisées Amazon Rekognition entraîné.

Vous pouvez étiqueter les modèles à l'aide de la console Amazon Rekognition ou à l'aide du. AWS SDKs

Rubriques

- Balisage de modèles (console)
- Affichage des balises de modèle
- Balisage de modèles (kit SDK)

Balisage de modèles (console)

Vous pouvez utiliser la console Rekognition pour ajouter des balises aux modèles, afficher les balises jointes 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 également ajouter des balises à un nouveau modèle lorsqu'il est entraîné. Pour plus d'informations, consultez Entraînement d'un modèle Étiquettes personnalisées Amazon Rekognition.

Pour ajouter ou supprimer des balises à un modèle existant à l'aide de la console

1. Ouvrez la console Amazon Rekognition à l'adresse. https://console.aws.amazon.com/rekognition/
2. Choisissez Démarrer.
3. Dans le panneau de navigation, choisissez Projects (Projets).

4. Sur la page des ressources Projets, choisissez le projet qui contient le modèle entraîné que vous souhaitez baliser.
5. Dans le volet de navigation, sous le projet que vous avez choisi précédemment, choisissez Modèles.
6. Dans la section Modèles, choisissez le modèle auquel vous souhaitez ajouter une balise.
7. Sur la page de détails du modèle, choisissez l'onglet Balises.
8. Dans la section Tags (Balises) choisissez Manage tags (Gérer les balises).
9. Sur la page Gérer les balises, choisissez Ajouter une nouvelle balise.
10. Saisissez une clé et une valeur.
    a. Pour Clé, saisissez le nom de 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 que vous souhaitez supprimer. Si vous supprimez une balise précédemment enregistrée, elle sera supprimée lorsque vous enregistrerez 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 jointes à un modèle.

Pour afficher les balises jointes à tous les modèles d'un projet, vous devez utiliser le kit AWS SDK. Pour plus d'informations, consultez Établissement de la liste des balises de modèle.

Pour afficher les balises jointes à un modèle

1. Ouvrez la console Amazon Rekognition à l'adresse. https://console.aws.amazon.com/rekognition/
2. Choisissez Démarrer.
3. Dans le panneau de navigation, choisissez Projects (Projets).
4. Sur la page des ressources Projets, choisissez le projet qui contient le modèle entraîné que vous souhaitez afficher.
5. Dans le volet de navigation, sous le projet que vous avez choisi précédemment, choisissez Modèles.

6. Dans la section Modèles, choisissez le modèle dont vous souhaitez afficher la balise.
7. Sur la page de détails du modèle, choisissez l'onglet Balises. Les balises sont affichées dans la section Balises.

Balisage de modèles (kit SDK)

Vous pouvez utiliser le AWS SDK pour :

- Ajouter des balises à un nouveau modèle
- Ajouter des balises à un modèle existant
- Répertorier les balises jointes à un modèle
- Supprimer des balises d'un modèle

Les balises des AWS CLI exemples suivants sont au format suivant.

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

Vous pouvez également utiliser ce format.

```
--tags key1=value1,key2=value2
```

Si vous n'avez pas installé le AWS CLI, consultezÉtape 4 : Configurez le AWS CLI et AWS SDKs.

Ajout de balises à un nouveau modèle

Vous pouvez ajouter des balises à un modèle lorsque vous le créez à l'aide de cette CreateProjectVersionopération. Spécifiez une ou plusieurs balises dans le paramètre d'entrée du tableau `Tags`.

```
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 la formation d'un modèle, consultez Entraînement d'un modèle (kit SDK).

Ajout de balises à un modèle existant

Pour ajouter une ou plusieurs balises à un modèle existant, utilisez l'TagResourceopération. Spécifiez l'Amazon Resource Name (ARN) (`ResourceArn`) et les balises (`Tags`) du modèle que vous voulez 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 de modèle

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

```
aws rekognition list-tags-for-resource --resource-arn resource-arn \
  --profile custom-labels-access
```

La sortie affiche la liste des balises jointes au modèle.

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

Pour voir quels modèles d'un projet possèdent une balise spécifique, appelez `DescribeProjectVersions` pour obtenir la liste des modèles. Appelez ensuite `ListTagsForResource` pour chaque modèle dans la réponse de `DescribeProjectVersions`. Examinez la réponse de `ListTagsForResource` pour vérifier si la balise 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 *key* et *value* par le nom de clé et la valeur de clé que vous souhaitez rechercher.

