aws

Guide du développeur

Amazon Lookout for Vision

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

Amazon Lookout for Vision: Guide du développeur

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

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

Table of Contents

..... ix
Qu'est-ce qu'Amazon Lookout for Vision ? 1
Principaux avantages 1
Êtes-vous un utilisateur final d'Amazon Lookout for Vision pour la première fois ? 2
Configuration d'Amazon Lookout for Vision 3
Étape 1 : créer un AWS compte 3
Inscrivez-vous pour un Compte AWS 3
Création d’un utilisateur doté d’un accès administratif 4
Étape 2 : configurer les autorisations 5
Configuration de l'accès à la console avec des politiques AWS gérées 6
Configuration des autorisations du compartiment Amazon S3 6
Attribution d’autorisations 8
Étape 3 : Créer le compartiment de la console 8
Création du bucket de console avec la console Amazon Lookout for Vision 9
Création du compartiment de console avec Amazon S3 10
Paramètres du bucket de console 11
Étape 4 : Configurez le AWS CLI et AWS SDKs 11
Installez les AWS SDK 12
Octroi d’un accès par programmation 12
Configuration des autorisations du kit SDK 16
Appelez une opération Amazon Lookout for Vision 20
Étape 5 : (Facultatif) Utiliser votre propre AWS KMS clé 25
Comprendre Amazon Lookout for Vision 26
Choisissez votre type de modèle 27
Modèle de classification d'images 27
Modèle de segmentation d'image 27
Créez votre modèle 29
Création d’un projet 29
Créer un jeu de données 29
Entraînement de votre modèle 31
Évaluation de votre modèle 32
Utilisez votre modèle 32
Utiliser votre modèle sur un appareil Edge 33
Utilisez votre tableau de bord 33

Premiers pas 34
Étape 1 : créer le fichier manifeste et télécharger des images 36
Étape 2 : Création du modèle 37
Étape 3 : démarrer le modèle 45
Étape 4 : Analyser une image 47
Étape 5 : Arrêter le modèle 52
Étapes suivantes 54
Création de votre modèle 55
Création de votre projet 55
Création d'un projet (console) 56
Création d'un projet (kit SDK) 56
Création de votre jeu de données 58
Préparation d'images pour un jeu de données 59
Création du jeu de données 61
Ordinateur local 62
Compartiment Amazon S3 64
Fichier manifeste 67
Étiquetage des images 96
Choix du type de modèle 96
Classification des images (console) 97
Segmentation d'images (console) 98
Entraînement de votre modèle 102
Entraînement d'un modèle (console) 103
Entraînement d'un modèle (kit SDK) 104
Modèle de résolution des problèmes 110
Les couleurs des étiquettes d'anomalies ne correspondent pas à la couleur des anomalies dans l'image du masque 110
Les images du masque ne sont pas au format PNG 112
Les étiquettes de segmentation ou de classification sont inexactes ou manquantes 113
Amélioration de votre modèle 115
Étape 1 : Évaluez les performances de votre modèle 115
Métriques de classification des images 115
Métriques du modèle de segmentation d'images 116
Précision 116
Rappel 117
Score de F1 118

Intersection moyenne au-dessus de l'Union (IoU) 118
Résultats des tests 119
Étape 2 : Améliorez votre modèle 119
Consultation des métriques de performances 121
Affichage des indicateurs de performance (console) 121
Affichage des indicateurs de performance (SDK) 123
Vérification de votre modèle 127
Exécution d'une tâche de détection d'essai 128
Vérification des résultats de détection des essais 129
Corriger les étiquettes de segmentation avec l'outil d'annotation 130
Exécution de votre modèle 133
Unités d’inférence 133
Gestion du débit à l’aide d’unités d’inférence 134
Zones de disponibilité 136
Démarrage de votre modèle 136
Démarrage de votre modèle (console) 137
Démarrage de votre modèle (SDK) 138
Arrêter votre modèle 143
Arrêt de votre modèle (console) 144
Arrêter votre modèle (SDK) 145
Détecter des anomalies dans une image 150
Appel DetectAnomalies 150
Comprendre la réponse de DetectAnomalies 154
Modèle de classification 154
Modèle de segmentation 155
Déterminer si une image est anormale 157
Classification 157
Segmentation 159
Affichage des informations de classification et de segmentation 164
Trouver des anomalies avec une AWS Lambda fonction 179
Étape 1 : Création d'une AWS Lambda fonction (console) 179
Étape 2 : (Facultatif) Créer une couche (console) 182
Étape 3 : Ajouter le code Python (console) 183
Étape 4 : Essayer votre fonction Lambda 187
Utilisation de votre modèle sur un appareil Edge 192
Déploiement d'un modèle sur un appareil principal 194

Exigences de base relatives à l'appareil 195
Appareils, architectures de puces et systèmes d'exploitation testés 195
Mémoire et stockage de l'appareil principal 197
Logiciel requis 197
Configuration de votre appareil principal 198
Configuration de votre appareil principal 199
Emballage de votre modèle 200
Réglages du package 201
Emballage de votre modèle (console) 203
Emballage de votre modèle (SDK) 204
Obtenir des informations sur les travaux de maquette d'emballage 208
Écrire le composant de votre application client 210
Configuration de votre environnement 211
Utilisation d'un modèle 213
Création du composant d'application client 218
Déploiement de vos composants sur un appareil 223
Autorisations IAM pour le déploiement de composants 224
Déploiement de vos composants (console) 224
Déploiement des composants (SDK) 226
Référence de l'API Lookout for Vision Edge Agent 228
Détecter les anomalies à l'aide d'un modèle 228
Obtenir des informations sur le modèle 228
Exécution d'un modèle 228
DetectAnomalies 229
DescribeModel 235
ListModels 237
StartModel 238
StopModel 240
ModelStatus 241
Utilisation du tableau de bord 243
Gérer vos ressources 246
Afficher vos projets 246
Afficher vos projets (console) 247
Afficher vos projets (SDK) 247
Suppression d’un projet 250
Supprimer un projet (console) 250

Suppression d’un projet (kit SDK) 251
Afficher vos ensembles de données 253
Afficher les ensembles de données d'un projet (console) 253
Afficher les ensembles de données d'un projet (SDK) 253
Ajouter des images à votre jeu de données 256
Ajout d’autres images 257
Ajout d’autres images (kit SDK) 257
Supprimer des images de votre jeu de données 263
Supprimer des images d'un jeu de données (console) 263
Supprimer des images d'un jeu de données (SDK) 265
Suppression d’un jeu de données 265
Supprimer un ensemble de données (console) 253
Suppression d’un jeu de données (kit SDK) 266
Exportation de jeux de données depuis un projet (SDK) 268
Visualisation de vos modèles 278
Affichage de vos modèles (console) 278
Affichage de vos modèles (SDK) 278
Suppression d’un modèle 281
Supprimer un modèle (console) 281
Suppression d’un modèle (kit SDK) 282
Modèles de balisage 285
Balisage de modèles (console) 286
Balisage de modèles (kit SDK) 287
Afficher les tâches de détection de vos essais 289
Afficher les tâches de détection de votre essai (console) 289
Exemples de code et de jeux de données 291
Exemple de code 291
Exemples de jeux de données 291
Ensembles de données de segmentation d'images 292
Ensemble de données de classification d'images 292
Sécurité 295
Protection des données 296
Chiffrement des données 297
Confidentialité du trafic inter-réseau 298
Gestion des identités et des accès 298
Public ciblé 299

Authentification par des identités 300
Gestion des accès à l’aide de politiques 304
Comment Amazon Lookout for Vision fonctionne avec IAM 307
Exemples de politiques basées sur l’identité 314
Politiques gérées par AWS 317
Résolution des problèmes 330
Validation de conformité 332
Résilience 333
Sécurité de l’infrastructure 333
Surveillance 335
Surveillance avec CloudWatch 336
CloudTrail journaux 338
Informations sur Lookout for Vision dans CloudTrail 339
Comprendre les entrées du fichier journal Lookout for Vision 340
AWS CloudFormation ressources 343
Lookout for Vision et modèles AWS CloudFormation 343
En savoir plus sur AWS CloudFormation 343
AWS PrivateLink 345
Considérations relatives aux points de terminaison VPC Lookout for Vision 345
Création d'un point de terminaison VPC d'interface pour Lookout for Vision 345
Création d'une politique de point de terminaison VPC pour Lookout for Vision 346
Quotas 348
Quotas de modèle 348
Historique de la documentation 351
AWS Glossaire 357

Avis de fin de support : le 31 octobre 2025, le support d'Amazon Lookout for Vision AWS sera interrompu. Après le 31 octobre 2025, vous ne pourrez plus accéder à la console Lookout for Vision ni aux ressources Lookout for Vision. Pour plus d'informations, consultez ce billet de blog.

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.

Qu'est-ce qu'Amazon Lookout for Vision ?

Vous pouvez utiliser Amazon Lookout for Vision pour détecter les défauts visuels des produits industriels, avec précision et à grande échelle. Il utilise la vision par ordinateur pour identifier les composants manquants dans un produit industriel, les dommages aux véhicules ou aux structures, les irrégularités dans les chaînes de production et même les défauts infimes dans les plaquettes de silicium, ou tout autre élément physique dont la qualité est importante, comme un condensateur manquant sur des cartes de circuit imprimé.

Principaux avantages

Amazon Lookout for Vision offre les avantages suivants :

- Améliorez les processus rapidement et efficacement : vous pouvez utiliser Amazon Lookout for Vision pour implémenter l'inspection basée sur la vision par ordinateur dans les processus industriels de manière rapide et efficace, à grande échelle. Vous pouvez fournir seulement 30 bonnes images de référence et Lookout for Vision peut créer automatiquement un modèle d'apprentissage automatique personnalisé pour la détection des défauts. Vous pouvez ensuite traiter les images des caméras IP, par lots ou en temps réel, afin d'identifier rapidement et précisément les anomalies telles que les bosses, les fissures et les rayures.
- Améliorez rapidement la qualité de la production : avec Amazon Lookout for Vision, vous pouvez réduire les défauts dans les processus de production en temps réel. Il identifie et signale les anomalies visuelles dans un tableau de bord afin que vous puissiez agir rapidement pour empêcher l'apparition de nouveaux défauts, en augmentant la qualité de la production et en réduisant les coûts.
- Réduisez les coûts d'exploitation : Amazon Lookout for Vision indique les tendances relatives à vos données d'inspection visuelle, par exemple en identifiant les processus présentant le taux de défauts le plus élevé ou en signalant les variations récentes des défauts. À l'aide de ces informations, vous pouvez déterminer s'il convient de planifier la maintenance sur la chaîne de traitement ou de rediriger la production vers une autre machine avant que des temps d'arrêt coûteux et imprévus ne se produisent.

Êtes-vous un utilisateur final d'Amazon Lookout for Vision pour la première fois ?

Si vous utilisez Amazon Lookout for Vision pour la première fois, nous vous recommandons de lire les sections suivantes dans l'ordre :

1. Configuration d'Amazon Lookout for Vision : dans cette section, vous apprendrez à configurer votre compte.
2. Commencer à utiliser Amazon Lookout for Vision— Dans cette section, vous découvrirez comment créer votre premier modèle Amazon Lookout for Vision.

Configuration d'Amazon Lookout for Vision

Dans cette section, vous créez un compte AWS et configurez Amazon Lookout for Vision.

Pour plus d'informations sur les AWS régions qui prennent en charge Amazon Lookout for Vision, consultez Amazon Lookout for Vision Endpoints and Quotas.

Rubriques

- Étape 1 : créer un AWS compte
- Étape 2 : configurer les autorisations
- Étape 3 : Créer le compartiment de la console
- Étape 4 : Configurez le AWS CLI et AWS SDKs
- Étape 5 : (Facultatif) Utilisation de votre propre clé AWS Key Management Service

Étape 1 : créer un AWS compte

Au cours de cette étape, vous créez un AWS compte et créez un utilisateur administratif.

Rubriques

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

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.

   Dans le cadre de la procédure d’inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique 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 le gérer 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 .

Étape 2 : configurer les autorisations

Pour utiliser Amazon Lookout for Vision, vous devez disposer d'autorisations d'accès à la console Lookout for Vision AWS , aux opérations du SDK et au compartiment Amazon S3 que vous utilisez pour la formation des modèles.

> **Note**
>
> Si vous utilisez uniquement des opérations du AWS SDK, vous pouvez utiliser des politiques qui s'appliquent aux opérations du AWS SDK. Pour de plus amples informations, veuillez consulter Configuration des autorisations du kit SDK.

Rubriques

- Configuration de l'accès à la console avec des politiques AWS gérées

- Configuration des autorisations du compartiment Amazon S3
- Attribution d'autorisations

Configuration de l'accès à la console avec des politiques AWS gérées

Utilisez les politiques AWS gérées suivantes pour appliquer les autorisations d'accès appropriées aux opérations du SDK et de la console Amazon Lookout for Vision.

- AmazonLookoutVisionConsoleFullAccess— permet un accès complet à la console Amazon Lookout for Vision et aux opérations du SDK. Vous avez besoin `AmazonLookoutVisionConsoleFullAccess` d'autorisations pour créer le bucket de console. Pour de plus amples informations, veuillez consulter Étape 3 : Créer le compartiment de la console.
- AmazonLookoutVisionConsoleReadOnlyAccess— permet un accès en lecture seule à la console Amazon Lookout for Vision et aux opérations du SDK.

Pour attribuer des autorisations, consultez Attribution d'autorisations.

Pour plus d'informations sur les politiques AWS gérées, consultez la section Politiques gérées par AWS.

Configuration des autorisations du compartiment Amazon S3

Amazon Lookout for Vision utilise un compartiment Amazon S3 pour stocker les fichiers suivants :

- Images du jeu de données : images utilisées pour entraîner un modèle. Pour de plus amples informations, veuillez consulter Création de votre jeu de données.
- Fichiers manifestes au format Amazon SageMaker AI Ground Truth. Par exemple, la sortie du fichier manifeste d'une GroundTruth tâche SageMaker AI. Pour de plus amples informations, veuillez consulter Création d'un ensemble de données à l'aide d'un fichier manifeste Amazon SageMaker AI Ground Truth.
- Le résultat de l'entraînement des modèles.

Si vous utilisez la console, Lookout for Vision crée un compartiment Amazon S3 (compartiment console) pour gérer vos projets. Les `LookoutVisionConsoleReadOnlyAccess` politiques `LookoutVisionConsoleFullAccess` gérées incluent les autorisations d'accès Amazon S3 pour le compartiment de console.

Vous pouvez utiliser le bucket de console pour stocker des images de jeux de données et des fichiers manifestes au format SageMaker AI Ground Truth. Vous pouvez également utiliser un autre compartiment Amazon S3. Le bucket doit appartenir à votre compte AWS et doit être situé dans la AWS région dans laquelle vous utilisez Lookout for Vision.

Pour utiliser un autre bucket, ajoutez la politique suivante à l'utilisateur ou au groupe souhaité. `amzn-s3-demo-bucket`Remplacez-le par le nom du compartiment souhaité. Pour plus d'informations sur l'ajout de politiques IAM, consultez la section Création de politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionS3BucketAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        },
        {
            "Sid": "LookoutVisionS3ObjectAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

Pour attribuer des autorisations, consultez Attribution d'autorisations.

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

Pour utiliser la console Amazon Lookout for Vision, vous avez besoin d'un compartiment Amazon S3 connu sous le nom de compartiment de console. Le bucket de console contient les éléments suivants :

- Images que vous chargez dans un ensemble de données à l'aide de la console.
- Résultats d'entraînement pour les modèles d'entraînement que vous commencez avec la console.
- Résultats de détection des essais.
- Fichiers manifestes temporaires créés par la console lorsque vous utilisez la console pour créer un ensemble de données en étiquetant automatiquement les images dans un compartiment S3. La console ne supprime pas les fichiers manifestes.

Lorsque vous ouvrez la console Amazon Lookout for Vision pour la première fois dans une AWS nouvelle région, Lookout for Vision crée le bucket de console en votre nom. Notez le nom du compartiment de console, 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.

Vous pouvez également créer le compartiment de console à l'aide d'Amazon S3. Utilisez cette approche si les politiques du compartiment Amazon S3 ne permettent pas à la console Amazon Lookout for Vision de créer correctement le compartiment de console. Par exemple, une politique qui interdit la création automatique d'un compartiment Amazon S3.

> **Note**
>
> Si vous utilisez uniquement le AWS SDK et non la console Lookout for Vision, il n'est pas nécessaire de créer le bucket de console. Vous pouvez utiliser un autre compartiment S3 portant le nom de votre choix.

Le format du nom du bucket de console est `lookoutvision-<region>-<random value>`. La valeur aléatoire garantit l'absence de conflit entre les noms de compartiments.

Rubriques

- Création du bucket de console avec la console Amazon Lookout for Vision
- Création du compartiment de console avec Amazon S3
- Paramètres du bucket de console

Création du bucket de console avec la console Amazon Lookout for Vision

Utilisez la procédure suivante pour créer le bucket de console pour une AWS région à l'aide de la console Amazon Lookout for Vision. Pour plus d'informations sur les paramètres du compartiment S3 que nous activons, consultezParamètres du bucket de console.

Pour créer le bucket de console à l'aide de la console Amazon Lookout for Vision

1. Assurez-vous que l'utilisateur ou le groupe que vous utilisez est `AmazonLookoutVisionConsoleFullAccess` autorisé. Pour de plus amples informations, veuillez consulter Étape 2 : configurer les autorisations.
2. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
3. Dans la barre de navigation, choisissez Sélectionner une région. Choisissez ensuite la AWS région pour laquelle vous souhaitez créer le bucket de console.
4. Choisissez Démarrer.

5. Si c'est la première fois que vous ouvrez la console dans la région AWS actuelle, 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 Create S3 bucket pour permettre à Amazon Lookout for Vision de créer le bucket de console en votre nom.

   La boîte de dialogue Première configuration n'apparaît pas si le bucket de console pour la AWS région actuelle existe déjà.

6. Fermez la fenêtre du navigateur.

Création du compartiment de console avec Amazon S3

Vous pouvez utiliser Amazon S3 pour créer le compartiment de console. Vous devez créer le compartiment avec le versionnement d'Amazon S3 activé. Nous vous recommandons d'utiliser une configuration du cycle de vie Amazon S3 pour supprimer les versions non actuelles (précédentes) d'un objet et supprimer les téléchargements partitionnés incomplets. Nous ne recommandons pas une configuration du cycle de vie qui supprime les versions actuelles d'un objet. Pour plus d'informations sur les paramètres des compartiments S3 que nous activons pour les compartiments de console que vous créez avec la console Amazon Lookout for Vision, consultez. Paramètres du bucket de console

1. Choisissez la AWS région dans laquelle vous souhaitez créer un bucket de console. Pour plus d'informations sur les régions prises en charge, consultez Amazon Lookout for Vision pour les points de terminaison et les quotas.
2. Créez un compartiment en suivant les instructions de la console S3 dans Création d'un compartiment. Procédez comme suit :

   a. Pour l'étape 3, spécifiez un nom de compartiment précédé de. `lookoutvision-`*`region`*`-`*`your-identifier` Modifiez `region` le code de région que vous avez choisi à l'étape précédente. Optez `your-identifier` pour un identifiant unique de votre choix.

   b. Pour l'étape 4, choisissez la AWS région que vous souhaitez utiliser.

3. Activez le contrôle de version pour le compartiment en suivant les instructions de la console S3 à la section Activation du contrôle de version sur les compartiments.

4. (Facultatif) Spécifiez une configuration du cycle de vie pour le compartiment en suivant les instructions de la console S3 dans Configuration de la configuration du cycle de vie d'un compartiment. Procédez comme suit pour supprimer les versions non actuelles (précédentes) d'un objet et supprimer les téléchargements partitionnés incomplets. Vous n'avez pas besoin de suivre les étapes 6, 8, 9, 10.

   a. Pour l'étape 5, choisissez Appliquer à tous les objets du compartiment.

   b. Pour l'étape 7, sélectionnez Supprimer définitivement les versions non actuelles des objets et Supprimer les objets expirés, supprimer les marqueurs ou les chargements partitionnés incomplets.

   c. Pour l'étape 11, entrez le nombre de jours à attendre avant de supprimer les versions non actuelles d'un objet.

   d. Pour l'étape 12, entrez le nombre de jours à attendre avant de supprimer les téléchargements partitionnés incomplets.

Paramètres du bucket de console

Si vous créez le bucket de console avec la console Amazon Lookout for Vision, nous activons les paramètres suivants sur le bucket de console.

- Versionnage des objets dans le bucket de console.
- Chiffrement côté serveur des objets du compartiment de console.
- Configuration du cycle de vie pour la suppression des objets non courants (30 jours) et des téléchargements partitionnés incomplets (3 jours).
- Bloquez l'accès public au bucket de console.

Étape 4 : Configurez le AWS CLI et AWS SDKs

Les étapes suivantes vous montrent comment installer les AWS Command Line Interface (AWS CLI) et AWS SDKs. Les exemples présentés dans cette documentation utilisent Python et Java AWS SDKs. AWS CLI

Rubriques

- Installez les AWS SDK
- Octroi d’un accès par programmation

- Configuration des autorisations du kit SDK
- Appelez une opération Amazon Lookout for Vision

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 `lookoutvision-access`. Par exemple :

```
session = boto3.Session(profile_name='lookoutvision-access')
lookoutvision_client = session.client("lookoutvision")
```

L'utilisateur représenté par le profil doit être autorisé à appeler les opérations du SDK Lookout for Vision et les AWS 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 Attribution d'autorisations.

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 `lookoutvision-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.

- Appareil principal Greengrass : utilisez des certificats X.509 pour vous connecter à AWS IoT Core à l'aide des protocoles d'authentification mutuelle TLS. Ces certificats permettent aux appareils d'interagir avec AWS IoT sans informations d'identification AWS. Le fournisseur d'informations d'identification AWS IoT authentifie les appareils à l'aide du certificat X.509 et émet des informations d'identification AWS sous la forme d'un jeton de sécurité temporaire à privilèges limités. 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 SDK Amazon Lookout for Vision, vous devez disposer d'autorisations d'accès à l'API Lookout for Vision et au compartiment Amazon S3 utilisé pour la formation des modèles.

Rubriques

- Octroi de droits d'utilisation du kit SDK
- Octroi d'autorisations pour Amazon S3 Bucket
- 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 DetectAnomalies, vous avez besoin d'une autorisation pour effectuer un appel`lookoutvision:DetectAnomalies`. 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.

- AmazonLookoutVisionFullAccess— permet un accès complet aux opérations du SDK Amazon Lookout for Vision.

- AmazonLookoutVisionReadOnlyAccess— permet d'accéder aux opérations du SDK en lecture seule.

Les politiques gérées pour la console fournissent également des autorisations d'accès pour les opérations du SDK. Pour de plus amples informations, veuillez consulter Étape 2 : configurer les autorisations.

Pour plus d'informations sur les politiques AWS gérées, consultez la section Politiques gérées par AWS.

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.

> **Note**
>
> Les instructions de démarrage nécessitent `s3:PutObject` des autorisations. Pour de plus amples informations, veuillez consulter Étape 1 : créer le fichier manifeste et télécharger des images.

Pour attribuer des autorisations, consultez Attribution d'autorisations.

Octroi d'autorisations pour Amazon S3 Bucket

Pour entraîner un modèle, vous avez besoin d'un compartiment Amazon S3 doté des autorisations appropriées pour stocker les images, les fichiers manifestes et les résultats de formation. Le compartiment doit appartenir à votre compte AWS et doit être situé dans la région AWS dans laquelle vous utilisez Amazon Lookout for Vision.

Les politiques gérées uniquement par le SDK (`AmazonLookoutVisionFullAccessetAmazonLookoutVisionReadOnlyAccess`) n'incluent pas les autorisations de compartiment Amazon S3 et vous devez appliquer la politique d'autorisation suivante pour accéder aux compartiments que vous utilisez, y compris les compartiments de console existants.

Les politiques gérées par la console (`AmazonLookoutVisionConsoleFullAccessetAmazonLookoutVisionConsoleReadOnlyAccess`)

incluent les autorisations d'accès au compartiment de console. Si vous accédez au compartiment de console via des opérations du SDK et que vous disposez d'autorisations de politique gérées par la console, vous n'avez pas besoin d'utiliser la politique suivante. Pour de plus amples informations, veuillez consulter Étape 2 : configurer les autorisations.

Décider des autorisations relatives aux tâches

Utilisez les informations suivantes pour déterminer les autorisations nécessaires pour les tâches que vous souhaitez effectuer.

Création d'un jeu de données

Pour créer un ensemble de données avec CreateDataset, vous devez disposer des autorisations suivantes.

- `s3:GetBucketLocation`— permet à Lookout for Vision de vérifier que votre bucket se trouve dans la même région que celle dans laquelle vous utilisez Lookout for Vision.
- `s3:GetObject`— Permet d'accéder au fichier manifeste spécifié dans le paramètre `DatasetSource` d'entrée. Si vous souhaitez spécifier une version d'objet S3 exacte du fichier manifeste, vous devez également `s3:GetObjectVersion` utiliser le fichier manifeste. Pour plus d'informations, consultez la section Utilisation de la gestion des versions dans les compartiments S3.

Création d’un modèle

Pour créer un modèle avec CreateModel, vous devez disposer des autorisations suivantes.

- `s3:GetBucketLocation`— permet à Lookout for Vision de vérifier que votre bucket se trouve dans la même région que celle dans laquelle vous utilisez Lookout for Vision.
- `s3:GetObject`— permet d'accéder aux images spécifiées dans les ensembles de données d'entraînement et de test du projet.
- `s3:PutObject`— autorise le stockage des résultats d'entraînement dans le compartiment spécifié. Vous spécifiez l'emplacement du compartiment de sortie dans le `OutputConfig` paramètre. Vous pouvez éventuellement limiter les autorisations aux seules clés d'objet spécifiées dans le `Prefix` champ du champ de `S3Location` saisie. Pour de plus amples informations, veuillez consulter OutputConfig.

Accès aux images, aux fichiers manifestes et aux résultats de formation

Les autorisations du compartiment Amazon S3 ne sont pas requises pour consulter les réponses aux opérations Amazon Lookout for Vision. Vous avez besoin d'une `s3:GetObject` autorisation pour accéder aux images, aux fichiers manifestes et aux résultats de formation référencés dans les réponses aux opérations. Si vous accédez à un objet Amazon S3 versionné, vous devez disposer d'une `s3:GetObjectVersion` autorisation.

Configuration de la politique relative aux compartiments Amazon S3

Vous pouvez utiliser la politique suivante pour spécifier les autorisations du compartiment Amazon S3 nécessaires pour créer un ensemble de données (`CreateDataset`), créer un modèle (`CreateModel`) et accéder aux images, aux fichiers manifestes et aux résultats de formation. Remplacez la valeur *`amzn-s3-demo-bucket`* de par le nom du compartiment que vous souhaitez utiliser.

Vous pouvez adapter la politique à vos besoins. Pour de plus amples informations, veuillez consulter Décider des autorisations relatives aux tâches. Ajoutez la politique à l'utilisateur de votre choix. Pour plus d'informations, consultez la section Création de politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionS3BucketAccess",
            "Effect": "Allow",
            "Action": "s3:GetBucketLocation",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ],
            "Condition": {
                "Bool": {
                    "aws:ViaAWSService": "true"
                }
            }
        },
        {
            "Sid": "LookoutVisionS3ObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
```

```
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ],
            "Condition": {
                "Bool": {
                    "aws:ViaAWSService": "true"
                }
            }
        }
    ]
}
```

Pour attribuer des autorisations, consultez Attribution d'autorisations.

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.

Appelez une opération Amazon Lookout for Vision

Exécutez le code suivant pour confirmer que vous pouvez passer des appels vers l'API Amazon Lookout for Vision. 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 `ListProjects`.

En général, l'appel d'une fonction d'exemple nécessite un client AWS SDK Lookout for Vision et tous les autres paramètres requis. Le client Lookout for Vision du SDK AWS 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 Amazon Lookout for Vision n'est pas disponible dans AWS toutes les régions.

Pour appeler une opération Amazon Lookout for Vision

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 `list-projects` pour répertorier les projets de votre compte.

   ```
   aws lookoutvision list-projects \
   --profile lookoutvision-access
   ```

   Python

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

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


   from botocore.exceptions import ClientError
   import boto3

   class GettingStarted:

       @staticmethod
   ```

```
    def list_projects(lookoutvision_client):
        """
        Lists information about the projects that are in in your AWS account
        and in the current AWS Region.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        """
        try:
            response = lookoutvision_client.list_projects()
            for project in response["Projects"]:
                print("Project: " + project["ProjectName"])
                print("ARN: " + project["ProjectArn"])
                print()
            print("Done!")
        except ClientError:
            raise
def main():
    session = boto3.Session(profile_name='lookoutvision-access')
    lookoutvision_client = session.client("lookoutvision")

    GettingStarted.list_projects(lookoutvision_client)

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

Java V2

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

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

package com.example.lookoutvision;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.lookoutvision.LookoutVisionClient;
import software.amazon.awssdk.services.lookoutvision.model.ProjectMetadata;
import
 software.amazon.awssdk.services.lookoutvision.paginators.ListProjectsIterable;
```

```
import software.amazon.awssdk.services.lookoutvision.model.ListProjectsRequest;
import
 software.amazon.awssdk.services.lookoutvision.model.LookoutVisionException;

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

public class GettingStarted {

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

    /**
     * Lists the Amazon Lookoutfor Vision projects in the current AWS account
 and
     * AWS Region.
     *
     * @param lfvClient   An Amazon Lookout for Vision client.
     * @return List<ProjectMetadata> Metadata for each project.
     */
    public static List<ProjectMetadata> listProjects(LookoutVisionClient
 lfvClient)
            throws LookoutVisionException {

        logger.log(Level.INFO, "Getting projects:");
        ListProjectsRequest listProjectsRequest = ListProjectsRequest.builder()
                .maxResults(100)
                .build();

        List<ProjectMetadata> projectMetadata = new ArrayList<>();

        ListProjectsIterable projects =
 lfvClient.listProjectsPaginator(listProjectsRequest);

        projects.stream().flatMap(r -> r.projects().stream())
                .forEach(project -> {
                    projectMetadata.add(project);
                    logger.log(Level.INFO, project.projectName());
                });

        logger.log(Level.INFO, "Finished getting projects.");
```

```
        return projectMetadata;

    }

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

        try {

            // Get the Lookout for Vision client.
            LookoutVisionClient lfvClient = LookoutVisionClient.builder()

 .credentialsProvider(ProfileCredentialsProvider.create("lookoutvision-access"))
                    .build();

            List<ProjectMetadata> projects = Projects.listProjects(lfvClient);

            System.out.printf("Projects%n--------%n");

            for (ProjectMetadata project : projects) {
                System.out.printf("Name: %s%n", project.projectName());
                System.out.printf("ARN: %s%n", project.projectArn());
                System.out.printf("Date: %s%n%n",
 project.creationTimestamp().toString());
            }

        } catch (LookoutVisionException lfvError) {
            logger.log(Level.SEVERE, "Could not list projects: {0}: {1}",
                    new Object[] { lfvError.awsErrorDetails().errorCode(),
                            lfvError.awsErrorDetails().errorMessage() });
            System.out.println(String.format("Could not list projects: %s",
 lfvError.getMessage()));
            System.exit(1);
        }

    }

}
```

Étape 5 : (Facultatif) Utilisation de votre propre clé AWS Key Management Service

Vous pouvez utiliser AWS Key Management Service (KMS) pour gérer le chiffrement des images d'entrée que vous stockez dans des compartiments Amazon S3.

Par défaut, vos images sont chiffrées à l'aide d'une clé détenue et gérée par AWS. Vous pouvez également choisir d'utiliser votre propre clé AWS Key Management Service (KMS). Pour plus d'informations, consultez Concepts d'AWS Key Management Service.