```
python find_tag.py key value
```

Suppression des balises d'un modèle

Pour supprimer une ou plusieurs balises d'un modèle, utilisez l'UntagResourceopération. 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 également spécifier `tag-keys` dans ce format.

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

Description d'un modèle (kit SDK)

Vous pouvez utiliser l'API `DescribeProjectVersions` pour obtenir des informations sur la version d'un modèle. Si vous ne spécifiez pas de `VersionName`, `DescribeProjectVersions` renvoie les descriptions de toutes les versions de modèle du projet.

Pour décrire un modèle (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez l'exemple de code suivant pour décrire une version d'un modèle.

   AWS CLI

   Remplacez la valeur de `project-arn` par l'ARN du projet que vous souhaitez décrire. Remplacez la valeur de `version-name` 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 — ARN du modèle que vous souhaitez 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 prject 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 — ARN du modèle que vous souhaitez 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 Étiquettes personnalisées Amazon Rekognition (kit SDK)

Vous pouvez utiliser cette CopyProjectVersionopération pour copier une version du modèle Amazon Rekognition Custom Labels depuis un projet Amazon Rekognition Custom Labels source vers un projet de destination. Le projet de destination peut se trouver dans un autre AWS compte 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 avec le jeu de données source. L’utilisation de l’opération `CopyProjectVersion` présente les avantages suivants.

- Le comportement du modèle est cohérent. L’entraînement de modèle n’est pas déterministe et il n’est pas garanti que deux modèles entraînés avec le même jeu de données fassent les mêmes prédictions. La copie de modèle avec `CopyProjectVersion` permet de s’assurer que le comportement du modèle copié est cohérent avec le modèle source et vous n’avez pas besoin de retester le modèle.
- L’entraînement de modèle n’est pas obligatoire. Cela vous permet d’économiser de l’argent, car vous êtes facturé pour chaque entraînement réussi d’un modèle.

Pour copier un modèle vers un autre AWS compte, vous devez disposer d'un projet Amazon Rekognition Custom Labels dans le compte de destination. AWS Pour en savoir plus sur la création d’un projet, consultez Création d’un projet. Assurez-vous de créer le projet dans le AWS compte de destination.

Une politique de projet est une politique 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 entre comptes si vous souhaitez mieux contrôler ces ressources.

Vous associez la politique du projet au projet source en appelant l'PutProjectPolicyopération.

Vous ne pouvez pas l'utiliser `CopyProjectVersion` pour copier un modèle dans un projet d'une autre AWS région. Vous ne pouvez pas non plus copier un modèle à l'aide de la console Étiquettes personnalisées Amazon Rekognition. Dans ces cas, vous pouvez entraîner le modèle dans le projet de destination avec les jeux de données utilisés pour entraîner le modèle source. Pour plus d'informations, consultez Entraînement d'un modèle Étiquettes personnalisées Amazon Rekognition.

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'opération `CopyProjectVersion`.

Pour supprimer une politique de projet d'un projet, appelez DeleteProjectPolicy. Pour obtenir une 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 (kit SDK)
- Copie d'un modèle (kit SDK)
- Répertorier les politiques du projet (kit SDK)
- Suppression d'une politique de projet (kit SDK)

Création d'un document de politique de projet

Étiquettes personnalisées Amazon Rekognition utilise une politique basée sur les ressources, connue sous le nom de politique de projet, pour gérer les autorisations de copie pour une version de modèle. Une politique de projet est un document au format JSON.

Une politique de projet autorise ou refuse au principal l'autorisation de copier une version de modèle d'un projet source vers un projet de destination. Vous avez besoin d'une politique de projet si le projet de destination se trouve dans un autre AWS compte. C'est également vrai si le projet de destination se trouve dans le même compte AWS que le projet source et que vous souhaitez restreindre l'accès à des versions de modèles spécifiques. Par exemple, vous souhaiterez peut-être refuser les autorisations de copie pour un rôle IAM spécifique au sein d'un AWS compte.

L'exemple suivant permet au `arn:aws:iam::111111111111:role/Admin` principal 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`, `Resource`, `Principal`, et `Effect` sont des champs obligatoires dans un document de politique de projet.
> La seule `action` prise en charge est `rekognition:CopyProjectVersion`.
> `NotAction`, `NotResource`, et `NotPrincipal` sont des champs interdits qui ne doivent pas figurer dans le document de politique de 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, s'il dispose d'une politique basée sur l'identité, telle que `AmazonRekognitionCustomLabelsFullAccess`, qui autorise l'appel. `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 (kit 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. Remplacez les valeurs suivantes :

- Effet — spécifiez `ALLOW` pour accorder l'autorisation de copie. Spécifiez `DENY` pour refuser l'autorisation de copie.
- Principal — pour le principal à qui vous souhaitez autoriser ou refuser l'accès aux versions du modèle que vous spécifiez dans `Resource`. Par exemple, vous pouvez spécifier le principal du compte AWS pour un autre AWS compte. Nous ne limitons pas les principaux que vous pouvez utiliser. Pour plus d'informations, consultez Spécification d'un principal.
- Ressource — Amazon Resource Name (ARN) de la version du modèle pour laquelle vous souhaitez spécifier les autorisations de copie. Si vous souhaitez accorder des autorisations à toutes les versions du modèle dans le projet source, utilisez le format `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 du projet sur votre ordinateur.
3. Joignez la politique du projet au projet source en suivant les instructions de la page Joindre une politique de projet (kit SDK).

Joindre une politique de projet (kit SDK)

Vous associez une politique de projet à un projet Amazon Rekognition Custom Labels en appelant l'opération. PutProjectpolicy

Joignez plusieurs politiques de projet à un projet en appelant `PutProjectPolicy` pour chaque politique de projet que vous souhaitez ajouter. Vous pouvez ajouter jusqu'à cinq politiques de

projet à un projet. Si vous devez joindre d'autres politiques 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 d'entrée `PolicyRevisionId`. La réponse de `PutProjectPolicy` est un ID de révision pour la politique de projet créée pour vous par Étiquettes personnalisées Amazon Rekognition. Vous pouvez utiliser l'ID de révision pour mettre à jour ou supprimer la dernière révision d'une politique de projet. Étiquettes personnalisées Amazon Rekognition conserve uniquement la dernière révision d'une politique de projet. Si vous essayez de mettre à jour ou de supprimer une révision précédente d'une politique de projet, une erreur `InvalidPolicyRevisionIdException` s'affiche.

Pour mettre à jour une politique de projet existante, spécifiez l'ID de révision de la politique de projet dans le paramètre d'entrée `PolicyRevisionId`. Vous pouvez obtenir la révision IDs des politiques de projet dans un projet en appelant ListProjectPolicies.

Après avoir joint une politique de projet à un projet source, vous pouvez copier le modèle du projet source vers le projet de destination. Pour de plus amples informations, veuillez consulter Copie d'un modèle (kit SDK).

Pour supprimer une politique de projet d'un projet, appelez DeleteProjectPolicy. Pour obtenir une liste des politiques de projet associées à un projet, appelez ListProjectPolicies.

Pour joindre une politique de projet à un projet (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Créez un document de politique de projet.
3. Utilisez le code suivant pour associer la politique du projet au projet, dans le AWS compte fiduciaire, qui contient la version du modèle que vous souhaitez copier. Pour obtenir l'ARN du projet, appelez DescribeProjects. Pour obtenir la version du modèle, appelez l'ARN DescribeProjectVersions.

   AWS CLI

   Remplacez 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` par un nom de politique de votre choix.
- `principal` par le principal à qui vous souhaitez autoriser ou refuser l'accès aux versions du modèle que vous spécifiez dans `Model version ARN`.
- `project-version-arn` par l'ARN de la version du modèle que vous souhaitez copier.

Si vous voulez mettre à jour une politique de projet existante, spécifiez le paramètre `policy-revision-id` 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` — ARN du projet source auquel vous souhaitez joindre la politique du projet.
- `policy_name` — un nom de politique de votre choix.
- `project_policy` — 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` — ARN du projet source auquel vous souhaitez joindre la politique du projet.
- `project_policy_name` — un nom de politique de votre choix.
- `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 sur la page Copie d'un modèle (kit SDK).

Copie d'un modèle (kit SDK)

Vous pouvez utiliser l'API `CopyProjectVersion` pour copier une version de modèle d'un projet source vers un projet de destination. Le projet de destination peut se trouver dans un autre AWS compte, mais il 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 du modèle copiée dans un AWS compte), vous devez joindre une politique de projet au projet source. Pour de plus amples informations, veuillez consulter Création d'un document de politique de projet. L'API `CopyProjectVersion` nécessite l'accès à votre compartiment Amazon S3.

Le modèle copié inclut les résultats d'entraînement du modèle source, mais pas les jeux de données source.

Le AWS compte source n'est pas propriétaire du modèle copié dans un compte de destination, sauf si vous configurez les autorisations appropriées.

Pour copier un modèle (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Joignez la politique du projet au projet source en suivant les instructions de la page Joindre une politique de projet (kit SDK).
3. Si vous copiez le modèle vers un autre AWS compte, assurez-vous que le AWS compte de destination contient un projet.
4. Utilisez le code suivant pour copier la version de modèle dans un projet de destination :

   AWS CLI

   Remplacez les valeurs suivantes :

   - `source-project-arn` par l'ARN du projet source qui contient la version du modèle que vous souhaitez copier.
   - `source-project-version-arn` par l'ARN de la version du modèle que vous souhaitez copier.
   - `destination-project-arn` par l'ARN du projet de destination vers lequel vous souhaitez copier le modèle.
   - `version-name` par le nom de version pour le modèle dans le projet de destination.
   - `bucket` par le compartiment S3 dans lequel vous souhaitez copier les résultats de l'entraînement pour le modèle source.
   - `folder` par le dossier dans le `bucket` dans lequel vous souhaitez copier les résultats de l'entraînement pour le modèle source.
   - (Facultatif) `kms-key-id` par l'ID de clé AWS Key Management Service pour le modèle.
   - (Facultatif) `key` par une clé de balise de votre choix.
   - (Facultatif) `value` par une 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 le AWS compte source qui contient la version du modèle que vous souhaitez copier.
- `source_project_version-arn`— l'ARN de la version du modèle dans le AWS compte source que vous souhaitez copier.
- `destination_project_arn` — ARN du projet de destination vers lequel vous souhaitez copier le modèle.
- `destination_version_name` — nom de version pour le modèle dans le projet de destination.
- `training_results` — emplacement S3 dans lequel vous souhaitez copier les résultats de l'entraînement pour la version de modèle source.
- (Facultatif) `kms_key_id` par l'ID de clé AWS Key Management Service pour le modèle.
- (Facultatif) `tag_name` par une clé de balise de votre choix.
- (Facultatif) `tag_value` par une 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 le AWS compte source qui contient la version du modèle que vous souhaitez copier.
- `source_project_version-arn`— l'ARN de la version du modèle dans le AWS compte source que vous souhaitez copier.
- `destination_project_arn` — ARN du projet de destination vers lequel vous souhaitez copier le modèle.
- `destination_version_name` — nom de version pour le modèle dans le projet de destination.
- `output_bucket` — compartiment S3 dans lequel vous souhaitez copier les résultats de l'entraînement pour la version de modèle source.
- `output_folder` — dossier du compartiment S3 dans lequel vous souhaitez copier les résultats de l'entraînement pour la version de 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 (kit SDK)

Vous pouvez utiliser cette ListProjectPoliciesopération pour répertorier les politiques de projet associées à un projet Amazon Rekognition Custom Labels.

Pour répertorier les politiques de projet jointes à un projet (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez le code suivant pour répertorier les politiques de projet.

AWS CLI

Remplacez `project-arn` par l'Amazon Resource Name du projet pour lequel vous souhaitez répertorier les politiques de projet jointes.

```
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 — Amazon Resource Name du projet pour lequel vous souhaitez répertorier les politiques de projet jointes.