Si vous souhaitez utiliser votre propre clé KMS, utilisez la politique suivante pour spécifier la clé KMS. Passez *kms_key_arn* à l'ARN de la clé KMS (ou ARN alias KMS) que vous souhaitez utiliser. Vous pouvez également spécifier * d'utiliser n'importe quelle clé KMS. Pour plus d'informations sur l'ajout de la politique à un utilisateur ou à un rôle, consultez la section Création de politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionKmsDescribeAccess",
            "Effect": "Allow",
            "Action": "kms:DescribeKey",
            "Resource": "kms_key_arn"
        },
        {
            "Sid": "LookoutVisionKmsCreateGrantAccess",
            "Effect": "Allow",
            "Action": "kms:CreateGrant",
            "Resource": "kms_key_arn",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": "lookoutvision.*.amazonaws.com"
                },
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
    ]
}
```

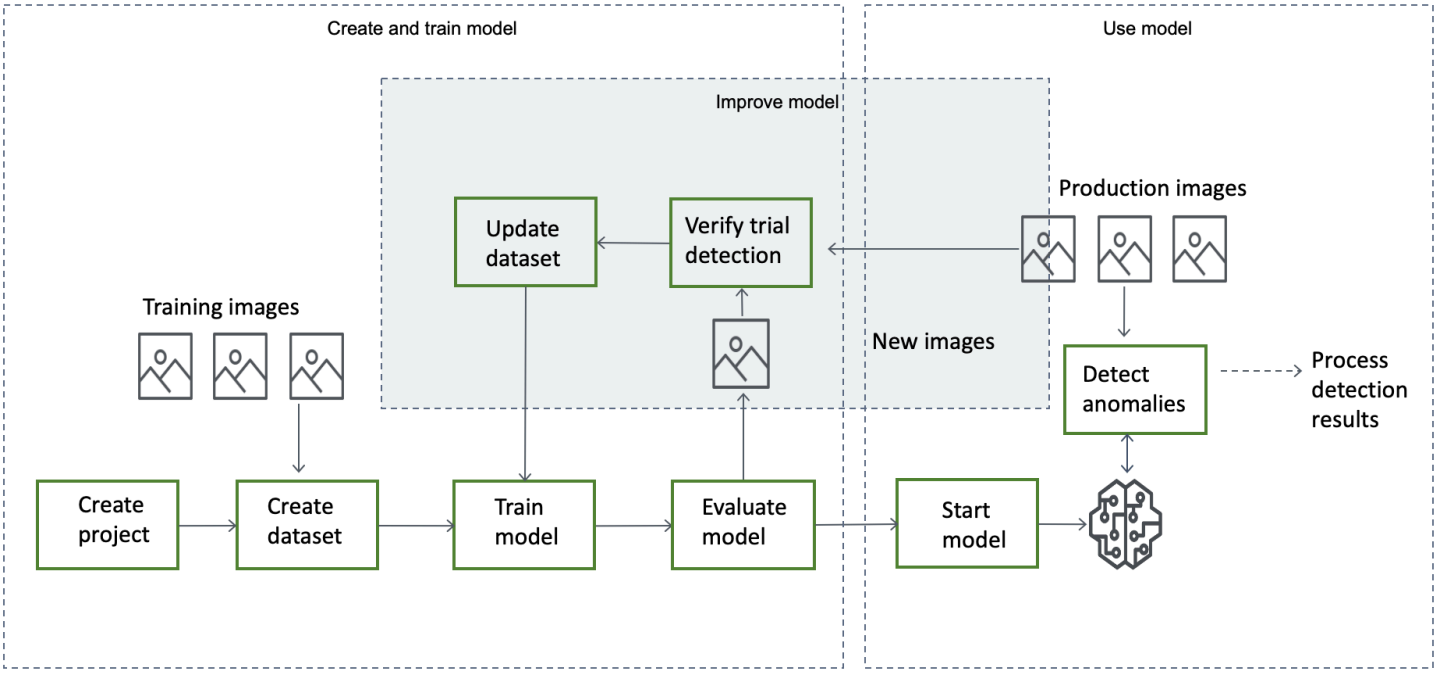
Comprendre Amazon Lookout for Vision

Vous pouvez utiliser Amazon Lookout for Vision pour détecter les défauts visuels des produits industriels, avec précision et à grande échelle, pour des tâches telles que :

- Détection des pièces endommagées : repérez les dommages causés à la qualité de surface, à la couleur et à la forme d'un produit pendant le processus de fabrication et d'assemblage.
- Identification des composants manquants : déterminez les composants manquants en fonction de l'absence, de la présence ou de l'emplacement des objets. Par exemple, un condensateur manquant sur une carte de circuit imprimé.
- Découverte des problèmes liés au processus — Détectez les défauts présentant des motifs répétitifs, tels que des rayures répétées au même endroit sur une plaquette de silicone.

Lookout for Vision vous permet de créer un modèle de vision par ordinateur qui prédit la présence d'anomalies dans une image. Vous fournissez les images qu'Amazon Lookout for Vision utilise pour entraîner et tester votre modèle. Amazon Lookout for Vision fournit des indicateurs que vous pouvez utiliser pour évaluer et améliorer votre modèle entraîné. Vous pouvez héberger le modèle entraîné dans le AWS cloud ou le déployer sur un appareil périphérique. Une simple opération d'API renvoie les prédictions effectuées par votre modèle.

Le flux de travail général pour créer, évaluer et utiliser un modèle est le suivant :



Rubriques

- Choisissez votre type de modèle
- Créez votre modèle
- Évaluation de votre modèle
- Utilisez votre modèle
- Utiliser votre modèle sur un appareil Edge
- Utilisez votre tableau de bord

Choisissez votre type de modèle

Avant de créer un modèle, vous devez choisir le type de modèle que vous souhaitez. Vous pouvez créer deux types de modèles : la classification des images et la segmentation des images. Vous décidez du type de modèle à créer en fonction de votre cas d'utilisation.

Modèle de classification d'images

Si vous avez uniquement besoin de savoir si une image contient une anomalie, mais pas de connaître son emplacement, créez un modèle de classification d'images. Un modèle de classification d'images permet de prédire si une image contient une anomalie. La prédiction inclut la confiance du modèle dans la précision de la prédiction. Le modèle ne fournit aucune information sur l'emplacement des anomalies détectées sur l'image.

Modèle de segmentation d'image

Si vous avez besoin de connaître l'emplacement d'une anomalie, par exemple l'emplacement d'une rayure, créez un modèle de segmentation d'image. Les modèles Amazon Lookout for Vision utilisent la segmentation sémantique pour identifier les pixels d'une image présentant les types d'anomalies (comme une rayure ou une partie manquante).

> **Note**
>
> Un modèle de segmentation sémantique permet de localiser différents types d'anomalies. Il ne fournit pas d'informations d'instance pour les anomalies individuelles. Par exemple, si une image contient deux bosses, Lookout for Vision renvoie des informations sur les deux bosses dans une seule entité représentant le type d'anomalie de bosse.

Un modèle de segmentation Amazon Lookout for Vision prédit ce qui suit :

Classification

Le modèle renvoie une classification pour une image analysée (normale/anomalie), qui inclut la confiance du modèle dans la prédiction. Les informations de classification sont calculées séparément des informations de segmentation et vous ne devez pas supposer qu'il existe une relation entre elles.

Segmentation

Le modèle renvoie un masque d'image qui marque les pixels où des anomalies apparaissent sur l'image. Les différents types d'anomalies sont codés par couleur en fonction de la couleur attribuée à l'étiquette d'anomalie dans le jeu de données. Une étiquette d'anomalie représente le type d'anomalie. Par exemple, le masque bleu de l'image suivante indique l'emplacement d'une anomalie de rayure détectée sur une voiture.

Le modèle renvoie le code couleur pour chaque étiquette d'anomalie du masque. Le modèle renvoie également le pourcentage de couverture de l'image que possède une étiquette d'anomalie.

Avec un modèle de segmentation Lookout for Vision, vous pouvez utiliser différents critères pour analyser les résultats d'analyse du modèle. Par exemple :

- Localisation des anomalies : si vous avez besoin de connaître l'emplacement des anomalies, utilisez les informations de segmentation pour voir les masques qui masquent les anomalies.
- Types d'anomalies : utilisez les informations de segmentation pour déterminer si une image contient plus qu'un nombre acceptable de types d'anomalies.
- Zone de couverture : utilisez les informations de segmentation pour déterminer si un type d'anomalie couvre plus qu'une zone acceptable d'une image.
- Classification des images : si vous n'avez pas besoin de connaître l'emplacement des anomalies, utilisez les informations de classification pour déterminer si une image contient des anomalies.

Pour obtenir un exemple de code, consultez Détecter des anomalies dans une image.

Après avoir choisi le type de modèle que vous souhaitez, vous créez un projet et un jeu de données pour gérer votre modèle. À l'aide des étiquettes, vous pouvez classer les images comme normales ou comme anormales. Les étiquettes identifient également les informations de segmentation telles que les masques et les types d'anomalies. La façon dont vous étiquetez les images de votre jeu de données détermine le type de modèle que Lookout for Vision crée pour vous.

L'étiquetage d'un modèle de segmentation d'image est plus complexe que l'étiquetage d'un modèle de classification d'images. Pour entraîner un modèle de segmentation, vous devez classer les images d'entraînement comme normales ou anormales. Vous devez également définir des masques d'anomalies et des types d'anomalies pour chaque image anormale. Un modèle de classification vous oblige uniquement à identifier les images d'entraînement comme normales ou anormales.

Créez votre modèle

Les étapes de création d'un modèle, à savoir la création d'un projet, la création d'un jeu de données et l'entraînement du modèle, sont les suivantes :

Création d’un projet

Créez un projet pour gérer les ensembles de données et les modèles que vous créez. Un projet doit être utilisé pour un seul cas d'utilisation, tel que la détection d'anomalies dans un seul type de pièce de machine.

Vous pouvez utiliser le tableau de bord pour avoir une vue d'ensemble de vos projets. Pour de plus amples informations, veuillez consulter Utilisation du tableau de bord Amazon Lookout for Vision.

Pour plus d'informations : Créez votre projet.

Créer un jeu de données

Pour entraîner un modèle, Amazon Lookout for Vision a besoin d'images d'objets normaux et anormaux pour votre cas d'utilisation. Vous fournissez ces images dans un jeu de données.

Un jeu de données est un ensemble d’images et d’étiquettes qui décrivent ces images. Les images doivent représenter un seul type d'objet sur lequel des anomalies peuvent apparaître. Pour de plus amples informations, veuillez consulter Préparation d'images pour un jeu de données.

Avec Amazon Lookout for Vision, vous pouvez avoir un projet utilisant un seul ensemble de données ou un projet comportant des ensembles de données d'entraînement et de test distincts. Nous vous

recommandons d'utiliser un seul projet de jeu de données, sauf si vous souhaitez mieux contrôler la formation, les tests et le réglage des performances.

Vous créez un jeu de données en important les images. Selon la manière dont vous importez les images, celles-ci peuvent également être étiquetées. Dans le cas contraire, utilisez la console pour étiqueter les images.

Importation d'images

Si vous créez le jeu de données à l'aide de la console Lookout for Vision, vous pouvez importer les images de l'une des manières suivantes :

- Importez des images depuis votre ordinateur local. Les images ne sont pas étiquetées.
- Importez des images depuis un compartiment S3. Amazon Lookout for Vision peut classer les images à l'aide des noms de dossiers qui les contiennent. À utiliser `normal` pour des images normales. À utiliser `anomaly` pour les images anormales. Vous ne pouvez pas attribuer automatiquement des étiquettes de segmentation.
- Importez un fichier manifeste Amazon SageMaker AI Ground Truth. Les images d'un fichier manifeste sont étiquetées. Vous pouvez créer et importer votre propre fichier manifeste. Si vous avez de nombreuses images, pensez à utiliser le service d'étiquetage SageMaker AI Ground Truth. Vous importez ensuite le fichier manifeste de sortie depuis le job Amazon SageMaker AI Ground Truth.

Étiquetage des images

Les étiquettes décrivent une image dans un ensemble de données. Les étiquettes indiquent si une image est normale ou anormale (classification). Les étiquettes décrivent également l'emplacement des anomalies sur une image (segmentation).

Si vos images ne sont pas étiquetées, vous pouvez utiliser la console pour les étiqueter.

Les étiquettes que vous attribuez aux images de votre jeu de données déterminent le type de modèle créé par Lookout for Vision :

Classification d’images

Pour créer un modèle de classification d'images, utilisez la console Lookout for Vision pour classer les images du jeu de données comme normales ou comme anormales.

Vous pouvez également utiliser cette `CreateDataset` opération pour créer un ensemble de données à partir d'un fichier manifeste contenant des informations de classification.

Segmentation d'images

Pour créer un modèle de segmentation d'images, utilisez la console Lookout for Vision pour classer les images du jeu de données comme normales ou comme anormales. Vous spécifiez également des masques de pixels pour les zones anormales de l'image (s'ils existent) ainsi qu'une étiquette d'anomalie pour les masques d'anomalie individuels.

Vous pouvez également utiliser cette `CreateDataset` opération pour créer un ensemble de données à partir d'un fichier manifeste contenant des informations de segmentation et de classification.

Si votre projet comporte des ensembles de données d'entraînement et de test distincts, Lookout for Vision utilise le jeu de données d'entraînement pour apprendre et déterminer le type de modèle. Vous devez étiqueter les images de votre jeu de données de test de la même manière.

Pour plus d'informations : Création de votre jeu de données.

Entraînement de votre modèle

L'entraînement crée un modèle et l'entraîne à prévoir la présence d'anomalies dans les images. Une nouvelle version de votre modèle est créée à chaque entraînement.

Au début de la formation, Amazon Lookout for Vision choisit l'algorithme le plus approprié pour entraîner votre modèle. Le modèle est entraîné puis testé. Dans le Commencer à utiliser Amazon Lookout for Vision cas où vous entraînez un seul projet de jeu de données, le jeu de données est divisé en interne pour créer un ensemble de données d'entraînement et un ensemble de données de test. Vous pouvez également créer un projet comportant des ensembles de données d'entraînement et de test distincts. Dans cette configuration, Amazon Lookout for Vision entraîne votre modèle avec le jeu de données d'entraînement et teste le modèle avec le jeu de données de test.

> **Important**
>
> Le temps nécessaire pour entraîner avec succès votre modèle vous est facturé.

Pour plus d'informations : Entraînez votre modèle.

Évaluation de votre modèle

Évaluez les performances de votre modèle à l'aide des indicateurs de performance créés lors du test.

Grâce aux indicateurs de performance, vous pouvez mieux comprendre les performances de votre modèle entraîné et décider si vous êtes prêt à l'utiliser en production.

Pour plus d'informations : Améliorer votre modèle.

Si les indicateurs de performance indiquent que des améliorations sont nécessaires, vous pouvez ajouter des données d'entraînement supplémentaires en exécutant une tâche de détection d'essai avec de nouvelles images. Une fois la tâche terminée, vous pouvez vérifier les résultats et ajouter les images vérifiées à votre jeu de données d'entraînement. Vous pouvez également ajouter de nouvelles images d'entraînement directement dans le jeu de données. Ensuite, vous réentraînez votre modèle et vous revérifiez les indicateurs de performance.

Informations supplémentaires : Vérification de votre modèle à l'aide d'une tâche de détection d'essai.

Utilisez votre modèle

Avant de pouvoir utiliser votre modèle dans le AWS cloud, vous devez démarrer le modèle avec l'StartModelopération. Vous pouvez obtenir la commande `StartModel` CLI pour votre modèle depuis la console.

Pour plus d'informations : Démarrez votre modèle.

Un modèle Amazon Lookout for Vision expérimenté prédit si une image d'entrée contient un contenu normal ou anormal. Si votre modèle est un modèle de segmentation, la prédiction inclut un masque d'anomalie qui marque les pixels où les anomalies sont détectées.

Pour faire une prédiction avec votre modèle, appelez l'DetectAnomaliesopération et transmettez une image d'entrée depuis votre ordinateur local. Vous pouvez obtenir la commande CLI qui appelle `DetectAnomalies` depuis la console.

Pour plus d'informations : Détecter les anomalies dans une image.

> **Important**
>
> La durée de fonctionnement de votre modèle vous est facturée.

Si vous n'utilisez plus votre modèle, utilisez l'StopModelopération pour arrêter le modèle. Vous pouvez obtenir la commande CLI depuis la console.

Pour plus d'informations : Arrêtez votre modèle.

Utiliser votre modèle sur un appareil Edge

Vous pouvez utiliser un modèle Lookout for Vision sur AWS IoT Greengrass Version 2 un appareil principal.

Pour plus d'informations : Utilisation de votre modèle Amazon Lookout for Vision sur un appareil périphérique.

Utilisez votre tableau de bord

Vous pouvez utiliser le tableau de bord pour obtenir une vue d'ensemble de tous vos projets et des informations générales pour des projets individuels.

Pour plus d'informations : Utilisez votre tableau de bord.

Commencer à utiliser Amazon Lookout for Vision

Avant de commencer ces instructions de Mise en route, nous vous recommandons de lire Comprendre Amazon Lookout for Vision.

Les instructions de démarrage vous montrent comment créer un exemple de modèle de segmentation d'image. Si vous souhaitez créer un exemple de modèle de classification d'images, consultezEnsemble de données de classification d'images.

Si vous souhaitez essayer rapidement un exemple de modèle, nous fournissons des exemples d'images d'entraînement et des images de masque. Nous fournissons également un script Python qui crée un fichier manifeste de segmentation d'image. Vous utilisez le fichier manifeste pour créer un ensemble de données pour votre projet et vous n'avez pas besoin d'étiqueter les images du jeu de données. Lorsque vous créez un modèle avec vos propres images, vous devez étiqueter les images dans le jeu de données. Pour de plus amples informations, veuillez consulter Création de votre jeu de données.

Les images que nous fournissons sont des cookies normaux et anormaux. Un cookie anormal présente une fissure sur la forme du cookie. Le modèle que vous entraînez avec les images prédit une classification (normale ou anormale) et trouve la zone (masque) de fissures dans un cookie anormal, comme illustré dans l'exemple suivant.

Rubriques

- Étape 1 : créer le fichier manifeste et télécharger des images
- Étape 2 : Création du modèle
- Étape 3 : démarrer le modèle
- Étape 4 : Analyser une image
- Étape 5 : Arrêter le modèle
- Étapes suivantes

Étape 1 : créer le fichier manifeste et télécharger des images

Au cours de cette procédure, vous devez cloner le référentiel de documentation Amazon Lookout for Vision sur votre ordinateur. Vous utilisez ensuite un script Python (version 3.7 ou supérieure) pour créer un fichier manifeste et télécharger les images d'entraînement et les images de masque vers un emplacement Amazon S3 que vous spécifiez. Vous utilisez le fichier manifeste pour créer votre modèle. Plus tard, vous utiliserez des images de test dans le référentiel local pour essayer votre modèle.

Pour créer le fichier manifeste et télécharger des images

1. Configurez Amazon Lookout for Vision en suivant les instructions de la section Configuration d'Amazon Lookout for Vision. Assurez-vous d'installer le AWS SDK pour Python.
2. Dans la AWS région dans laquelle vous souhaitez utiliser Lookout for Vision, créez un compartiment S3.
3. Dans le compartiment Amazon S3, créez un dossier nommé`getting-started`.
4. Notez l'URI Amazon S3 et le nom de ressource Amazon (ARN) du dossier. Vous les utilisez pour configurer les autorisations et exécuter le script.
5. Assurez-vous que l'utilisateur qui appelle le script est autorisé à lancer l'`s3:PutObject`opération. Vous pouvez utiliser la politique suivante. Pour attribuer des autorisations, consultez Attribution d'autorisations.

   ```
   {
    "Version": "2012-10-17",
    "Statement": [{
     "Sid": "Statement1",
     "Effect": "Allow",
     "Action": [
      "s3:PutObject"
     ],
     "Resource": [
      "arn:aws:s3::: ARN for S3 folder in step 4/*"
     ]
    }]
   }
   ```

6. Assurez-vous qu'un profil local est nommé `lookoutvision-access` et que l'utilisateur du profil dispose de l'autorisation obtenue à l'étape précédente. Pour de plus amples informations, veuillez consulter Utiliser un profil sur votre ordinateur local.

7. Téléchargez le fichier zip, getting-started.zip. Le fichier zip contient le jeu de données de démarrage et le script de configuration.
8. Décompressez le fichier `getting-started.zip`.
9. À l'invite de commande, procédez comme suit :

   a. Accédez au dossier `getting-started`.

   b. Exécutez la commande suivante pour créer un fichier manifeste et télécharger les images d'entraînement et les masques d'image sur le chemin Amazon S3 que vous avez indiqué à l'étape 4.

   ```
   python getting_started.py S3-URI-from-step-4
   ```

   c. Lorsque le script est terminé, notez le chemin du `train.manifest` fichier affiché ensuite par le script`Create dataset using manifest file:`. Le chemin doit être similaire às3://*path to getting started folder*/manifests/train.manifest.

Étape 2 : Création du modèle

Dans cette procédure, vous créez un projet et un ensemble de données à l'aide des images et du fichier manifeste que vous avez précédemment chargés dans votre compartiment Amazon S3. Vous créez ensuite le modèle et visualisez les résultats de l'évaluation issus de l'entraînement du modèle.

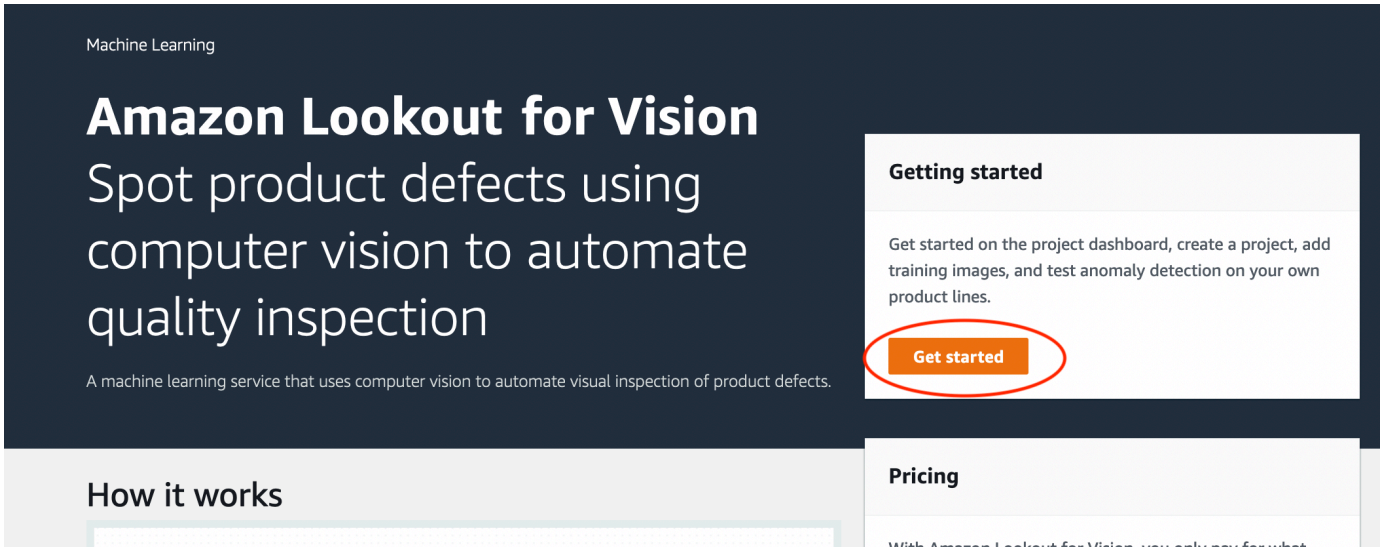
Comme vous créez l'ensemble de données à partir du fichier manifeste de démarrage, il n'est pas nécessaire d'étiqueter les images du jeu de données. Lorsque vous créez un jeu de données avec vos propres images, vous devez étiqueter les images. Pour de plus amples informations, veuillez consulter Étiquetage des images.

> **Important**
>
> La réussite de la formation d'un mannequin vous est facturée.

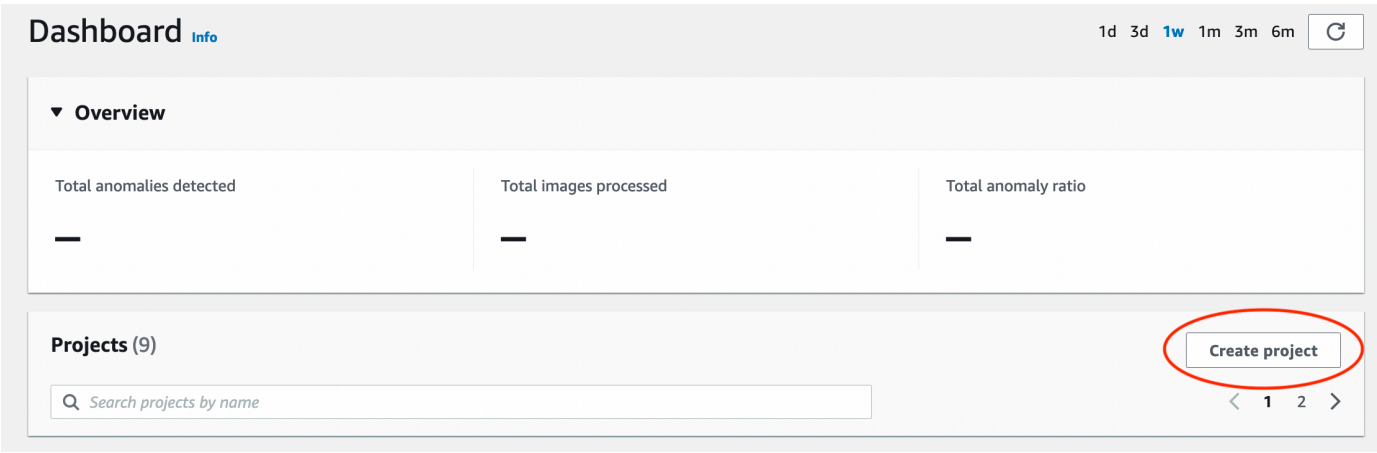
Pour créer un modèle

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/

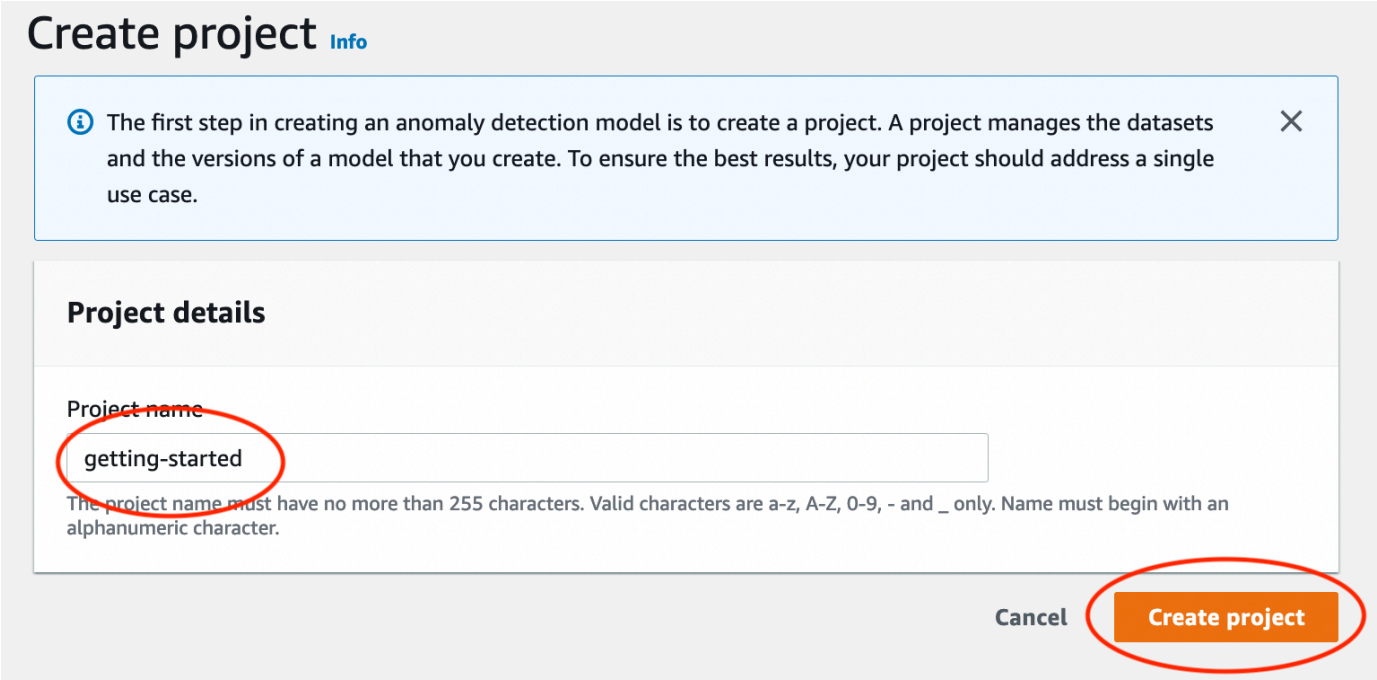
2. Assurez-vous que vous vous trouvez dans la même AWS région que celle dans laquelle vous avez créé le compartiment Amazon S3Étape 1 : créer le fichier manifeste et télécharger des images. Pour modifier la région, choisissez le nom de la région actuellement affichée dans la barre de navigation. Sélectionnez ensuite la région vers laquelle vous souhaitez passer.
3. Choisissez Démarrer.



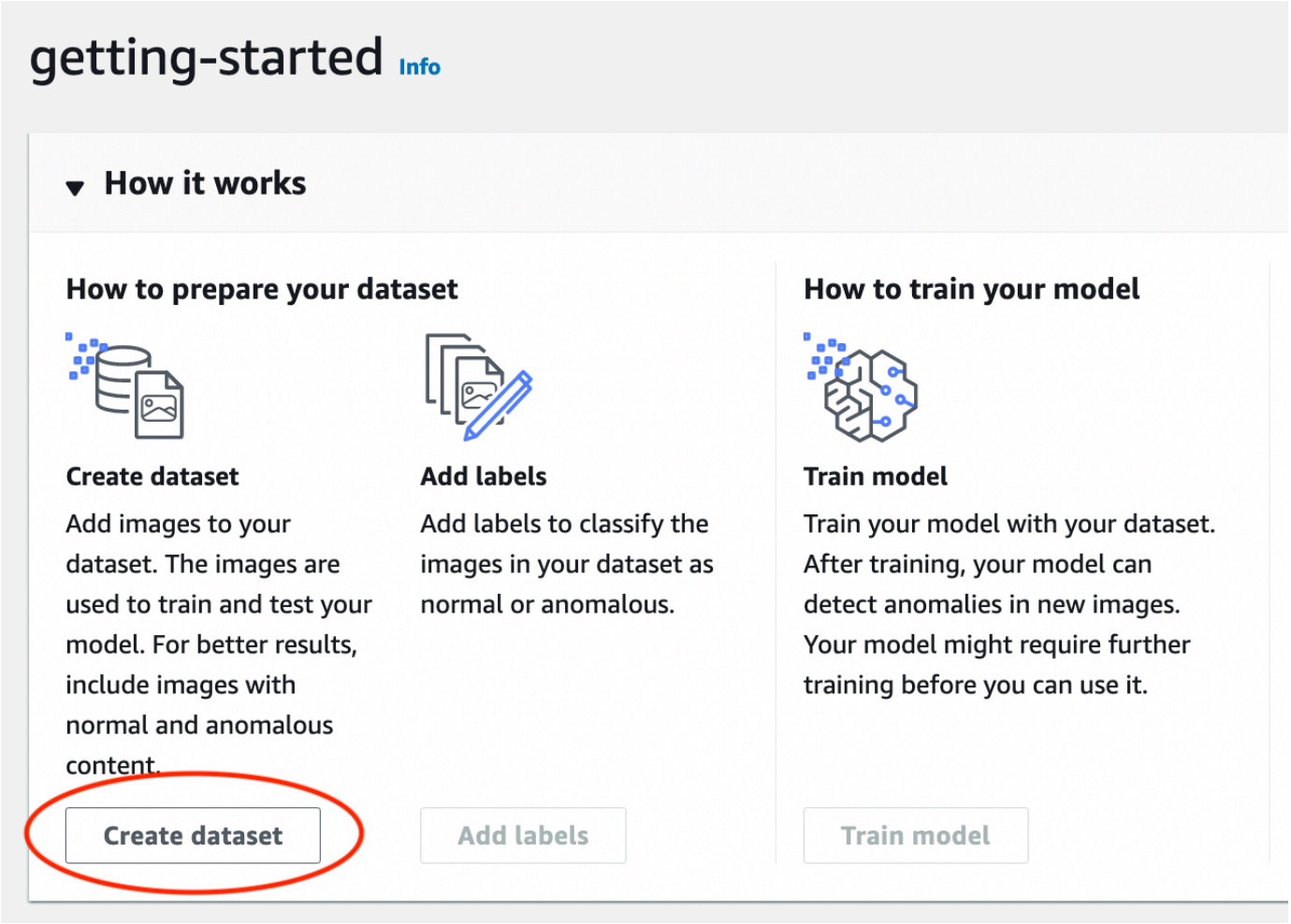
4. Dans la section Projets, choisissez Créer un projet.



5. Sur la page Créer un projet, procédez comme suit :
   a. Dans Nom du projet, entrez`getting-started`.
   b. Sélectionnez Create a project (Créer un projet).



6. Sur la page du projet, dans la section Fonctionnement, sélectionnez Créer un ensemble de données.



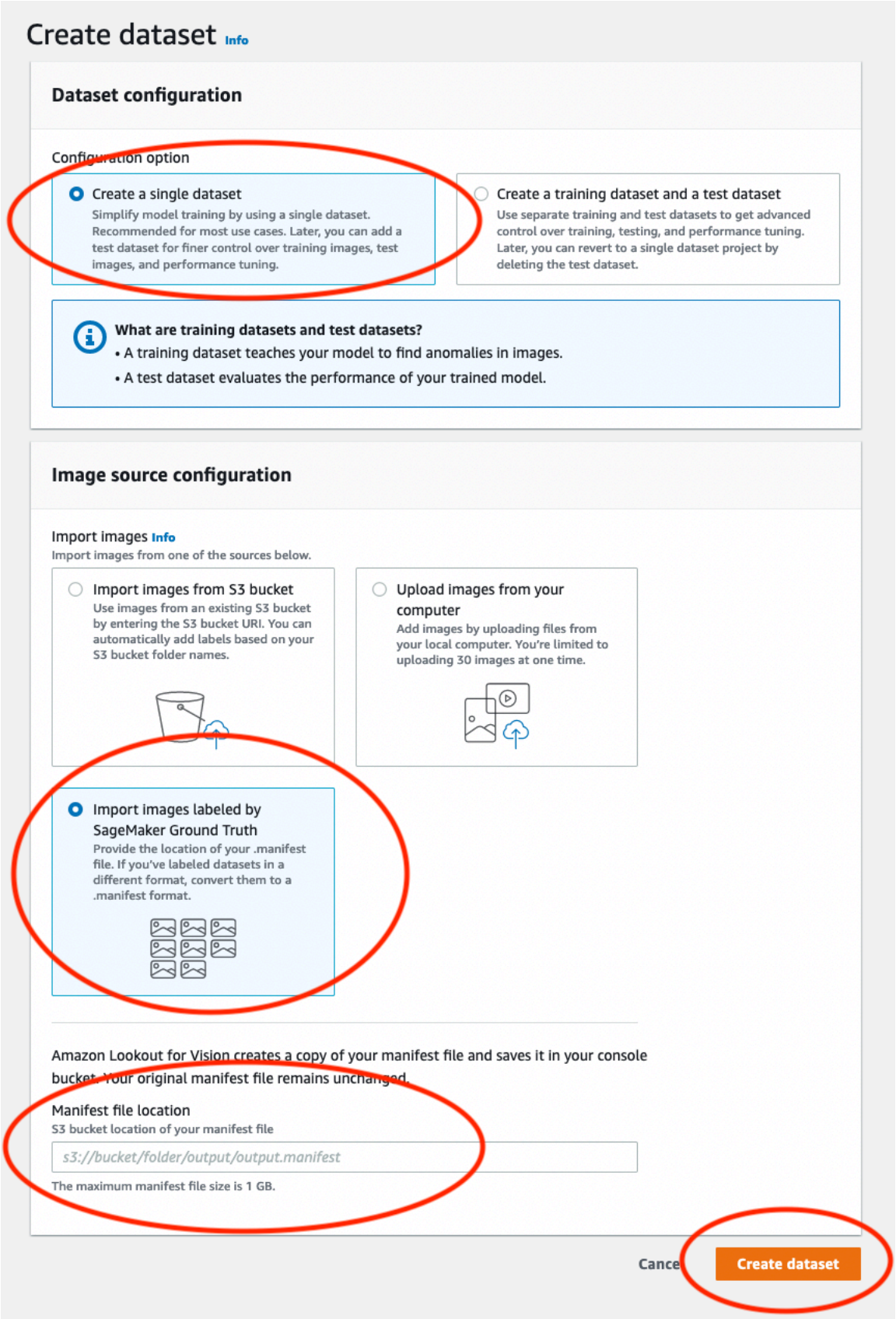
7. Sur la page Créer un jeu de données, procédez comme suit :

   a. Choisissez Créer un jeu de données unique.

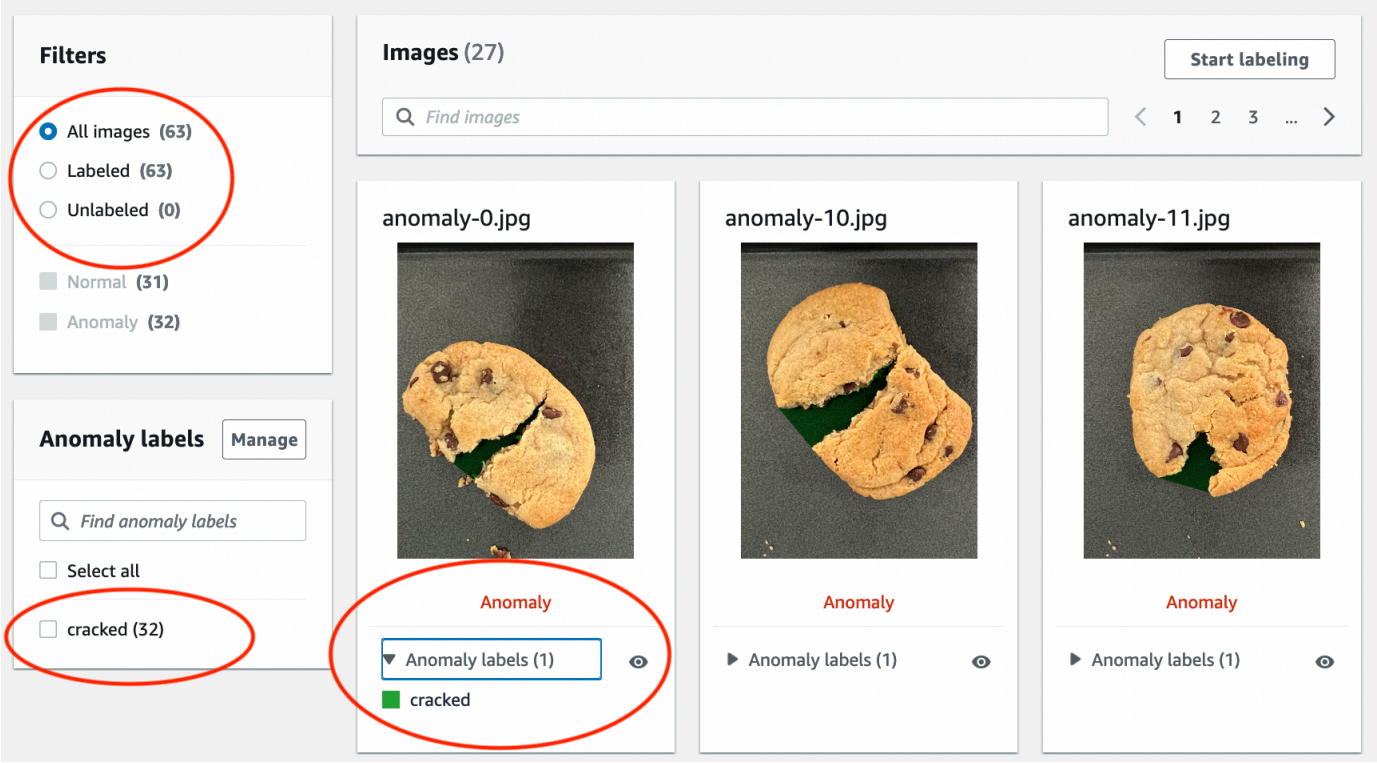
   b. Dans la section Configuration de la source d'image, choisissez Importer des images étiquetées par SageMaker Ground Truth.

   c. Pour l'emplacement du fichier .manifest, entrez l'emplacement Amazon S3 du fichier manifeste que vous avez noté à l'étape 6.c. de. Étape 1 : créer le fichier manifeste et télécharger des images L'emplacement d'Amazon S3 doit être similaire à `s3://path to getting started folder/manifests/train.manifest`

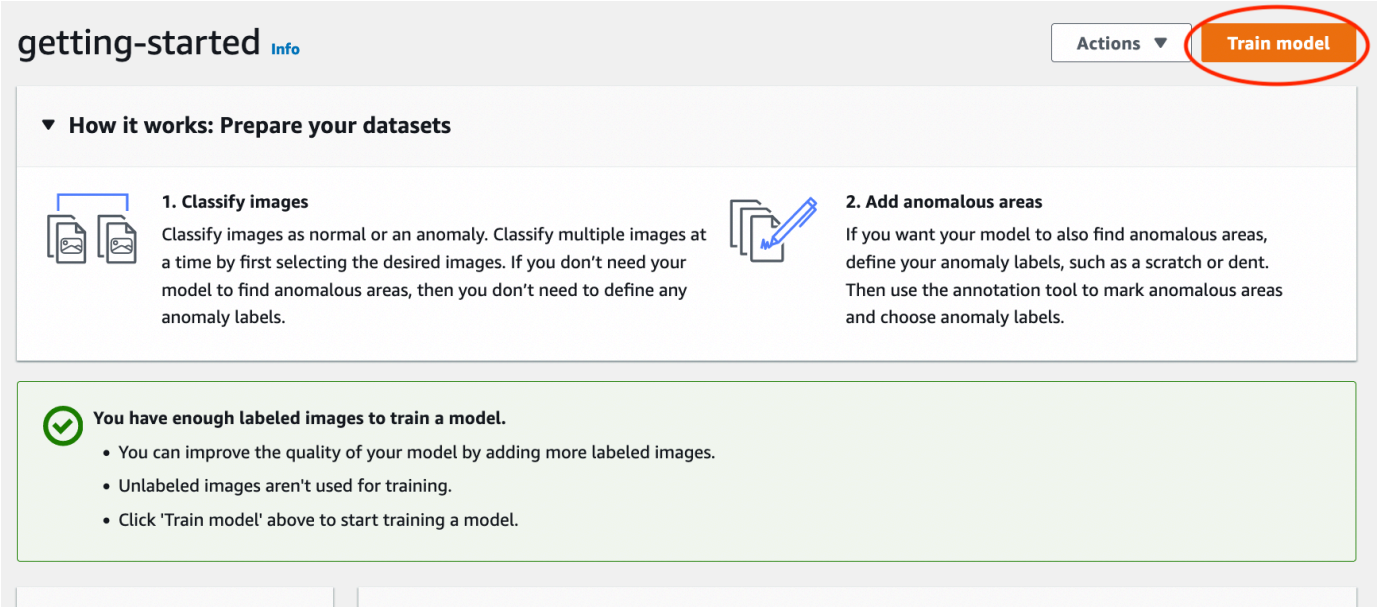
   d. Choisissez Créer un jeu de données.


Create dataset Info
Dataset configuration
Configuration option
Create a single dataset
Simplify model training by using a single dataset. Recommended for most use cases. Later, you can add a test dataset for finer control over training images, test images, and performance tuning.
Create a training dataset and a test dataset
Use separate training and test datasets to get advanced control over training, testing, and performance tuning. Later, you can revert to a single dataset project by deleting the test dataset.
What are training datasets and test datasets?
• A training dataset teaches your model to find anomalies in images.
• A test dataset evaluates the performance of your trained model.
Image source configuration
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 URI. 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.
Import images labeled by SageMaker Ground Truth
Provide the location of your .manifest file. If you've labeled datasets in a different format, convert them to a .manifest format.
Amazon Lookout for Vision creates a copy of your manifest file and saves it in your console bucket. Your original manifest file remains unchanged.
Manifest file location
S3 bucket location of your manifest file
s3://bucket/folder/output/output.manifest
The maximum manifest file size is 1 GB.
Cancel
Create dataset

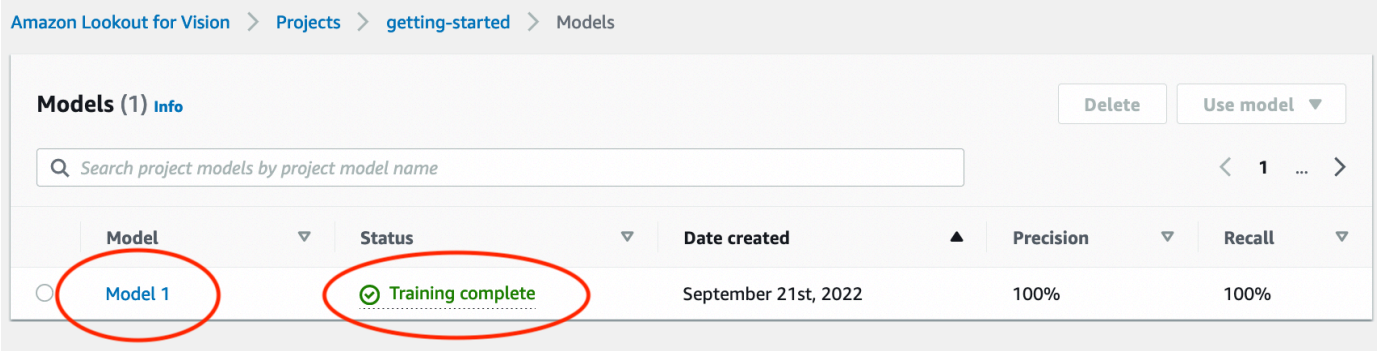
8. Sur la page des détails du projet, dans la section Images, visualisez les images du jeu de données. Vous pouvez consulter les informations de classification et de segmentation d'image (étiquettes de masque et d'anomalie) pour chaque image du jeu de données. Vous pouvez également rechercher des images, filtrer les images en fonction de leur statut d'étiquetage (étiqueté/non étiqueté) ou filtrer les images en fonction des étiquettes d'anomalie qui leur sont attribuées.



9. Sur la page des détails du projet, choisissez Train model.

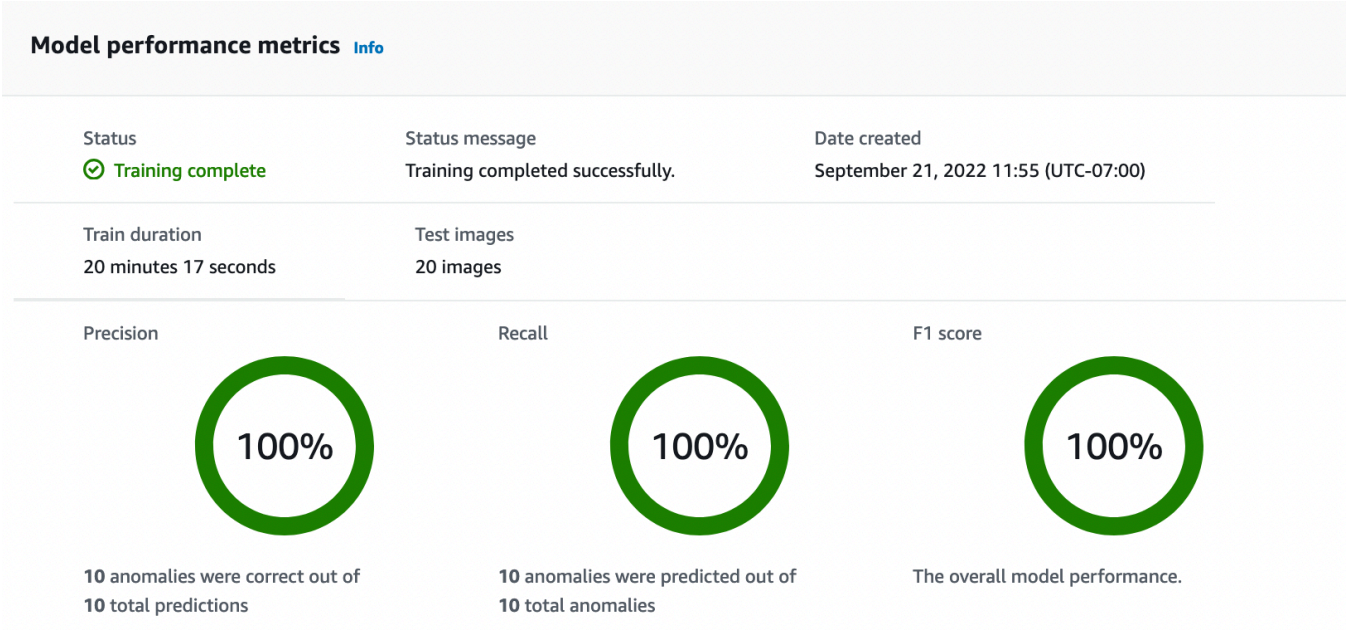


10. Sur la page des détails du modèle de train, choisissez Modèle de train.
11. Dans la boîte de dialogue Voulez-vous entraîner votre modèle ?, choisissez Entraîner un modèle.
12. Sur la page Modèles du projet, vous pouvez voir que la formation a commencé. Vérifiez l'état actuel en consultant la colonne État de la version du modèle. L'entraînement du modèle prend au moins 30 minutes. L'entraînement est terminé avec succès lorsque le statut passe à Entraînement terminé.
13. Lorsque l'entraînement est terminé, choisissez le modèle 1 sur la page Modèles.

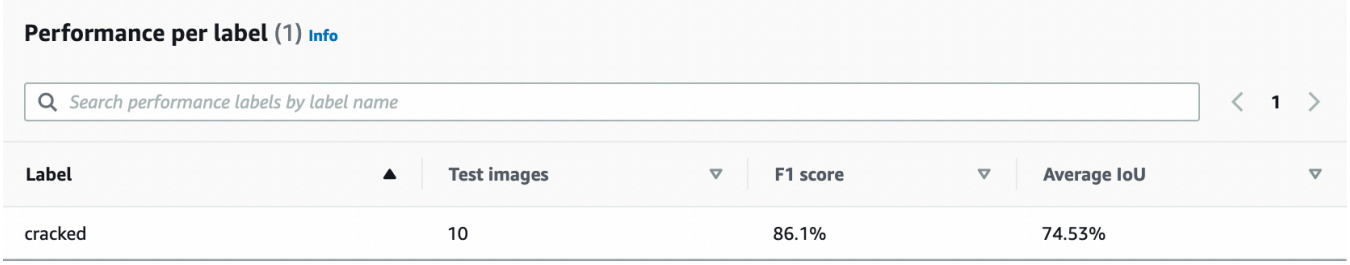


14. Sur la page de détails du modèle, consultez les résultats de l'évaluation dans l'onglet Mesures de performance. Il existe des indicateurs pour les éléments suivants :

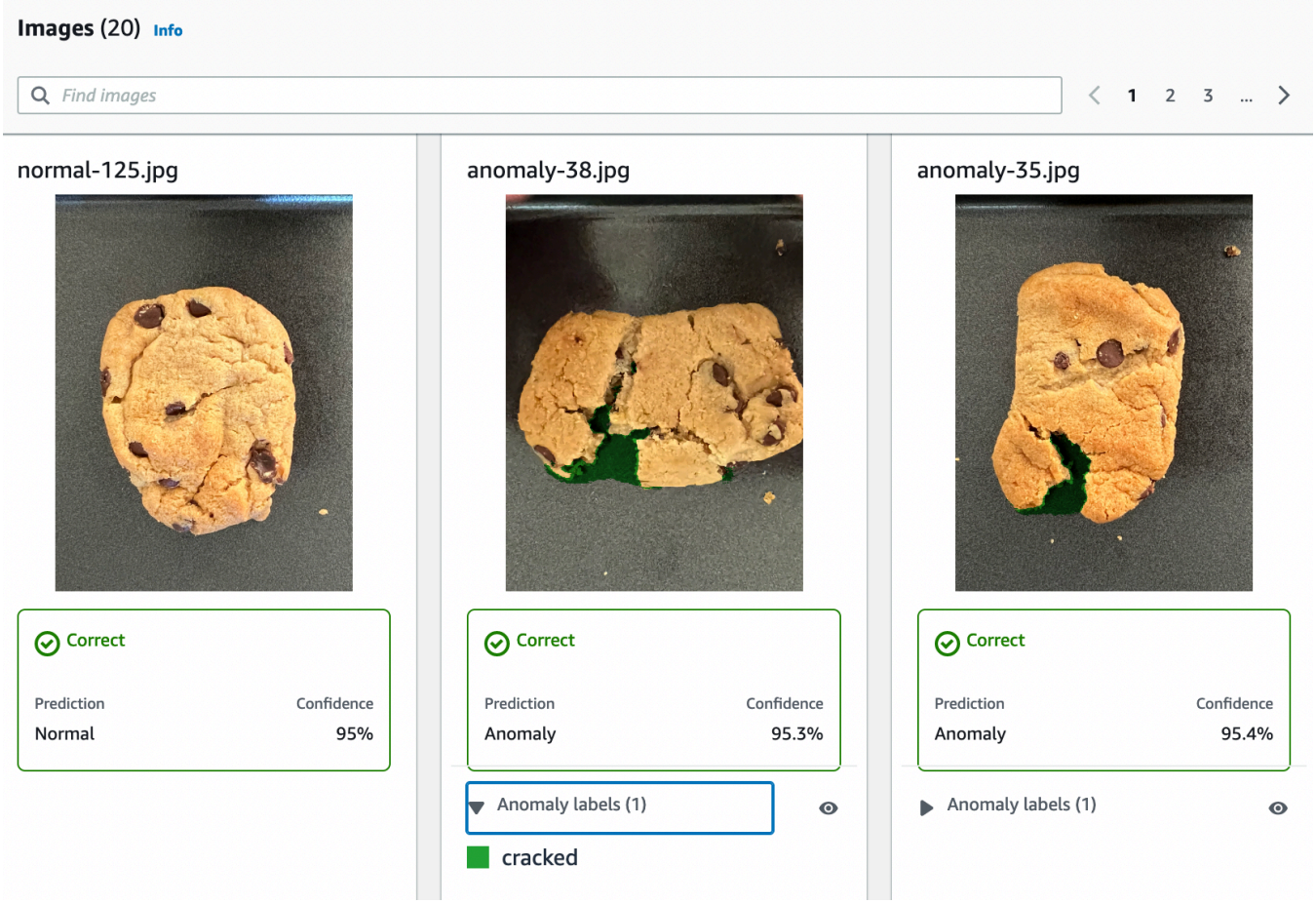
    - Mesures de performance globales du modèle (précision, rappel et score F1) pour les prédictions de classification effectuées par le modèle.



- Mesures de performance pour les étiquettes d'anomalies trouvées dans les images de test (IoU moyen, score F1)



- Prédictions pour les images de test (classification, masques de segmentation et étiquettes d'anomalies)



L'entraînement du modèle n'étant pas déterministe, les résultats de votre évaluation peuvent différer des résultats présentés sur cette page. Pour de plus amples informations, veuillez consulter Améliorer votre modèle Amazon Lookout for Vision.

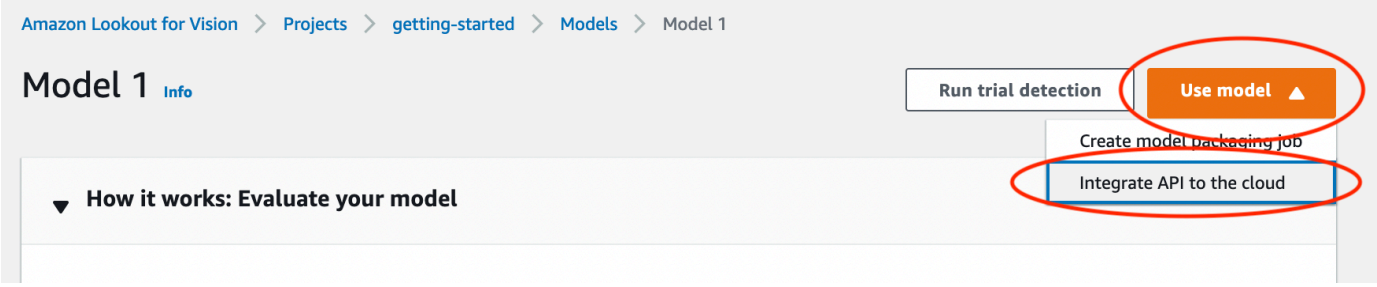
Étape 3 : démarrer le modèle

Au cours de cette étape, vous commencez à héberger le modèle afin qu'il soit prêt à analyser les images. Pour de plus amples informations, veuillez consulter Exécution de votre modèle Amazon Lookout for Vision entraîné.

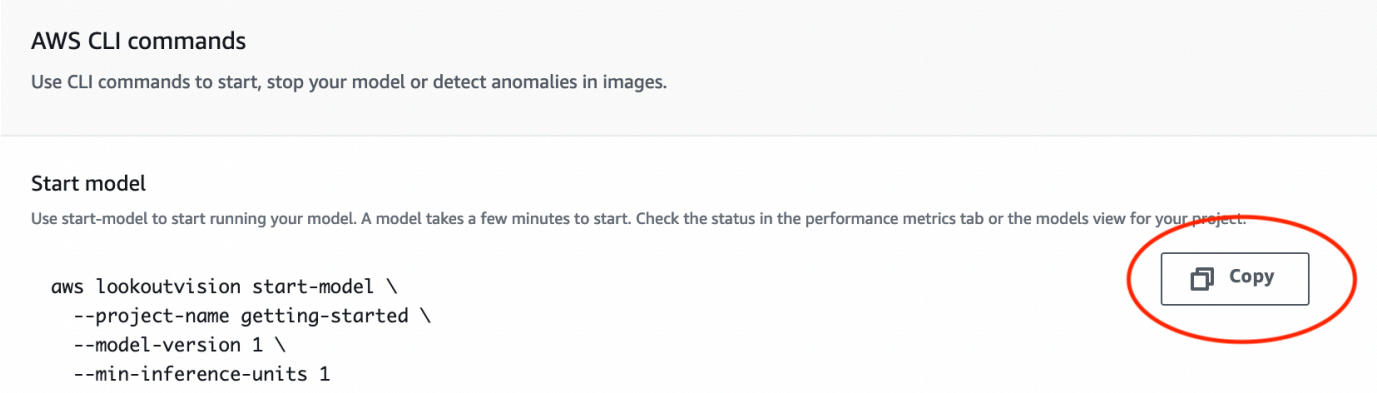
> **Note**
>
> La durée de fonctionnement de votre modèle vous est facturée. Tu arrêtes ton mannequinÉtape 5 : Arrêter le modèle.

Pour démarrer le modèle.

1. Sur la page de détails du modèle, choisissez Utiliser le modèle, puis choisissez Intégrer l'API au cloud.



2. Dans la section AWS CLI des commandes, copiez la `start-model` AWS CLI commande.



3. Assurez-vous que le AWS CLI est configuré pour fonctionner dans la même AWS région que celle dans laquelle vous utilisez la console Amazon Lookout for Vision. Pour modifier la AWS région qu'il AWS CLI utilise, voirInstallez les AWS SDK.

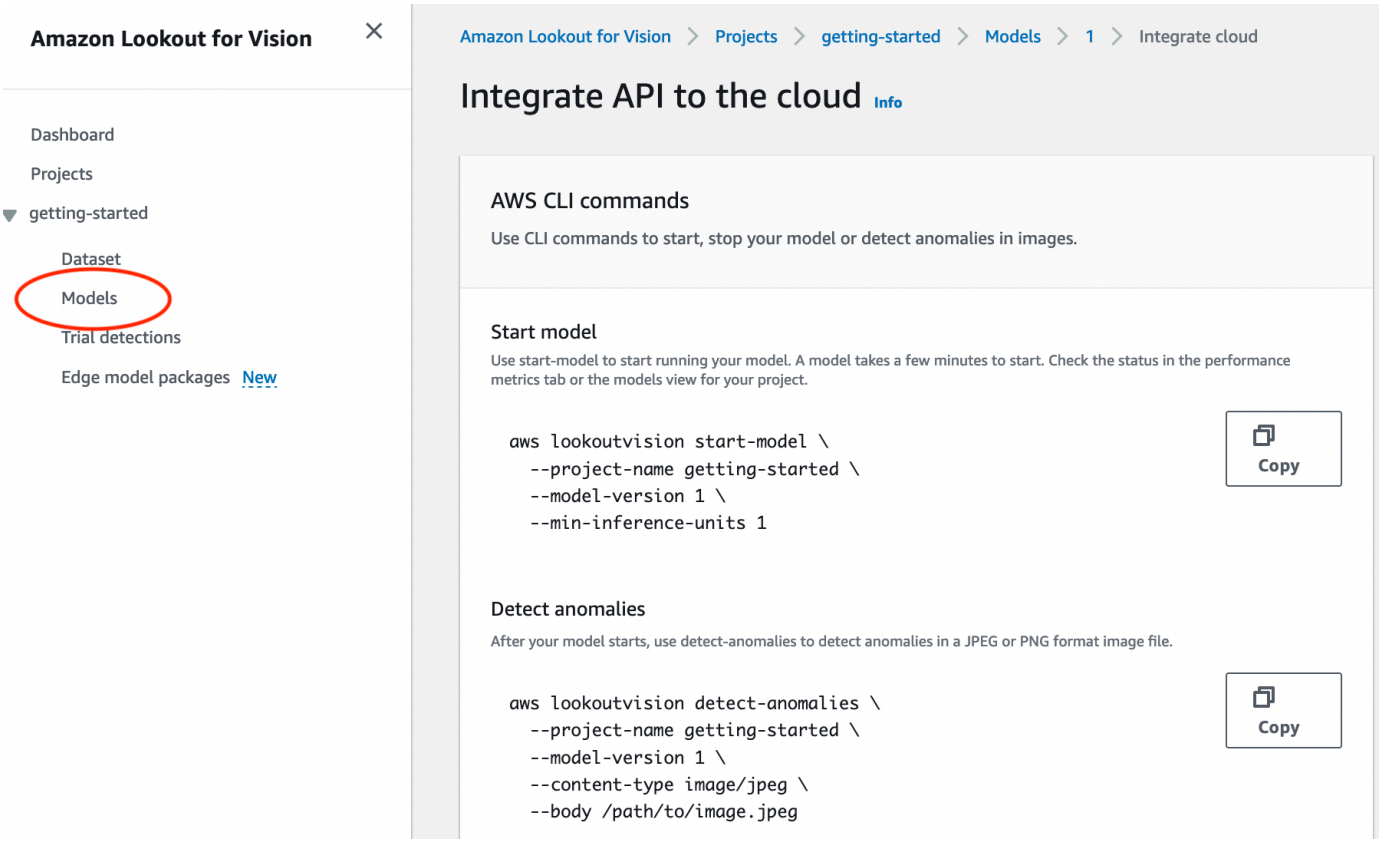
4. À l'invite de commande, démarrez le modèle en saisissant la `start-model` commande. Si vous utilisez le `lookoutvision` profil pour obtenir des informations d'identification, ajoutez le `--profile lookoutvision-access` paramètre. Par exemple :

```
aws lookoutvision start-model \
  --project-name getting-started \
  --model-version 1 \
  --min-inference-units 1 \
  --profile lookoutvision-access
```

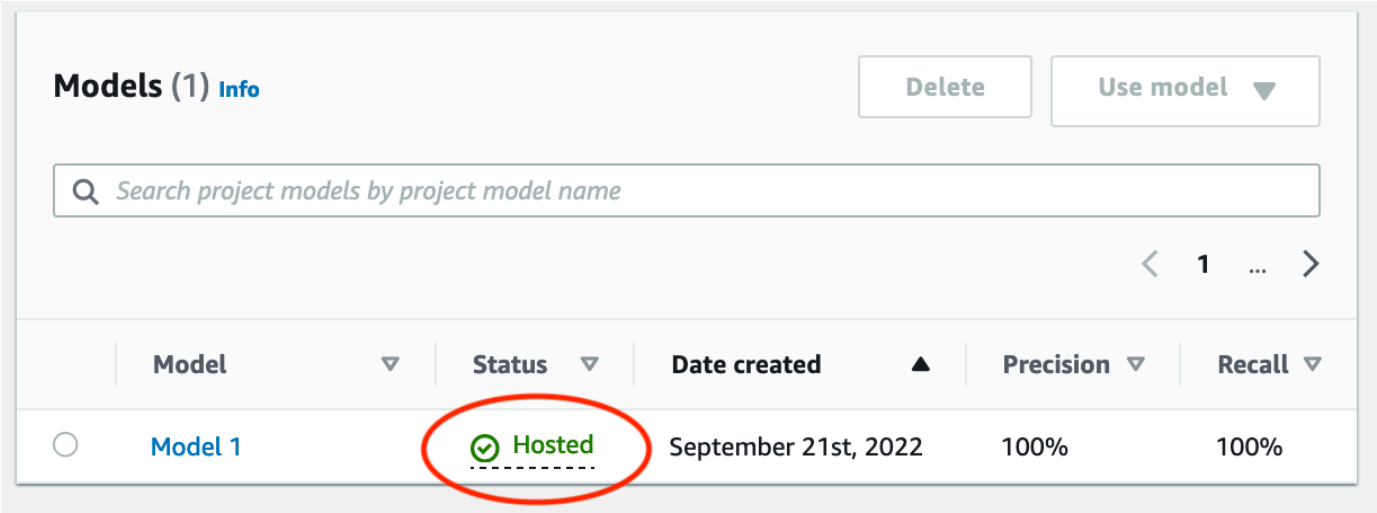
Si l'appel est réussi, le résultat suivant s'affiche :