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 Rekogntion 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 — 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);
        }

    }

}
```

Suppression d’une politique de projet (kit SDK)

Vous pouvez utiliser cette DeleteProjectPolicyopération pour supprimer une révision d'une politique de projet existante d'un projet Amazon Rekognition Custom Labels. Si vous souhaitez supprimer toutes les révisions d'une politique de projet associées à un projet, utilisez cette option ListProjectPoliciespour obtenir la révision IDs de chaque politique de projet attachée au projet. Appelez ensuite `DeletePolicy` pour le nom de chaque politique.

Pour supprimer une révision d’une politique de projet (kit SDK)

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.
2. Utilisez le code suivant pour supprimer une politique de projet.

   DeletePolicy prend`ProjectARN`, `PolicyName` et`PolicyRevisionId`. `ProjectARN`et `PolicyName` sont nécessaires pour cette API. `PolicyRevisionId`est facultatif, mais peut être inclus à des fins de mises à jour atomiques.

   AWS CLI

   Remplacez les valeurs suivantes :

   - `policy-name` par le nom de la politique de projet que vous souhaitez supprimer.
   - `policy-revision-id` par l’ID de révision de la politique de projet que vous souhaitez supprimer.
   - `project-arn` par l’Amazon Resource Name du projet qui contient la politique de 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` — nom de la politique de projet que vous souhaitez supprimer.

- `project-arn` — Amazon Resource Name du projet qui contient la politique de projet que vous souhaitez supprimer.
- `policy-revision-id` — 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 revsion 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 wany 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` — nom de la politique de projet que vous souhaitez supprimer.
- `project-arn` — Amazon Resource Name du projet qui contient la politique de projet que vous souhaitez supprimer.
- `policy-revision-id` — 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 d'étiquettes personnalisées

Cette section contient des exemples qui vous montrent comment utiliser les fonctionnalités d'Amazon Rekognition Custom Labels.

exemple	Description
Améliorer un modèle grâce aux commentaires sur le modèle	Montre comment améliorer un modèle à l'aide de la vérification humaine pour créer un nouveau jeu de données d'entraînement.
Démonstration d'Étiquettes personnalisées Amazon Rekognition	Démonstration d'une interface utilisateur qui affiche les résultats d'un appel à `DetectCustomLabels` .
Détection d'étiquettes personnalisées dans les vidéos	Montre comment utiliser `DetectCustomLabels` avec des trames extraites d'une vidéo.
Analyse d'images à l'aide d'une AWS Lambda fonction	Montre comment utiliser `DetectCustomLabels` 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 d'objets, de scènes et de concepts associés à une image entière (classification).

Améliorer un modèle grâce aux commentaires sur le modèle

La solution de commentaires sur un modèle vous permet de fournir un commentaire sur les prédictions de votre modèle et de l'améliorer en utilisant la vérification humaine. Selon le cas d'utilisation, vous pouvez réussir avec un jeu de données d'entraînement contenant seulement quelques images. Un jeu d'entraînement annoté plus important peut être nécessaire pour créer un modèle plus précis. À l'aide de la solution de commentaires sur un modèle, vous pouvez créer un jeu de données plus important grâce à l'assistance du modèle.

Pour installer et configurer la solution de commentaires sur un modèle, consultez Model Feedback Solution.

Le flux de travail pour l'amélioration continue d'un modèle est comme suit :

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 de commentaires sur un modèle.
3. La solution de commentaires sur un modèle utilise le modèle actuel. Elle lance des tâches de vérification humaine pour annoter un nouveau jeu de données.
4. Sur la base des commentaires humains, la solution de commentaires sur un modèle génère un fichier manifeste que vous utilisez pour créer un nouveau modèle.

Démonstration d'Étiquettes personnalisées Amazon Rekognition

La démonstration des étiquettes personnalisées Amazon Rekognition présente une interface utilisateur qui analyse les images de votre ordinateur local à l'aide de l'API. DetectCustomLabels

L'application affiche des informations sur les modèles d'étiquettes personnalisées Amazon Rekognition présents dans votre compte. AWS Après avoir sélectionné un modèle en cours d'exécution, vous pouvez analyser une image à partir de 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, consultez Amazon Rekognition Custom Labels Demo.

Détection d'étiquettes personnalisées dans les vidéos

L'exemple suivant montre comment vous pouvez utiliser `DetectCustomLabels` avec des trames extraites d'une vidéo. Le code a été testé avec des fichiers vidéo au format mov et mp4.

Utilisation de **`DetectCustomLabels`** avec des trames capturées

1. Si ce n'est pas déjà fait, installez et configurez le AWS CLI et le AWS SDKs. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.

2. Veillez à disposer des autorisations `rekognition:DetectCustomLabels` et `AmazonS3ReadOnlyAccess`. Pour plus d'informations, consultez Étape 4 : Configurez le AWS CLI et AWS SDKs.

3. Utilisez l'exemple de code suivant. Remplacez la valeur de `videoFile` par le nom du fichier vidéo. Remplacez la valeur de `projectVersionArn` par l'Amazon Resource Name (ARN) du modèle 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'une AWS Lambda fonction

AWS Lambda est un service de calcul qui vous permet d'exécuter du code sans provisionner ni gérer de serveurs. Par exemple, vous pouvez analyser les images soumises 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 dans Python qui appelle DetectCustomLabels. La fonction analyse une image fournie et renvoie une liste des étiquettes trouvées dans l'image. Les instructions incluent un exemple de code Python montrant comment appeler la fonction Lambda avec une image dans un compartiment Amazon S3 ou une image fournie depuis un ordinateur local.

Rubriques

Étape 1 : Création d'une AWS Lambda fonction (console)

Au cours de cette étape, vous créez une AWS fonction vide et un rôle d'exécution IAM qui permet à votre fonction d'appeler l'`DetectCustomLabels`opération. Il donne également accès au compartiment Amazon S3 qui stocke des images à des fins d’analyse. Vous spécifiez également des variables d’environnement pour les éléments suivants :

- Le modèle Étiquettes personnalisées Amazon Rekognition que vous souhaitez que votre fonction Lambda utilise.
- La limite de confiance que vous souhaitez que le modèle utilise.

Vous ajoutez ensuite le code source et éventuellement une couche à la fonction Lambda.

Pour créer une AWS Lambda fonction (console)

1. Connectez-vous à la AWS Lambda console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/lambda/.
2. Sélectionnez Create function (Créer une fonction). Pour plus d’informations, consultez Créer une fonction Lambda à l’aide de la console.
3. Choisissez les options suivantes.
   - Choisissez Créer à partir de zéro.
   - Entrez une valeur pour Nom de la fonction.
   - Pour Environnement d’exécution, choisissez Python 3.10.
4. Choisissez Créer une fonction pour créer la fonction AWS Lambda .
5. Sur la page de la fonction, choisissez l’onglet Configuration.
6. Dans le volet Variables d’environnement, choisissez Modifier.
7. Ajoutez les variables d’environnement suivantes. Pour chaque variable, choisissez Ajouter des variables d’environnement, puis entrez la clé et la valeur de la variable.

Clé	Valeur
MODEL_ARN	L’Amazon Resource Name (ARN) du modèle que vous souhaitez que votre fonction Lambda utilise. Vous pouvez obtenir l’ARN du modèle depuis l’onglet Utiliser le modèle

Clé	Valeur
	de la page de détails du modèle dans la console Étiquettes personnalisées Amazon Rekognition.
CONFIDENCE	La valeur maximum (0 à 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. Dans le volet Autorisations, sous Nom du rôle, choisissez le rôle d'exécution pour ouvrir le rôle dans la console IAM.
10. Dans l'onglet Autorisations, choisissez Ajouter des autorisations, puis Créer une politique en ligne.
11. Choisissez JSON et remplacez la politique existante avec la suivante.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "rekognition:DetectCustomLabels",
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "DetectCustomLabels"
        }
    ]
}
```

12. Choisissez Suivant.
13. Dans Détails de la politique, entrez le nom de la politique, tel que DetectCustomLabels-access.
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, utilisez la politique suivante. *bucket/folder path*Remplacez-le par 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 S3Bucket-access.

Étape 2 : (Facultatif) Créer une couche (console)

Pour exécuter cet exemple, vous n'avez pas besoin de suivre cette étape. L'`DetectCustomLabels`opération est incluse dans l'environnement Lambda Python par défaut dans le cadre du AWS SDK pour Python (Boto3). Si d'autres parties de votre fonction Lambda nécessitent des mises à jour de AWS service récentes qui ne 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.

Vous devez d'abord créer une archive de fichier .zip contenant le kit SDK Boto3. Ensuite, vous créez une couche et ajoutez l'archive du fichier .zip à la couche. Pour plus d'informations, consultez Utilisation de couches avec votre fonction Lambda.

Pour créer et ajouter une couche (console)

1. Ouvrez une invite de commande 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 AWS Lambda console à l'adresse https://console.aws.amazon.com/lambda/.
4. Choisissez Layers dans le volet de navigation.
5. Choisissez Créer un calque.
6. Saisissez un Name (Nom) et une Description pour la règle.
7. Choisissez Charger un fichier .zip, puis Charger.
8. Dans la boîte de dialogue, sélectionnez 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 Créer pour créer la couche.
11. Choisissez l'icône du menu du volet de navigation.
12. Dans le volet de navigation, choisissez Fonctions.
13. Dans la liste des ressources, choisissez la fonction que vous avez créée dans Étape 1 : Création d'une AWS Lambda fonction (console).
14. Cliquez sur l'onglet Code.
15. Dans la zone Couches, choisissez Ajouter une couche.
16. Choisissez Couches personnalisées.
17. Dans Couches personnalisées, choisissez le nom de couche que vous avez saisi à l'étape 6.
18. Dans Version, choisissez la version de la couche, qui doit être 1.
19. Choisissez Ajouter.

Étape 3 : Ajouter le code Python (console)

Au cours de cette étape, vous ajoutez le 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 `DetectCustomLabels` et renvoie une liste des étiquettes trouvées dans l'image. L'image fournie peut être située dans un compartiment Amazon S3 ou fournie sous forme d'octets d'image codés en byte64.

Pour ajouter le code Python (console)

1. Si vous n'êtes pas dans la console Lambda, procédez comme suit :

   a. Ouvrez la AWS Lambda console à l'adresse https://console.aws.amazon.com/lambda/.

b. Ouvrez la fonction Lambda que vous avez créée dans Étape 1 : Création d'une AWS Lambda fonction (console).

2. Cliquez sur l'onglet Code.

3. Dans Source du code, remplacez le code dans lambda_function.py par le code suivant :

```
# 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 : Essayer votre fonction Lambda

Au cours de cette étape, vous exécutez du code Python sur votre ordinateur pour transmettre une image locale ou une image d'un compartiment Amazon S3 à votre fonction Lambda. Les images transmises par un ordinateur local doivent avoir une taille inférieure à 6 291 456 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 des fichiers image dans un compartiment Amazon S3, consultez Chargement d'objets.

Assurez-vous d'exécuter le code dans la même AWS région que celle 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 de la console Lambda.

Si la AWS Lambda fonction renvoie une erreur de temporisation, prolongez le délai d'expiration de 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, consultez AWS Lambda Invoking Functions.

Pour essayer votre fonction Lambda

1. Veillez à disposer de l'autorisations `lambda:InvokeFunction`. 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 l'aperçu des fonctions dans la console Lambda.

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

   - Utilisateurs et groupes dans AWS IAM Identity Center :

     Créez un jeu d'autorisations. Suivez les instructions de la rubrique Création d'un jeu d'autorisations du Guide de l'utilisateur AWS IAM Identity Center .
   - Utilisateurs gérés dans IAM par un fournisseur d'identité :

     Créez un rôle pour la fédération d'identité. Suivez les instructions de la rubrique Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM.
   - Utilisateurs IAM :
     - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d'un rôle pour un utilisateur IAM dans le Guide de l'utilisateur IAM.
     - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique Ajout d'autorisations à un utilisateur (console) du Guide de l'utilisateur IAM.
2. Installez et configurez le AWS SDK pour Python. Pour de plus amples informations, veuillez consulter Étape 4 : Configurez le AWS CLI et AWS SDKs.

3. Démarrez le modèle que vous avez spécifié à l'étape 7 de l'Étape 1 : Création d'une 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 l'Étape 1 : Création d'une AWS Lambda fonction (console).

   En cas de réussite, le résultat est une liste d'étiquettes présentes dans l'image. Si aucune étiquette n'est renvoyée, pensez à réduire la valeur de confiance que vous avez définie à l'étape 7 de l'Étape 1 : Création d'une AWS Lambda fonction (console).

6. Si vous avez terminé avec la fonction Lambda et que le modèle n'est pas utilisé par d'autres applications, arrêtez le modèle. N'oubliez pas de démarrer le modèle la prochaine fois que vous utiliserez la fonction Lambda.

Sécurité

Vous pouvez sécuriser la gestion de vos projets, de vos modèles et de l'opération `DetectCustomLabels` utilisée par vos clients pour détecter les étiquettes personnalisées.

Pour plus d'informations sur la sécurisation d'Amazon Rekognition, consultez Sécurité Amazon Rekognition.

Sécurisation des projets Étiquettes personnalisées Amazon Rekognition

Vous pouvez sécuriser vos projets Étiquettes personnalisées Amazon Rekognition en spécifiant les autorisations au niveau des ressources 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 Étiquettes personnalisées Amazon Rekognition que vous pouvez sécuriser sont les suivantes :

Ressource	Format Amazon Resource Name
Projet	arn:aws:rekognition :*:*:project/ /datetime *project_name*
Modèle	arn:aws:rekognition :*:*:project/ /version/ / datetime *project_name name*

L'exemple de politique suivant montre comment accorder à une identité l'autorisation de :

- Décrire tous les projets.
- Créer, démarrer, arrêter et utiliser un modèle spécifique pour l'inférence.
- Créez un projet. Créer et décrire 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 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 pour 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 la politique `AmazonRekognitionCustomLabelsFullAccess` AWS gérée que vous pouvez utiliser pour contrôler l'accès aux étiquettes personnalisées Amazon Rekognition. Pour plus d'informations, consultez la politique gérée par AWS : AmazonRekognitionCustomLabelsFullAccess.

Directives et quotas dans Étiquettes personnalisées Amazon Rekognition

Les sections suivantes fournissent les directives et les quotas qui s'appliquent à l'utilisation d'Étiquettes personnalisées Amazon Rekognition.

Régions prises en charge

Pour obtenir la liste des AWS régions dans lesquelles les étiquettes personnalisées Amazon Rekognition sont disponibles, consultez la section Régions et points de terminaison AWS dans le manuel Amazon Web Services General Reference.

Quotas

Voici une liste de limitations dans Étiquettes personnalisées Amazon Rekognition. 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 maximum du fichier manifeste du jeu de données est de 1 Go.
- Le nombre minimum d'étiquettes uniques par jeu de données Objets, Scènes et Concepts (classification) est de 2.
- Le nombre minimum d'étiquettes uniques par jeu de données Emplacement d'objets (détection) est de 1.
- Le nombre maximum d'étiquettes uniques par manifeste est de 250.
- Le nombre minimum d'images par étiquette est de 1.
- Le nombre maximum d'images par jeu de données Emplacement 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 Objets, Scènes et Concepts (classification) est de 500 000. La valeur par défaut est 250 000. Pour demander une augmentation, consultez Créer une demande.

  La limite pour les AWS régions Asie-Pacifique (Mumbai) et Europe (Londres) est de 28 000 images. Vous ne pouvez pas demander une 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 minimum d'un fichier image dans un compartiment Amazon S3 est de 64 pixels x 64 pixels.
- La dimension maximum d'un fichier image dans un compartiment Amazon S3 est de 4 096 pixels x 4 096 pixels.