```
{
    "Status": "STARTING_HOSTING"
}
```

5. De retour dans la console, choisissez Models dans le volet de navigation.



6. Attendez que le statut du modèle (modèle 1) dans la colonne État affiche Hosted. Si vous avez déjà entraîné un modèle dans le cadre du projet, attendez que la dernière version du modèle soit terminée.

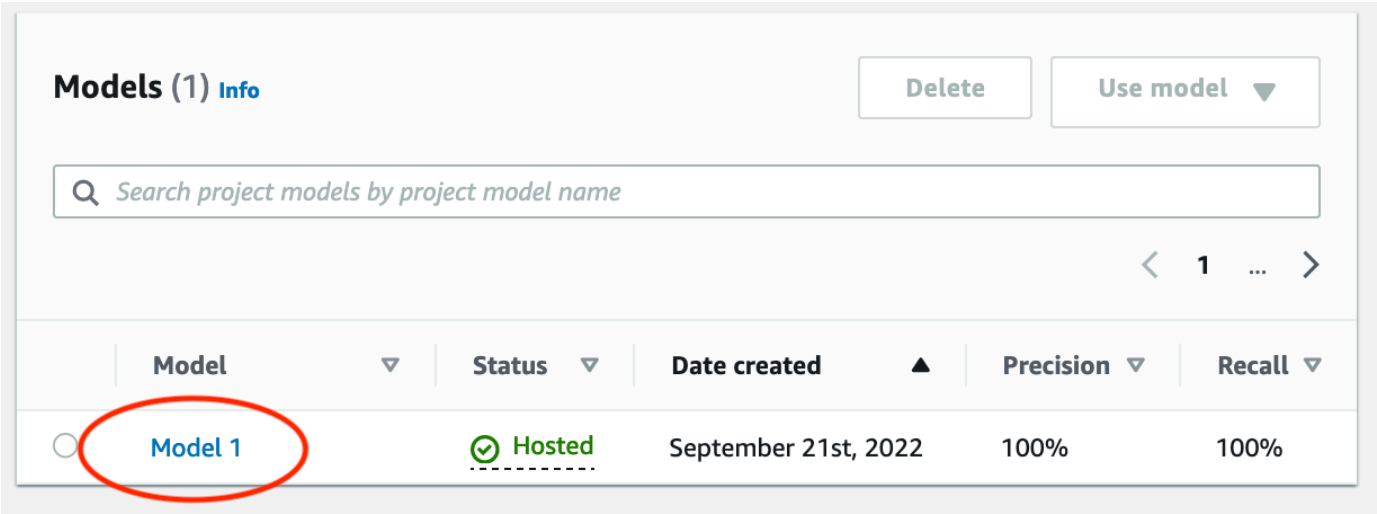


Étape 4 : Analyser une image

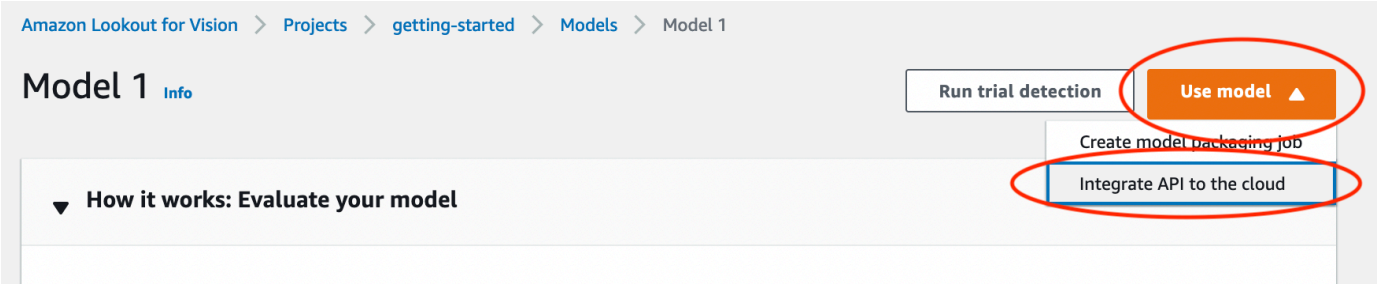
Au cours de cette étape, vous analysez une image avec votre modèle. Nous fournissons des exemples d'images que vous pouvez utiliser dans le `test-images` dossier de démarrage du référentiel de documentation Lookout for Vision sur votre ordinateur. Pour de plus amples informations, veuillez consulter Détecter des anomalies dans une image.

Pour analyser une image

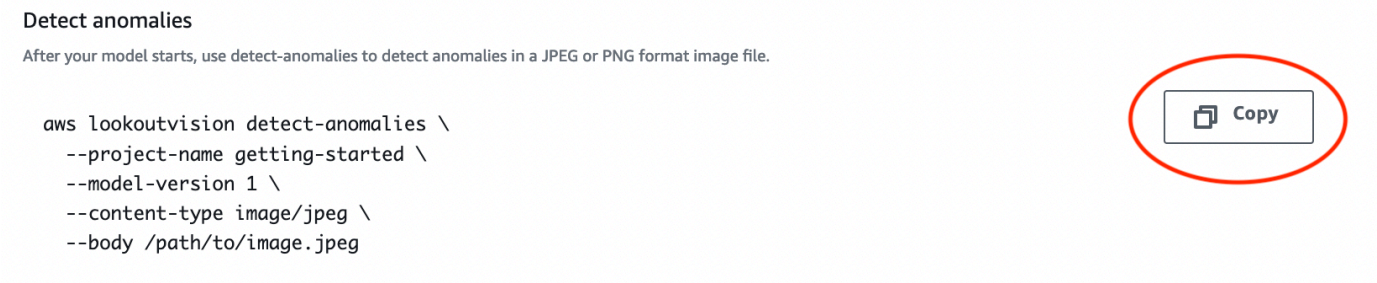
1. Sur la page Modèles, choisissez le modèle Modèle 1.



2. Sur la page de détails du modèle, choisissez Utiliser le modèle, puis choisissez Intégrer l'API au cloud.



3. Dans la section AWS CLI des commandes, copiez la `detect-anomalies` AWS CLI commande.



4. À l'invite de commande, analysez une image anormale en saisissant la `detect-anomalies` commande de l'étape précédente. Pour le `--body` paramètre, spécifiez une image anormale à partir du `test-images` dossier de démarrage de votre ordinateur. Si vous utilisez le `lookoutvision` profil pour obtenir des informations d'identification, ajoutez le `--profile lookoutvision-access` paramètre. Par exemple :

```
aws lookoutvision detect-anomalies \
  --project-name getting-started \
  --model-version 1 \
  --content-type image/jpeg \
  --body /path/to/test-images/test-anomaly-1.jpg \
  --profile lookoutvision-access
```

La sortie doit ressembler à ce qui suit :

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.983975887298584,
        "Anomalies": [
```

```
            {
                "Name": "background",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.9818974137306213,
                    "Color": "#FFFFFF"
                }
            },
            {
                "Name": "cracked",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.018102575093507767,
                    "Color": "#23A436"
                }
            }
        ],
        "AnomalyMask": "iVBORw0KGgoAAAANSUhEUgAAAkAAAAMACA......"
    }
}
```

5. Dans le résultat, notez ce qui suit :

   - `IsAnomalous`est un booléen pour la classification prédite. `true`si l'image est anormale, sinon`false`.
   - `Confidence`est une valeur flottante représentant le niveau de confiance d'Amazon Lookout for Vision dans la prédiction. 0 représente le niveau de confiance le plus faible, 1 le niveau de confiance le plus élevé.
   - `Anomalies`est une liste des anomalies détectées dans l'image. `Name`est l'étiquette d'anomalie. `PixelAnomaly`inclut le pourcentage total de surface de l'anomalie (`TotalPercentageArea`) et une couleur (`Color`) pour l'étiquette d'anomalie. La liste inclut également une anomalie « d'arrière-plan » qui couvre la zone située en dehors des anomalies détectées sur l'image.
   - `AnomalyMask`est une image de masque qui indique l'emplacement des anomalies sur l'image analysée.