- La taille maximum d'une image dans un compartiment Amazon S3 est de 15 Mo.
- Le rapport de forme 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 maximum du fichier manifeste du jeu de données est de 1 Go.
- Le nombre minimum d'étiquettes uniques par jeu de données Objets, Scènes et Concepts (classification) est de 2.
- Le nombre minimum d'étiquettes uniques par jeu de données Emplacement d'objets (détection) est de 1.
- Le nombre maximum d'étiquettes uniques par jeu de données est de 250.
- Le nombre minimum d'images par étiquette est de 0.
- Le nombre maximum d'images par étiquette est de 1 000.
- Le nombre maximum d'images par jeu de données Emplacement 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 Objets, Scènes et Concepts (classification) est de 500 000. La valeur par défaut est 250 000. Pour demander une augmentation, consultez Créer une demande.

La limite pour les AWS régions Asie-Pacifique (Mumbai) et Europe (Londres) est de 7 000 images. Vous ne pouvez pas demander une augmentation de limite.

- Le nombre minimum d'étiquettes par image par manifeste est de 0.
- Le nombre maximum d'étiquettes par image 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 minimum d'un fichier image dans un compartiment Amazon S3 est de 64 pixels x 64 pixels.
- La dimension maximum d'un fichier image dans un compartiment Amazon S3 est de 4 096 pixels x 4 096 pixels.
- La taille maximum d'une image dans 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 de forme de l'image est de 20:1.

Détection

- La taille maximum des images transmises sous forme d'octets bruts est de 4 Mo.
- La taille maximum d'une image dans un compartiment Amazon S3 est de 15 Mo.
- La dimension minimum d'un fichier image d'entrée (stocké dans un compartiment Amazon S3 ou fourni en tant qu'octets d'image) est de 64 pixels x 64 pixels.
- La dimension maximum d'un fichier image d'entrée (stocké dans un compartiment Amazon S3 ou fourni en tant qu'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 de forme de l'image est de 20:1.

Copie d'un modèle

- Le nombre maximum de politiques de projet que vous pouvez associer à un projet est de 5.
- Le nombre maximum de tâches de copie simultanées dans une destination est de 5.

Référence de l’API Étiquettes personnalisées Amazon Rekognition

L’API Étiquettes personnalisées Amazon Rekognition est documentée dans le cadre du contenu de référence d’API Amazon Rekognition. Voici une liste des opérations de l’API Étiquettes personnalisées Amazon Rekognition avec des liens vers la rubrique de référence d’API Amazon Rekognition appropriée. Les liens de référence d’API contenus dans ce document renvoient également à la rubrique de référence d’API appropriée du Guide du développeur Amazon Rekognition. Pour plus d’informations sur l’utilisation de l’API, consultez

Cette section présente un aperçu du flux de travail permettant de former et d'utiliser un modèle d'étiquettes personnalisées Amazon Rekognition avec la console et le SDK. AWS

> **Note**
>
> Étiquettes personnalisées Amazon Rekognition gère désormais l’association de jeux de données à un projet. Vous pouvez créer des ensembles de données pour vos projets à l'aide de la console et du AWS SDK. Si vous avez déjà utilisé Étiquettes personnalisées Amazon Rekognition, vos anciens jeux de données devront peut-être être associés à un nouveau projet. Pour plus d’informations, consultez Étape 6 (Facultatif) : Associer des jeux de données précédents à de nouveaux projets.

Rubriques

Choix de votre type de modèle

Vous décidez d’abord du type de modèle que vous souhaitez entraîner en fonction des objectifs de votre entreprise. Par exemple, vous pouvez entraîner un modèle à rechercher votre logo dans des

publications des réseaux sociaux, à identifier vos produits dans des rayons de magasins ou à classer les pièces d'une machine sur une chaîne de montage.

Étiquettes personnalisées Amazon Rekognition peut entraîner les types de modèles suivants :

- Recherche d'objets, de scènes et de concepts
- Recherche des emplacements d'objets
- Recherche de l'emplacement de marques

Pour vous aider à choisir le type de modèle à entraîner, Étiquettes personnalisées Amazon Rekognition fournit des exemples de projet que vous pouvez utiliser. Pour plus d'informations, consultez Mise en route sur Étiquettes personnalisées Amazon Rekognition.

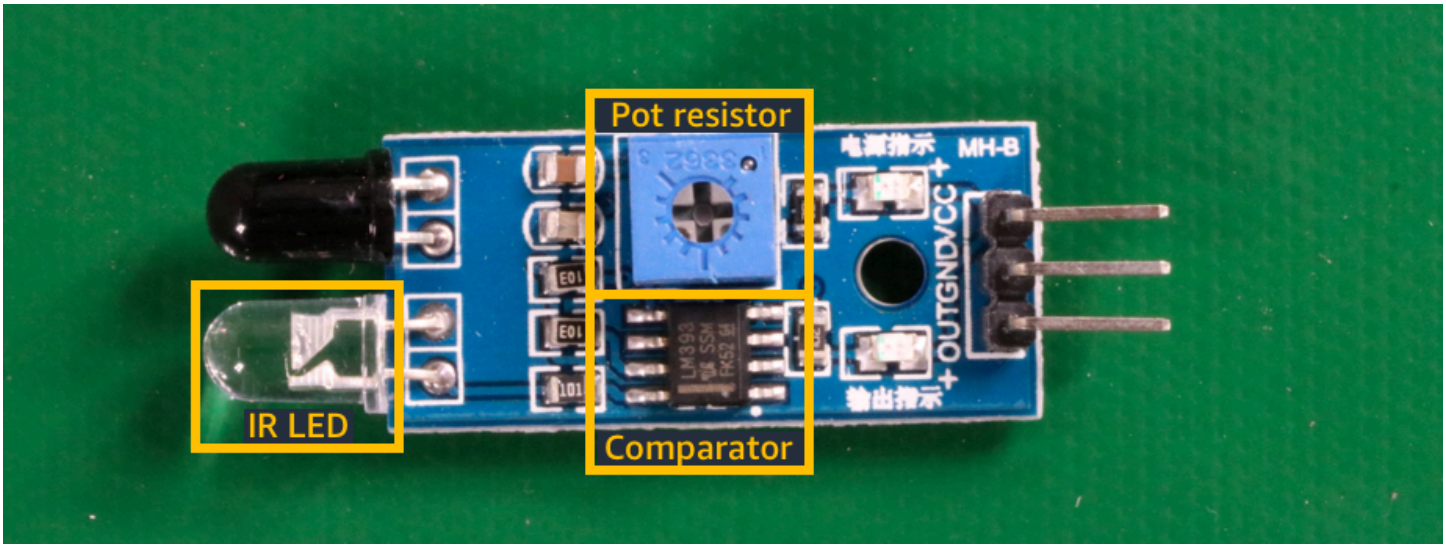
Recherche d'objets, de scènes et de concepts

Le modèle prédit des classifications pour les objets, les scènes et les concepts associés à une image entière. Par exemple, vous pouvez entraîner un modèle à déterminer si une image contient une attraction touristique ou non. Pour un exemple de projet, consultez Classification d'images. L'image suivante d'un lac est un exemple du type d'image dans laquelle vous pouvez reconnaître des objets, des scènes et des concepts.

Vous pouvez également entraîner un modèle à classer les images en plusieurs catégories. Par exemple, l'image précédente peut comporter des catégories telles que couleur du ciel, reflet ou lac. Pour un exemple de projet, consultez Classification des images à plusieurs étiquettes.

Recherche des emplacements d'objets

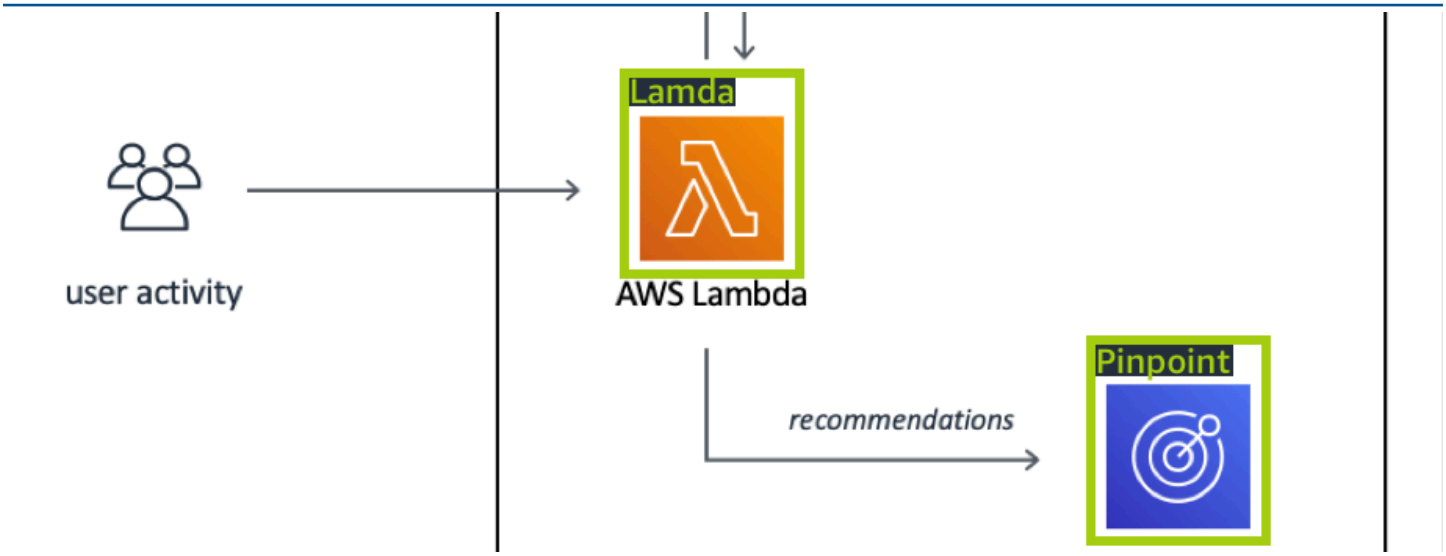
Le modèle prédit l'emplacement d'un objet sur une image. La prédiction comprend des informations sur le cadre de délimitation de l'emplacement d'objet et une étiquette qui identifie l'objet dans le cadre de délimitation. Par exemple, l'image suivante montre des cadres de délimitation autour de différents composant d'un circuit imprimé, comme comparateur ou potentiomètre à résistance.



L'exemple de projet Localisation d'objets montre comment Étiquettes personnalisées Amazon Rekognition utilise des cadres de délimitation étiquetés pour entraîner un modèle à rechercher les emplacements d'objets.

Recherche de l'emplacement de marques

Étiquettes personnalisées Amazon Rekognition peut entraîner un modèle à rechercher l'emplacement de marques (logos, par exemple) sur une image. La prédiction comprend des informations sur le cadre de délimitation de l'emplacement de la marque et une étiquette qui identifie l'objet dans le cadre de délimitation. Pour un exemple de projet, consultez Détection des marques. L'image suivante est un exemple de certaines marques que le modèle peut détecter.



Création d’un modèle

La création d’un modèle inclut la création d’un projet, la création de jeux de données d’entraînement et de test, et l’entraînement du modèle.

Création d’un projet