   Vous pouvez utiliser les informations contenues dans la réponse pour afficher une combinaison de l'image analysée et du masque d'anomalie, comme illustré dans l'exemple suivant. Pour obtenir un exemple de code, consultez Affichage des informations de classification et de segmentation.

Classification:
Prediction: Anomalous
Confidence: 99.9%
Segmentation:
Anomaly: cracked. Area: 6.2%

6. À l'invite de commande, analysez une image normale depuis le `test-images` dossier de démarrage. Si vous utilisez le `lookoutvision` profil pour obtenir des informations d'identification, ajoutez le `--profile lookoutvision-access` paramètre. Par exemple :

```
aws lookoutvision detect-anomalies \
  --project-name getting-started \
  --model-version 1 \
  --content-type image/jpeg \
  --body /path/to/test-images/test-normal-1.jpg \
  --profile lookoutvision-access
```

   La sortie doit ressembler à ce qui suit :

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": false,
        "Confidence": 0.9916400909423828,
        "Anomalies": [
            {
                "Name": "background",
                "PixelAnomaly": {
                    "TotalPercentageArea": 1.0,
                    "Color": "#FFFFFF"
                }
            }
        ],
        "AnomalyMask": "iVBORw0KGgoAAAANSUhEUgAAAkAAAA....."
    }
}
```

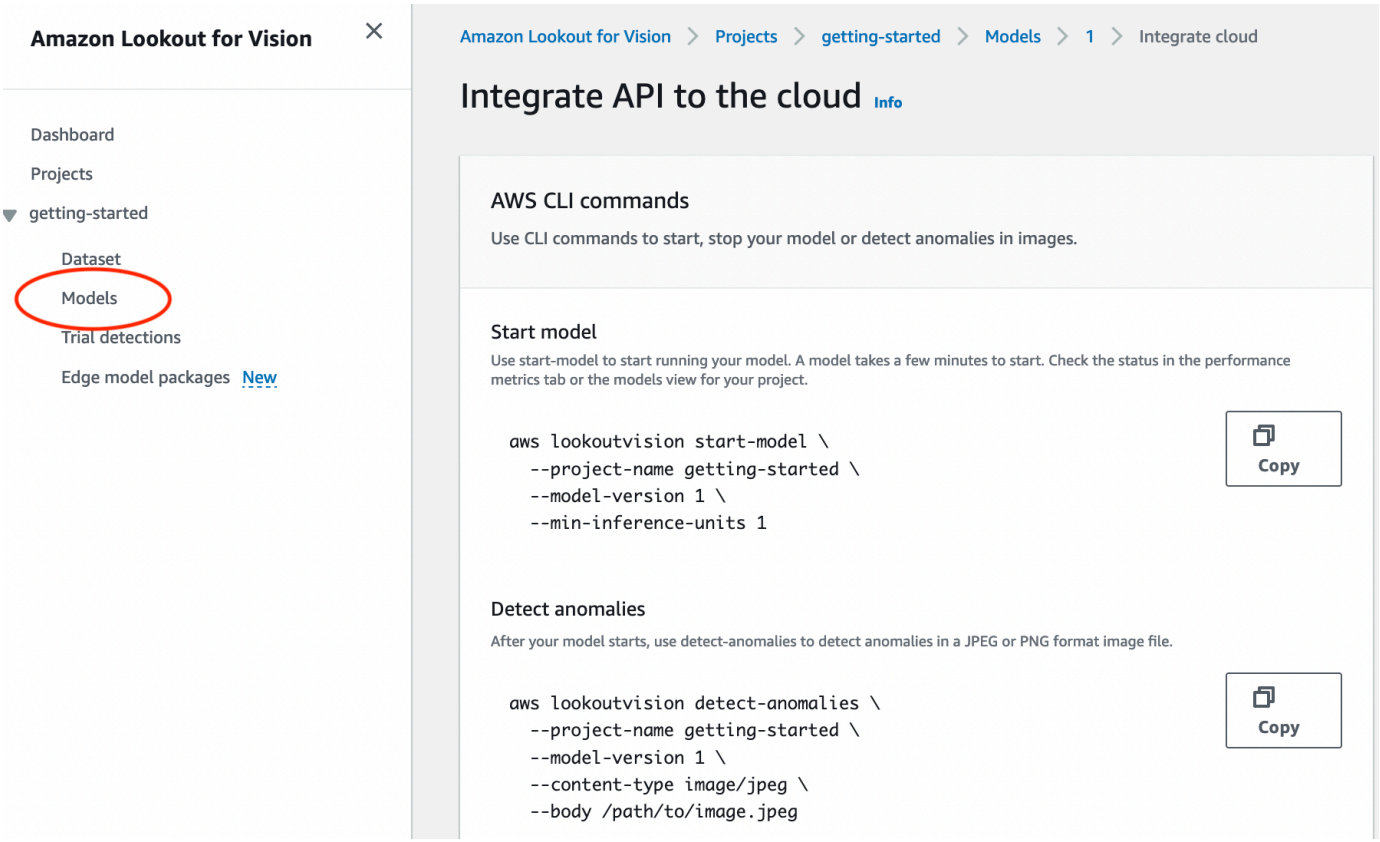
7. Dans le résultat, notez que la `false` valeur pour `IsAnomalous` classe l'image comme ne présentant aucune anomalie. `Confidence`À utiliser pour vous aider à déterminer votre niveau de confiance dans la classification. De plus, le `Anomalies` tableau ne possède que l'étiquette d'`background`anomalie.

Étape 5 : Arrêter le modèle

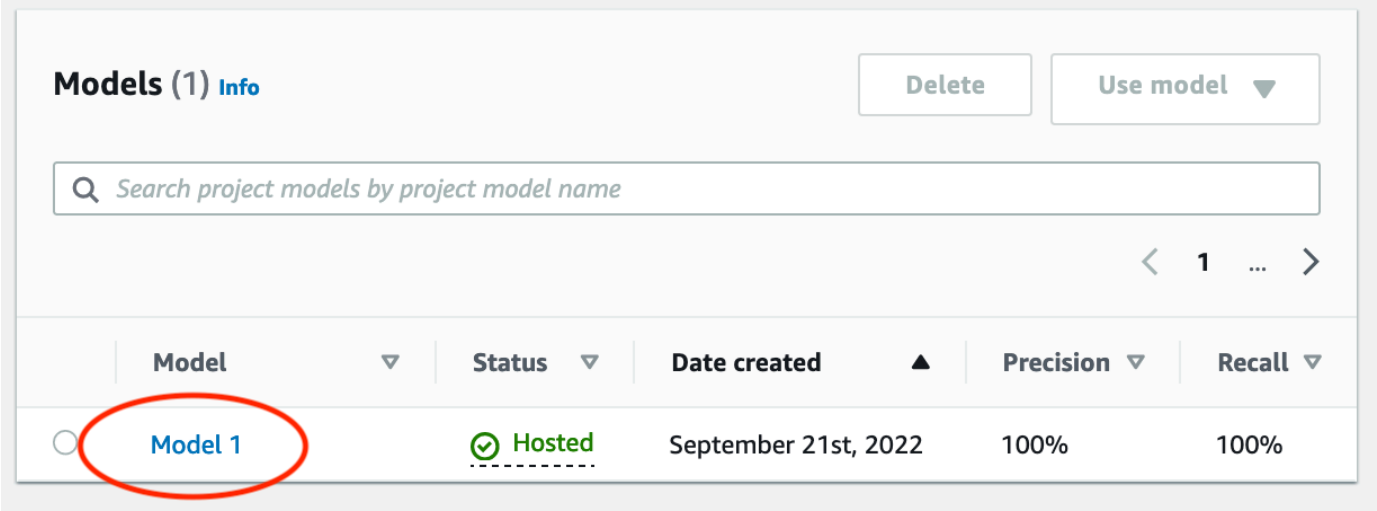
Au cours de cette étape, vous arrêtez d'héberger le modèle. La durée de fonctionnement de votre modèle vous est facturée. Si vous n'utilisez pas le modèle, vous devez l'arrêter. Vous pouvez redémarrer le modèle la prochaine fois que vous en aurez besoin. Pour de plus amples informations, veuillez consulter Démarrage de votre modèle Amazon Lookout for Vision.

Pour arrêter le modèle.

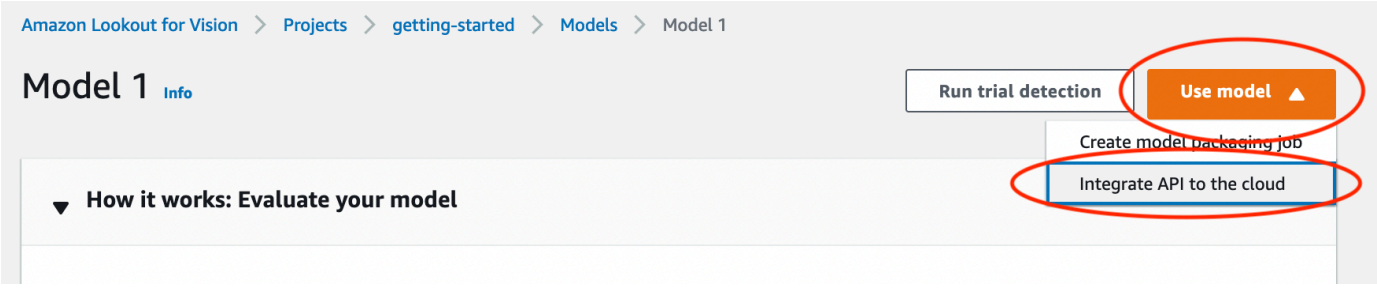
1. Choisissez Models dans le volet de navigation.



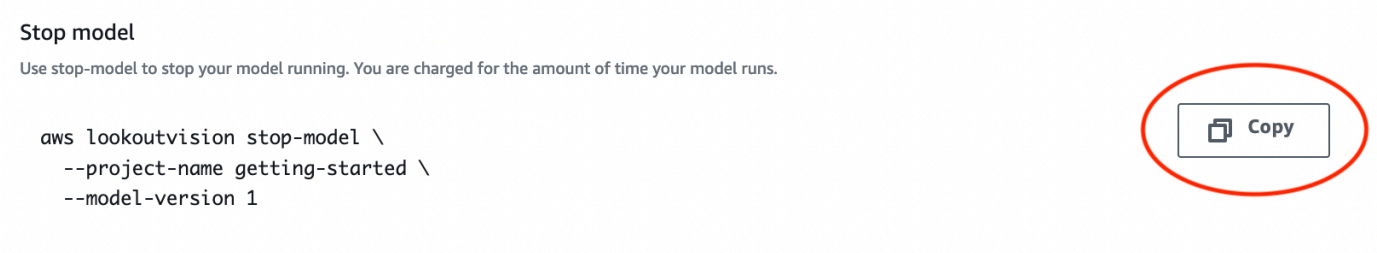
2. Sur la page Modèles, choisissez le modèle Modèle 1.



3. Sur la page de détails du modèle, choisissez Utiliser le modèle, puis choisissez Intégrer l'API au cloud.



4. Dans la section AWS CLI des commandes, copiez la `stop-model` AWS CLI commande.



5. À l'invite de commande, arrêtez le modèle en saisissant la `stop-model` AWS CLI commande de l'étape précédente. Si vous utilisez le `lookoutvision` profil pour obtenir des informations d'identification, ajoutez le `--profile lookoutvision-access` paramètre. Par exemple :

```
aws lookoutvision stop-model \
  --project-name getting-started \
  --model-version 1 \
  --profile lookoutvision-access
```

Si l'appel est réussi, le résultat suivant s'affiche :

```
{
    "Status": "STOPPING_HOSTING"
}
```

6. De retour dans la console, choisissez Modèles dans la page de navigation de gauche.
7. Le modèle s'est arrêté lorsque le statut du modèle dans la colonne État est « Entraînement terminé ».

Étapes suivantes

Lorsque vous êtes prêt à créer un modèle avec vos propres images, commencez par suivre les instructions deCréation de votre projet. Les instructions incluent les étapes de création d'un modèle à l'aide de la console Amazon Lookout for Vision et AWS du SDK.

Si vous souhaitez essayer d'autres exemples de jeux de données, consultezExemples de code et de jeux de données.

Création de votre modèle Amazon Lookout for Vision

Un modèle Amazon Lookout for Vision est un modèle d'apprentissage automatique qui prédit la présence d'anomalies dans les nouvelles images en trouvant des modèles dans les images utilisées pour entraîner le modèle. Cette section explique comment créer et entraîner un modèle. Après avoir entraîné votre modèle, vous évaluez ses performances. Pour de plus amples informations, veuillez consulter Améliorer votre modèle Amazon Lookout for Vision.

Avant de créer votre premier modèle, nous vous recommandons de lire Comprendre Amazon Lookout for Vision etCommencer à utiliser Amazon Lookout for Vision. Si vous utilisez le AWS SDK, lisezAppelez une opération Amazon Lookout for Vision.

Rubriques

- Création de votre projet
- Création de votre jeu de données
- Étiquetage des images
- Entraînement de votre modèle
- Modèle de résolution des problèmes

Création de votre projet

Un projet Amazon Lookout for Vision regroupe les ressources nécessaires à la création et à la gestion d'un modèle Lookout for Vision. Un projet gère les éléments suivants :

- Ensemble de données : images et étiquettes d'images utilisées pour entraîner un modèle. Pour de plus amples informations, veuillez consulter Création de votre jeu de données.
- Modèle : logiciel que vous entraînez pour détecter les anomalies. Vous pouvez avoir plusieurs versions d'un modèle. Pour de plus amples informations, veuillez consulter Entraînement de votre modèle.

Nous vous recommandons d'utiliser un projet pour un seul cas d'utilisation, tel que la détection d'anomalies dans un seul type de pièce de machine.

> **Note**
>
> Vous pouvez l'utiliser AWS CloudFormation pour approvisionner et configurer les projets Amazon Lookout for Vision. Pour de plus amples informations, veuillez consulter Création de ressources Amazon Lookout for Vision avec AWS CloudFormation.

Pour consulter vos projets, consultez Afficher vos projets ou ouvrez leUtilisation du tableau de bord Amazon Lookout for Vision. Pour supprimer un modèle, voirSuppression d'un modèle.

Rubriques

- Création d'un projet (console)
- Création d'un projet (kit SDK)

Création d'un projet (console)

La procédure suivante explique comment créer un projet à l'aide de la console.

Pour créer un projet (console)

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Dans le volet de navigation de gauche, choisissez Projets.
3. Sélectionnez Create a project (Créer un projet).
4. Dans Project name (Nom de projet), saisissez un nom pour votre projet.
5. Sélectionnez Create a project (Créer un projet). La page de détails de ce projet s'affiche.
6. Suivez les étapes décrites Création de votre jeu de données pour créer votre ensemble de données.

Création d'un projet (kit SDK)

Vous utilisez cette CreateProjectopération pour créer un projet Amazon Lookout for Vision. Le formulaire de réponse `CreateProject` inclut le nom du projet et le nom de ressource Amazon (ARN) du projet. Ensuite, appelez CreateDatasetpour ajouter un ensemble de données de formation et de test à votre projet. Pour de plus amples informations, veuillez consulter Création d'un ensemble de données à l'aide d'un fichier manifeste (SDK).

Pour consulter les projets que vous avez créés dans un projet, appelez`ListProjects`. Pour de plus amples informations, veuillez consulter Afficher vos projets.

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 l'exemple de code suivant pour créer un modèle.

   CLI

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

   ```
   aws lookoutvision create-project --project-name project name \
     --profile lookoutvision-access
   ```

   Python

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

   ```
       @staticmethod
       def create_project(lookoutvision_client, project_name):
           """
           Creates a new Lookout for Vision project.