Un projet Étiquettes personnalisées Amazon Rekognition regroupe les ressources nécessaires à la création et à la gestion d’un modèle. Un projet gère les éléments suivants :

- Jeux de données : images et étiquettes d’images utilisées pour entraîner un modèle. Un projet comprend un jeu de données d’entraînement et un jeu de données de test.

- Modèles : logiciel que vous entraînez à rechercher des concepts, des scènes et des objets propres à votre entreprise. Un projet peut comprendre plusieurs versions d’un modèle.

Nous vous recommandons d’utiliser un projet pour un seul cas d’utilisation (recherche de composants sur un circuit imprimé, par exemple).

Vous pouvez créer un projet à l'aide de la console Amazon Rekognition Custom Labels et de l'API. CreateProject Pour de plus amples informations, veuillez consulter Création d’un 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. Dans votre projet, vous créez un jeu de données d’entraînement et un jeu de données de test qu’Étiquettes personnalisées Amazon Rekognition utilise pour entraîner et 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 soit attribuées à une image entière (au niveau de l’image), soit à un cadre de délimitation qui entoure un objet sur une image.

> **Important**
>
> La façon dont vous étiquetez les images des jeux de données détermine le type de modèle créé par Étiquettes personnalisées Amazon Rekognition. Par exemple, pour entraîner un modèle à rechercher des objets, des scènes et des concepts, vous attribuez des étiquettes de niveau image aux images de vos jeux de données d’entraînement et de test. Pour plus d’informations, consultez 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, consultez Préparation des images.

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

Vous pouvez démarrer un projet avec un seul jeu de données, ou avec des jeux de données d'entraînement et 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 les images qui sont utilisées pour l'entraînement et le test. 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 les jeux de données d'un projet, vous devez importer les images de l'une des manières suivantes :

- Importer des images de votre ordinateur local.
- Importer des images d'un compartiment S3. Étiquettes personnalisées Amazon Rekognition peut étiqueter les images en utilisant les noms des dossiers qui les contiennent.
- Importez un fichier manifeste Amazon SageMaker AI Ground Truth.
- Copier un jeu de données Étiquettes personnalisées Amazon Rekognition existant.

Pour plus d'informations, consultez Création de jeux de données d'entraînement et de test avec des images.

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 AI 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.

Pour créer vos jeux de données d'entraînement et de test avec la console, consultez Création de jeux de données d'entraînement et de test avec des images. Pour accéder à un tutoriel sur la création de jeux de données d'entraînement et de test, consultez Classification des images.

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

Pour créer vos jeux de données d'entraînement et de test, vous utilisez l'API `CreateDataset`. Vous pouvez créer un jeu de données en utilisant un fichier manifeste au format Amazon SageMaker ou en copiant un jeu de données Étiquettes personnalisées Amazon Rekognition existant. Pour plus d'informations, consultez 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, consultez the section called "Création d'un fichier manifeste".

Entraînement de votre modèle

Entraînez votre modèle avec le jeu de données d'entraînement. Une nouvelle version du modèle est créée chaque fois que celui-ci est entraîné. Pendant l'entraînement, Étiquettes personnalisées Amazon Rekognition teste les performances de votre modèle. Vous pouvez utiliser les résultats pour évaluer et améliorer votre modèle. L'entraînement prend un certain temps. Seuls les entraînements de modèles réussis vous sont facturés. Pour plus d'informations, consultez Entraînement d'un modèle Étiquettes personnalisées Amazon Rekognition. Si l'entraînement du modèle échoue, Étiquettes personnalisées Amazon Rekognition fournit des informations de débogage que vous pouvez utiliser. Pour plus d'informations, consultez Débogage d'un entraînement de modèle en échec.

Entraînement de votre modèle (console)

Pour entraîner votre modèle avec la console, consultez Entraînement d'un modèle (console).

Entraînement d'un modèle (kit SDK)

Vous formez un modèle d'étiquettes personnalisées Amazon Rekognition en appelant. CreateProjectVersion Pour de plus amples informations, veuillez consulter Entraînement d'un modèle (kit SDK).

Amélioration de votre modèle

Au cours du test, Étiquettes personnalisées Amazon Rekognition crée des métriques d'évaluation que vous pouvez utiliser pour améliorer votre modèle entraîné.

Évaluation de votre modèle

Évaluez les performances de votre modèle à l'aide des indicateurs de performance créés lors du test. Les indicateurs de performance (score F1, précision, rappel, etc.) vous permettent de comprendre les

performances de votre modèle entraîné et de déterminer si vous pouvez l'utiliser en production. Pour plus d'informations, consultez Métriques d'évaluation du modèle.

Évaluation d'un modèle (console)

Pour visualiser les indicateurs de performance, consultez Accès aux métriques d'évaluation (console).

Évaluation d'un modèle (kit SDK)

Pour obtenir des mesures de performance, vous appelez DescribeProjectVersionspour obtenir les résultats des tests. Pour de plus amples informations, veuillez consulter Accès aux métriques d'évaluation d'Étiquettes personnalisées Amazon Rekognition (kit SDK). Les résultats des tests incluent des métriques qui ne sont disponibles dans la console (matrice de confusion pour les résultats de classification, par exemple). 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, consultez F1.

- Emplacement du fichier de synthèse : le résumé des tests inclut toutes les métriques d'évaluation du jeu de données de test et les métriques de chaque étiquette. `DescribeProjectVersions` renvoie le compartiment S3 et l'emplacement du dossier du fichier de synthèse. Pour plus d'informations, consultez Accès au fichier récapitulatif du modèle.
- Emplacement de l'instantané du manifeste d'évaluation : l'instantané contient des informations sur les résultats des tests, notamment les indices 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 instantanés. Pour plus d'informations, consultez Interprétation de l'instantané du manifeste d'évaluation.

Amélioration de votre modèle

Si des améliorations sont nécessaires, vous pouvez ajouter d'autres images d'entraînement ou améliorer l'étiquetage des jeux de données. Pour plus d'informations, consultez Amélioration d'un modèle Étiquettes personnalisées Amazon Rekognition. Vous pouvez également donner votre avis sur les prédictions de votre modèle et l'utiliser pour améliorer votre modèle. Pour plus d'informations, consultez Améliorer un modèle grâce aux commentaires sur le modèle.

Amélioration de votre modèle (console)

Pour ajouter des images à un jeu de données, consultez Ajout d'autres images à un jeu de données. Pour ajouter ou modifier des étiquettes, consultez the section called "Étiquetage des images".

Pour réentraîner votre modèle, consultez Entraînement d'un modèle (console).

Amélioration de votre modèle (kit SDK)

Pour ajouter des images à un jeu de données ou modifier l'étiquetage d'une image, utilisez l'API `UpdateDatasetEntries`. `UpdateDatasetEntries` met à jour le fichier manifeste ou y ou ajoute des lignes JSON. Chaque ligne JSON contient des informations sur une seule image (étiquettes attribuées, informations sur les cadres de délimitation, par exemple). Pour plus d'informations, consultez Ajout d'autres images (kit SDK). Pour afficher les entrées d'un jeu de données, utilisez l'API `ListDatasetEntries`.

Pour réentraîner votre modèle, consultez Entraînement d'un modèle (kit SDK).

Démarrage de votre modèle

Avant de pouvoir utiliser votre modèle, vous devez le démarrer à l'aide de la console Étiquettes personnalisées Amazon Rekognition ou de l'API `StartProjectVersion`. La durée de fonctionnement de votre modèle vous est facturée. Pour plus d'informations, consultez Exécution d'un modèle entraîné.

Démarrage de votre modèle (console)

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

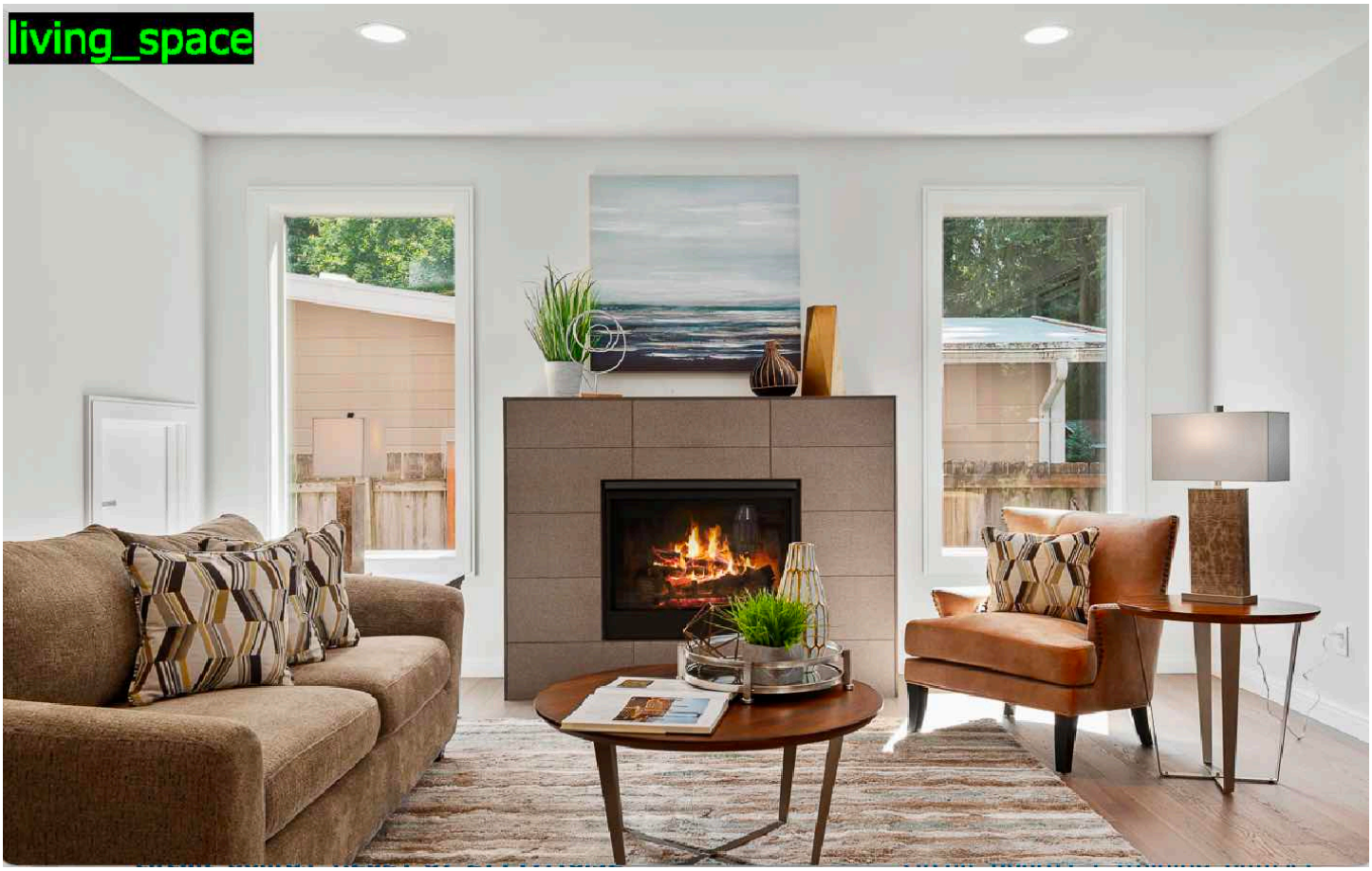
Démarrage de votre modèle

Vous commencez votre appel de mannequins StartProjectVersion. Pour de plus amples informations, veuillez consulter Démarrage d'un modèle Étiquettes personnalisées Amazon Rekognition (kit SDK).

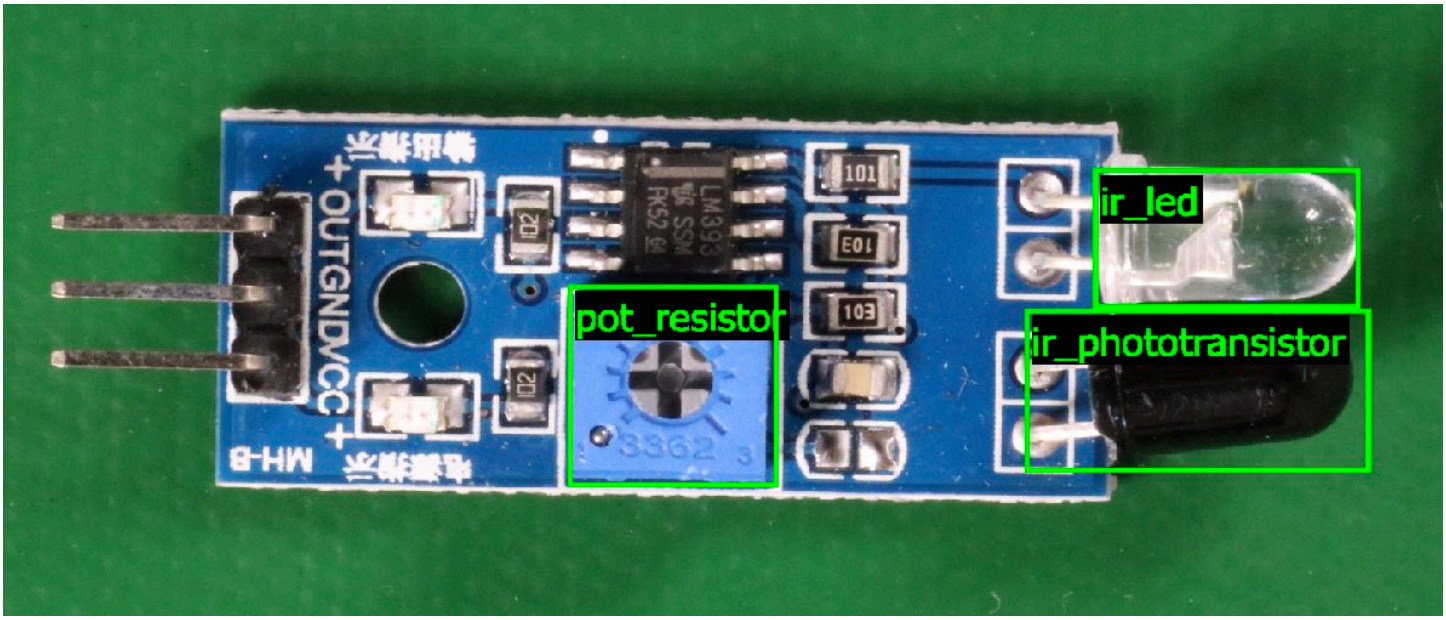
Analyse d'une image

Pour analyser une image avec votre modèle, vous utilisez l'API `DetectCustomLabels`. 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 détecte des objets, des scènes et des concepts, la réponse inclut la 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 détectées à l'aide d'un exemple de projet Pièces.



Si le modèle détecte des emplacements d'objets, la réponse inclut la liste des cadres de délimitation étiquetés trouvés 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 en dessiner un autour d'un objet. Par exemple, l'image suivante montre des cadres de délimitation autour des composants détectés à l'aide de l'exemple de projet Circuits imprimés.



Pour de plus amples informations, veuillez consulter Analyse d'une image avec un modèle entraîné.

Arrêt de votre modèle

La durée de fonctionnement de votre modèle vous est facturée. Si vous n'utilisez plus votre modèle, arrêtez-le à l'aide de la console Étiquettes personnalisées Amazon Rekognition ou de l'API `StopProjectVersion`. Pour plus d'informations, consultez Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition.

Arrêt de votre modèle (console)

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

Arrêt de votre modèle (kit SDK)

Pour arrêter un modèle en cours d'exécution, appelez StopProjectVersion. Pour de plus amples informations, veuillez consulter Arrêt d'un modèle Étiquettes personnalisées Amazon Rekognition (kit SDK).

.

Entraînement de votre modèle

Projets

- CreateProject— Crée votre projet Amazon Rekognition Custom Labels qui est un regroupement logique de ressources (images, étiquettes, modèles) et d'opérations (formation, évaluation et détection).
- DeleteProject— Supprime un projet Amazon Rekognition Custom Labels.
- DescribeProjects— Renvoie une liste de tous vos projets Amazon Rekognition Custom Labels.

Politiques du projet

- PutProjectPolicy— Associe une politique de projet à un projet Amazon Rekognition Custom Labels dans un compte de confiance. AWS
- 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 ensemble de données d'étiquettes personnalisées Amazon Rekognition.
- DeleteDataset— Supprime un ensemble de données Amazon Rekognition Custom Labels.
- DescribeDataset— Décrit un ensemble de données d'étiquettes personnalisées Amazon Rekognition.
- DistributeDatasetEntries— Distribue les entrées (images) d'un jeu de données d'apprentissage entre le jeu de données d'entraînement et le jeu de données de test d'un projet.
- ListDatasetEntries— Renvoie une liste d'entrées (images) d'un ensemble de données Amazon Rekognition Custom Labels.
- ListDatasetLabels— Renvoie une liste d'étiquettes attribuées à un ensemble de données d'étiquettes personnalisées Amazon Rekognition.
- UpdateDatasetEntries— Ajoute ou met à jour des entrées (images) dans un ensemble de données Amazon Rekognition Custom Labels.

Modèles

- CreateProjectVersion— Entraîne votre modèle d'étiquettes personnalisées Amazon Rekognition.
- CopyProjectVersion— Copie votre modèle d'étiquettes personnalisées Amazon Rekognition.
- DeleteProjectVersion— Supprime un modèle d'étiquettes personnalisées Amazon Rekognition.
- DescribeProjectVersions— Renvoie une liste de tous les modèles d'étiquettes personnalisées Amazon Rekognition inclus dans un projet spécifique.

Balises

- 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 de votre modèle

- DetectCustomLabels— Analyse une image à l'aide de votre modèle d'étiquettes personnalisé.
- StartProjectVersion— Démarre votre modèle d'étiquettes personnalisées.
- StopProjectVersion— Arrête votre modèle d'étiquettes personnalisées.

Historique de la documentation Étiquettes personnalisées Amazon Rekognition

Le tableau suivant décrit les modifications importantes apportées à chaque version du Guide du développeur Étiquettes personnalisées Amazon Rekognition. Pour recevoir les notifications de mise à jour de cette documentation, abonnez-vous à un flux RSS.

- Date de la dernière mise à jour de la documentation : 19 avril 2023

Modification	Description	Date
Ajout d'une rubrique sur la durée du modèle	Montre comment connaître le nombre d'heures d'exécution et les unités d'inférence utilisées par un modèle. Pour plus d'informations, consultez Rapport sur la durée d'exécution et les unités d'inférence utilisées.	19 avril 2023
Réorganisation du contenu des jeux de données	Déplacement du contenu relatif à la création d'un fichier manifeste vers Fichier manifeste. Déplacement des rubriques liées à la conversion des jeux de données vers Conversion d'autres formats de jeux de données en fichier manifeste.	20 février 2023
Mise à jour des directives IAM pour AWS WAF	Mise à jour du guide s'aligner sur les bonnes pratiques IAM. Pour plus d'informations, consultez Bonnes pratiques de sécurité dans IAM.	15 février 2023

Affichage de la matrice de confusion pour un modèle de classification	La console Étiquettes personnalisées Amazon Rekognition n'affiche pas la matrice de confusion pour modèle de classification. Vous pouvez plutôt utiliser le AWS SDK pour obtenir et afficher une matrice de confusion . Pour plus d'informations, consultez Affichage de la matrice de confusion pour un modèle.	4 janvier 2023
Mise à jour de l'exemple de fonction Lambda	L'exemple de fonction Lambda montre désormais comment analyser les images transmises à partir d'un fichier local ou d'un compartim ent Amazon S3. Pour plus d'informations, consultez Analyse des images à l'aide d'une fonction AWS Lambda.	2 décembre 2022
Copie par la fonctionnalité Étiquettes personnalisées Amazon Rekognition des modèles entraînés	Vous pouvez désormais copier un modèle entraîné d'un AWS compte à un autre AWS dans la même AWS région. Pour plus d'informations, consultez Copie d'un modèle Étiquette s personnalisées Amazon Rekognition (kit SDK).	16 août 2022

Mise à l'échelle automatique des unités d'inférence par la fonctionnalité Étiquettes personnalisées Amazon Rekognition	Pour faire face aux pics de demande, la fonctionnalité Étiquettes personnalisées Amazon Rekognition peuvent désormais mettre à l'échelle le nombre d'unités d'inférence utilisées par votre modèle. Pour plus d'informations, consultez Exécution d'un modèle Étiquettes personnalisées Amazon Rekognition entraîné.	16 août 2022
Création d'un fichier manifeste à partir d'un fichier CSV	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, consultez Création d'un fichier manifeste à partir d'un fichier CSV.	2 février 2022
Gestion des jeux de données par la fonctionnalité Étiquettes personnalisées Amazon Rekognition via des projets	Vous pouvez utiliser des projets pour gérer les jeux de données d'entraînement et de test que vous utilisez pour créer un modèle. Pour plus d'informations, consultez Présentation de la fonctionnalité Étiquettes personnalisées Amazon Rekognition.	1er novembre 2021

Les étiquettes personnalisées Amazon Rekognition sont intégrées à AWS CloudFormation	Vous pouvez l'utiliser AWS CloudFormation pour approvisionner et configurer les projets Amazon Rekognition Custom Labels. Pour plus d'informations, consultez la section Création d'un projet avec AWS CloudFormation.	21 octobre 2021
Mise à jour de l’expérience de mise en route	La console Étiquettes personnalisées Amazon Rekognition inclut désormais des didacticiels vidéo et des exemples de projets. Pour plus d’informations, consultez Mise en route sur Étiquettes personnalisées Amazon Rekognition.	22 juillet 2021
Mise à jour des informations sur les seuils et l’utilisation des métriques	Informations sur la définition d’une valeur de seuil souhaitée à l’aide du paramètre d’entrée `MinConfidence` pour détecter les étiquettes personnalisées (`DetectCustomLabels`). Pour plus d’informations, consultez la section Analyse d’une image à l’aide d’un modèle entraîné.	8 juin 2021
AWS KMS key Support supplémentaire	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 du balisage	La fonctionnalité Étiquettes personnalisées Amazon Rekognition prend désormais en charge le balisage. Vous pouvez utiliser des balises pour identifier, organiser, rechercher et filtrer vos modèles Étiquettes personnalisées Amazon Rekognition. Pour plus d'informations, consultez Balisage d'un modèle.	25 mars 2021
Mise à jour de la rubrique de configuration	Mises à jour des informations de configuration concernant le chiffrement des fichiers d'entraînement. Pour plus d'informations, consultez Étape 5 (Facultatif) : Chiffrer les fichiers d'entraînement.	18 mars 2021
Ajout d'une rubrique concernant la copie d'un jeu de données	Informations sur la façon de copier un jeu de données dans une autre AWS région. Pour plus d'informations, voir Copier un jeu de données dans une autre AWS région.	5 mars 2021

Ajout d'une rubrique sur la transformation du manifeste GroundTruth multi-étiquette Amazon SageMaker AI	Informations sur la façon de transformer un manifeste au format GroundTruth multi-étiquettes Amazon SageMaker AI en un fichier manifeste au format Amazon Rekognition Custom Labels. Pour plus d'informations, consultez la section Transformation des fichiers manifestes multi-labels SageMaker AI Ground Truth.	22 février 2021
Ajout d’informations de débogage pour l’entraînement des modèles	Vous pouvez désormais utiliser les manifestes de résultats de validation pour obtenir des informations de débogage détaillées sur les erreurs d’entraînement des modèles. Pour plus d’informations, consultez Débogage d’un entraînement de modèle en échec.	8 octobre 2020
Ajout d’informations et d’un exemple sur la transformation COCO	Informations sur la façon de transformer un jeu de données au format de détection d’objets COCO en fichier manifeste Étiquettes personnalisées Amazon Rekognition. Pour plus d’informations, consultez Transformations des jeux de données COCO.	2 septembre 2020

Prise en charge de l'entraînement d'objet unique par la fonctionnalité Étiquettes personnalisées Amazon Rekognition	Pour créer un modèle Étiquettes personnalisées Amazon Rekognition qui trouve l'emplacement d'un seul objet, vous pouvez désormais créer un jeu de données ne nécessitant qu'une seule étiquette. Pour plus d'informations, consultez Traçage de cadres de délimitation.	25 juin 2020
Ajout d'opérations de suppression de projets et de modèles	Vous pouvez désormais supprimer les modèles et les projets Étiquettes personnalisées Amazon Rekognition à l'aide de la console et de l'API. Pour plus d'informations, consultez Suppression d'un modèle Étiquettes personnalisées Amazon Rekognition et Suppression d'un projet Étiquettes personnalisées Amazon Rekognition.	1er avril 2020
Ajout d'exemples Java	Ajout d'exemples Java couvrant la création de projets, l'entraînement de modèles, l'exécution de modèles et l'analyse d'images.	13 décembre 2019

Nouvelle fonctionnalité et nouveau guide	Il s’agit de la version initiale de la fonctionnalité Étiquettes personnalisées Amazon Rekognition et du Guide du développeur Étiquettes personnalisées Amazon Rekognition.	3 décembre 2019

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.