           :param lookoutvision_client: A Boto3 Lookout for Vision client.
           :param  project_name: The name for the new project.
           :return project_arn: The ARN of the new project.
           """
           try:
               logger.info("Creating project: %s", project_name)
               response =
   lookoutvision_client.create_project(ProjectName=project_name)
               project_arn = response["ProjectMetadata"]["ProjectArn"]
               logger.info("project ARN: %s", project_arn)
           except ClientError:
               logger.exception("Couldn't create project %s.", project_name)
               raise
           else:
               return project_arn
   ```

Java V2

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

```
/**
 * Creates an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that you want to create.
 * @return ProjectMetadata Metadata information about the created project.
 */
public static ProjectMetadata createProject(LookoutVisionClient lfvClient,
 String projectName)
                throws LookoutVisionException {

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

        CreateProjectResponse response =
 lfvClient.createProject(createProjectRequest);

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

        return response.projectMetadata();

}
```

3. Suivez les étapes décrites Création d'un ensemble de données à l'aide d'un fichier manifeste Amazon SageMaker AI Ground Truth pour créer votre ensemble de données.

Création de votre jeu de données

Un jeu de données contient les images et les étiquettes attribuées que vous utilisez pour entraîner et tester un modèle. Vous créez l'ensemble de données pour votre projet avec la console Amazon Lookout for Vision ou avec CreateDatasetl'opération. Les images du jeu de données doivent être

étiquetées en fonction du type de modèle que vous souhaitez créer (classification d'images ou segmentation d'images).

Rubriques

- Préparation d'images pour un jeu de données
- Création du jeu de données
- Création d'un jeu de données à partir d'images stockées sur votre ordinateur local
- Création d'un ensemble de données à partir d'images stockées dans un compartiment Amazon S3
- Création d'un ensemble de données à l'aide d'un fichier manifeste Amazon SageMaker AI Ground Truth

Préparation d'images pour un jeu de données

Vous avez besoin d'une collection d'images pour créer un jeu de données. Vos images doivent être des fichiers au format PNG ou JPEG. Le nombre et le type d'images dont vous avez besoin varient selon que votre projet comporte un seul jeu de données ou des ensembles de données d'entraînement et de test distincts.

Projet de jeu de données unique

Pour créer un modèle de classification d'images, vous devez disposer des éléments suivants pour commencer l'entraînement :

- Au moins 20 images d'objets normaux.
- Au moins 10 images d'objets anormaux.

Pour créer un modèle de segmentation d'image, vous devez disposer des éléments suivants pour commencer l'entraînement :

- Au moins 20 images de chaque type d'anomalie.
- Chaque image anormale (image avec des types d'anomalies présents) ne doit comporter qu'un seul type d'anomalie.
- Au moins 20 images d'objets normaux.

Projet de jeu de données de formation et de test distinct

Pour créer un modèle de classification d'images, vous avez besoin des éléments suivants :

- Au moins 10 images d'objets normaux dans le jeu de données d'apprentissage.
- Au moins 10 images d'objets normaux dans l'ensemble de données de test.
- Au moins 10 images d'objets anormaux dans l'ensemble de données de test.

Pour créer un modèle de segmentation d'image, vous avez besoin des éléments suivants :

- Chaque jeu de données nécessite au moins 10 images de chaque type d'anomalie.
- Chaque image anormale (image avec des types d'anomalies présents) ne doit contenir qu'un seul type d'anomalie.
- Chaque jeu de données doit contenir au moins 10 images d'objets normaux.

Pour créer un modèle de meilleure qualité, utilisez un nombre d'images supérieur au minimum. Si vous créez un modèle de segmentation, nous vous recommandons d'inclure des images présentant plusieurs types d'anomalies, mais celles-ci ne sont pas prises en compte dans le minimum dont Lookout for Vision a besoin pour commencer l'entraînement.

Vos images doivent représenter un seul type d'objet. Vous devez également disposer de conditions de capture d'image cohérentes, telles que le positionnement de la caméra, l'éclairage et la pose de l'objet.

Toutes les images des ensembles de données d'entraînement et de test doivent avoir les mêmes dimensions. Par la suite, les images que vous analyserez avec votre modèle entraîné doivent avoir les mêmes dimensions que les images des jeux de données d'apprentissage et de test. Pour de plus amples informations, veuillez consulter Détecter des anomalies dans une image.

Toutes les images d'entraînement et de test doivent être des images uniques, de préférence d'objets uniques. Les images normales doivent capturer les variations normales de l'objet analysé. Les images anormales doivent capturer un échantillon diversifié d'anomalies.

Amazon Lookout for Vision fournit des exemples d'images que vous pouvez utiliser. Pour de plus amples informations, veuillez consulter Ensemble de données de classification d'images.

Pour les limites d'image, voirQuotas.

Création du jeu de données

Lorsque vous créez le jeu de données pour votre projet, vous choisissez la configuration initiale du jeu de données de votre projet. Vous pouvez également choisir l'endroit d'où Lookout for Vision importe les images.

Choix d'une configuration de jeu de données pour votre projet

Lorsque vous créez le premier jeu de données de votre projet, vous choisissez l'une des configurations de jeu de données suivantes :

- Jeu de données unique : un projet de jeu de données unique utilise un seul jeu de données pour entraîner et tester votre modèle. L'utilisation d'un jeu de données unique simplifie la formation en laissant Amazon Lookout for Vision choisir les images de formation et de test. Pendant la formation, Amazon Lookout for Vision divise en interne le jeu de données en un ensemble de données de formation et un ensemble de données de test. Vous n'avez pas accès aux ensembles de données divisés. Nous recommandons d'utiliser un seul projet de jeu de données pour la plupart des scénarios.
- Ensembles de données d'entraînement et de test distincts : si vous souhaitez mieux contrôler l'entraînement, les tests et le réglage des performances, vous pouvez configurer votre projet pour disposer d'ensembles de données d'entraînement et de test distincts. Utilisez un ensemble de données de test distinct si vous souhaitez contrôler les images utilisées pour les tests ou si vous disposez déjà d'un ensemble d'images de référence que vous souhaitez utiliser.

Vous pouvez ajouter un ensemble de données de test à un projet de jeu de données unique existant. L'ensemble de données unique devient alors le jeu de données d'entraînement. Si vous supprimez le jeu de données de test d'un projet comportant des ensembles de données d'entraînement et de test distincts, le projet devient un projet de jeu de données unique. Pour de plus amples informations, veuillez consulter Suppression d’un jeu de données.

Importation d'images

Lorsque vous créez un jeu de données, vous choisissez d'où importer les images. Selon la façon dont vous importez les images, celles-ci sont peut-être déjà étiquetées. Si les images ne sont pas étiquetées après la création du jeu de données, consultezÉtiquetage des images.

Vous créez un jeu de données et importez ses images de l'une des manières suivantes :

- Importez des images depuis votre ordinateur local. Les images ne sont pas étiquetées. Vous pouvez ajouter des étiquettes à l'aide de la console Lookout for Vision.
- Importez des images depuis un compartiment S3. Amazon Lookout for Vision peut classer les images en utilisant les noms de dossiers pour étiqueter les images. À utiliser `normal` pour des images normales. À utiliser `anomaly` pour les images anormales. Vous ne pouvez pas attribuer automatiquement des étiquettes de segmentation.
- Importez un fichier manifeste Amazon SageMaker AI Ground Truth, qui inclut des images étiquetées. Vous pouvez créer et importer votre propre fichier manifeste. Si vous avez de nombreuses images, pensez à utiliser le service d'étiquetage SageMaker AI Ground Truth. Vous importez ensuite le fichier manifeste de sortie depuis le job Amazon SageMaker AI Ground Truth. Si nécessaire, vous pouvez utiliser la console Lookout for Vision pour ajouter ou modifier des libellés.

Si vous utilisez le AWS SDK, vous créez un ensemble de données avec un fichier manifeste Amazon SageMaker AI Ground Truth. Pour de plus amples informations, veuillez consulter Création d'un ensemble de données à l'aide d'un fichier manifeste Amazon SageMaker AI Ground Truth.

Si, après avoir créé votre jeu de données, vos images sont étiquetées, vous pouvez entraîner le modèle. Si les images ne sont pas étiquetées, ajoutez les étiquettes en fonction du type de modèle que vous souhaitez créer. Pour de plus amples informations, veuillez consulter Étiquetage des images.

Vous pouvez ajouter d'autres images à un jeu de données existant. Pour de plus amples informations, veuillez consulter Ajouter des images à votre jeu de données.

Création d'un jeu de données à partir d'images stockées sur votre ordinateur local

Vous pouvez créer un jeu de données en utilisant des images chargées directement depuis votre ordinateur. Vous pouvez charger jusqu'à 30 images à la fois. Dans cette procédure, vous pouvez créer un projet de jeu de données unique ou un projet avec des ensembles de données d'entraînement et de test distincts.

> **Note**
>
> Si vous venez de terminerCréation de votre projet, la console devrait afficher le tableau de bord de votre projet et vous n'avez pas besoin de suivre les étapes 1 à 3.

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

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Dans le volet de navigation de gauche, choisissez Projets.
3. Sur la page Projets, choisissez le projet auquel vous souhaitez ajouter un jeu de données.
4. Sur la page des détails du projet, choisissez Créer un jeu de données.
5. Choisissez l'onglet Ensemble de données unique ou l'onglet Ensembles de données d'entraînement et de test séparés et suivez les étapes.

   Single dataset

   a. Dans la section Configuration du jeu de données, choisissez Create a single dataset.
   b. Dans la section Configuration de la source d'image, choisissez Télécharger des images depuis votre ordinateur.
   c. Choisissez Créer un jeu de données.
   d. Sur la page du jeu de données, choisissez Ajouter des images.
   e. 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.
   f. Choisissez Charger des images.

   Separate training and test datasets

   a. Dans la section Configuration du jeu de données, choisissez Créer un ensemble de données d'entraînement et un ensemble de données de test.
   b. Dans la section Détails du jeu de données d’entraînement, choisissez Charger des images depuis votre ordinateur.
   c. 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.

d. Choisissez Créer un jeu de données. Une page s'affiche avec un onglet Entraînement et un onglet Test pour vos jeux de données respectifs.

e. Choisissez Actions, puis sélectionnez Ajouter des images au jeu de données d'entraînement.

f. 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.

g. Choisissez Charger des images.

h. Répétez les étapes 5e à 5g. Pour l'étape 5e, choisissez Actions, puis sélectionnez Ajouter des images au jeu de données de test.

6. Suivez les étapes décrites dans Étiquetage des images pour étiqueter les images.
7. Suivez les étapes décrites dans Entraînement de votre modèle pour entraîner le modèle.

Création d'un ensemble de données à partir d'images stockées dans un compartiment Amazon S3

Vous pouvez créer un ensemble de données à l'aide d'images stockées dans un compartiment Amazon S3. Avec cette option, vous pouvez utiliser la structure de dossiers de votre compartiment Amazon S3 pour classer automatiquement vos images. Vous pouvez stocker les images dans le compartiment de console ou dans un autre compartiment Amazon S3 de votre compte.

Configuration des dossiers pour l'étiquetage automatique

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. Les dossiers doivent être un enfant du chemin du dossier Amazon S3 que vous spécifiez dans l'URI S3 lorsque vous créez l'ensemble de données.

Le `train` dossier suivant contient les images d'exemple de Getting Started. Si vous spécifiez l'emplacement du dossier Amazon S3`S3-bucket/circuitboard/train/`, l'étiquette est attribuée `normal` aux images du dossier`Normal`. L'étiquette est attribuée `anomaly` aux images du dossier`Anomaly`. Le nom des dossiers enfants situés plus loin dans la structure de dossiers n'est pas utilisé pour étiqueter les images.

```
S3-bucket
  ### circuitboard
    ### train
        ### anomaly
            ### train-anomaly_1.jpg
            ### train-anomaly_2.jpg
            ### .
            ### .
        ### normal
            ### train-normal_1.jpg
            ### train-normal_2.jpg
            ### .
            ### .
```

Création d'un ensemble de données à partir d'images provenant d'un compartiment Amazon S3

La procédure suivante crée un ensemble de données à l'aide des images d'exemple de classification stockées dans un compartiment Amazon S3. Pour utiliser vos propres images, créez la structure de dossiers décrite dansConfiguration des dossiers pour l'étiquetage automatique.

La procédure indique également comment créer un projet de jeu de données unique ou un projet utilisant des ensembles de données d'entraînement et de test distincts.

Si vous ne choisissez pas d'étiqueter automatiquement vos images, vous devez les étiqueter une fois les ensembles de données créés. Pour de plus amples informations, veuillez consulter Classification des images (console).

> **Note**
>
> Si vous venez de terminerCréation de votre projet, la console devrait afficher le tableau de bord de votre projet et vous n'avez pas besoin de suivre les étapes 1 à 4.

Pour créer un ensemble de données à partir d'images stockées dans un compartiment Amazon S3

1. Si ce n'est pas déjà fait, téléchargez les images de démarrage dans votre compartiment Amazon S3. Pour de plus amples informations, veuillez consulter Ensemble de données de classification d'images.

2. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
3. Dans le volet de navigation de gauche, choisissez Projets.
4. Sur la page Projets, choisissez le projet auquel vous souhaitez ajouter un jeu de données. La page de détails de ce projet s'affiche.
5. Choisissez Créer un jeu de données. La page Créer un jeu de données s'affiche.

> **Tip**
>
> Si vous suivez les instructions de mise en route, choisissez Créer un ensemble de données d'entraînement et un ensemble de données de test.

6. Choisissez l'onglet Ensemble de données unique ou l'onglet Ensembles de données d'entraînement et de test séparés et suivez les étapes.

   Single dataset

   a. Dans la section Configuration du jeu de données, choisissez Create a single dataset.
   b. Entrez les informations relatives aux étapes 7 à 9 dans la section Configuration de la source d'image.

   Separate training and test datasets

   a. Dans la section Configuration du jeu de données, choisissez Créer un ensemble de données d'entraînement et un ensemble de données de test.
   b. Pour votre jeu de données d'entraînement, entrez les informations relatives aux étapes 7 à 9 dans la section Détails de l'ensemble de données d'entraînement.
   c. Pour votre jeu de données de test, entrez les informations relatives aux étapes 7 à 9 dans la section Détails de l'ensemble de données de test.

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

7. Choisissez Importer des images depuis le compartiment Amazon S3.

8. Dans URI S3, entrez l'emplacement du compartiment Amazon S3 et le chemin du dossier. Modifiez `bucket` en spécifiant le nom de votre compartiment Amazon S3.

   a. Si vous créez un projet de jeu de données unique ou un ensemble de données d'entraînement, entrez ce qui suit :

   ```
   s3://bucket/circuitboard/train/
   ```

   b. Si vous créez un ensemble de données de test, entrez ce qui suit :

   ```
   s3://bucket/circuitboard/test/
   ```

9. Choisissez Associer automatiquement des étiquettes aux images en fonction du dossier.
10. Choisissez Créer un jeu de données. Une page de jeu de données s'ouvre avec vos images étiquetées.
11. Suivez les étapes décrites dans Entraînement de votre modèle pour entraîner le modèle.

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

Un fichier manifeste contient des informations sur les images et les étiquettes d'image que vous pouvez utiliser pour entraîner et tester un modèle. Vous pouvez stocker un fichier manifeste dans un compartiment Amazon S3 et l'utiliser pour créer un ensemble de données. Vous pouvez créer votre propre fichier manifeste ou utiliser un fichier manifeste existant, tel que le résultat d'une tâche Amazon SageMaker AI Ground Truth.

Rubriques

- Utilisation d'une tâche Amazon Sagemaker Ground Truth
- Création d'un fichier manifeste

Utilisation d'une tâche Amazon Sagemaker Ground Truth

L'étiquetage des images peut prendre beaucoup de temps. Par exemple, cela peut prendre des dizaines de secondes pour dessiner avec précision un masque autour d'une anomalie. Si vous avez des centaines d'images, leur étiquetage peut prendre plusieurs heures. Au lieu d'étiqueter vous-même les images, pensez à utiliser Amazon SageMaker 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 des employés internes du secteur privé pour créer un ensemble d'images étiqueté. Pour plus d'informations, consultez Utiliser Amazon SageMaker AI Ground Truth pour étiqueter les données.

L'utilisation d'Amazon Mechanical Turk est payante. En outre, plusieurs jours peuvent être nécessaires pour terminer une tâche d'étiquetage sur Amazon Ground Truth. En cas de problème de coût ou si vous devez entraîner rapidement votre modèle, nous vous recommandons d'utiliser la console Amazon Lookout for Vision pour étiqueter vos images.

Vous pouvez utiliser une tâche d'étiquetage Amazon SageMaker AI Ground Truth pour étiqueter des images adaptées aux modèles de classification d'images et aux modèles de segmentation d'images. Une fois le travail terminé, vous utilisez le fichier manifeste de sortie pour créer un ensemble de données Amazon Lookout for Vision.

Classification d’images

Pour étiqueter des images pour un modèle de classification d'images, créez une tâche d'étiquetage pour une tâche de classification d'images (étiquette unique).

Segmentation d'images

Pour étiqueter des images pour un modèle de segmentation d'image, créez une tâche d'étiquetage pour une tâche de classification d'images (étiquette unique). Ensuite, enchaînez la tâche pour créer une tâche d'étiquetage pour une tâche de segmentation sémantique des images.

Vous pouvez également utiliser une tâche d'étiquetage pour créer un fichier manifeste partiel pour un modèle de segmentation d'image. Par exemple, vous pouvez classer des images à l'aide d'une tâche de classification d'images (étiquette unique). Après avoir créé un jeu de données Lookout for Vision avec le résultat de la tâche, utilisez la console Amazon Lookout for Vision pour ajouter des masques de segmentation et des étiquettes d'anomalie aux images du jeu de données.

Étiqueter des images avec Amazon SageMaker AI Ground Truth

La procédure suivante explique comment étiqueter des images à l'aide des tâches d'étiquetage d'images Amazon SageMaker AI Ground Truth. La procédure crée un fichier manifeste de classification d'images et, éventuellement, enchaîne la tâche d'étiquetage d'image pour créer un fichier manifeste de segmentation d'image. Si vous souhaitez que votre projet dispose d'un ensemble de données de test distinct, répétez cette procédure pour créer le fichier manifeste pour l'ensemble de données de test.

Pour étiqueter des images avec Amazon SageMaker AI Ground Truth (console)

1. Créez une tâche Ground Truth pour une tâche de classification d'images (étiquette unique) en suivant les instructions de la section Create a Labeling Job (console).

   a. Pour l'étape 10, choisissez Image dans le menu déroulant des catégories de tâches, puis choisissez Classification des images (étiquette unique) comme type de tâche.

   b. Pour l'étape 16, dans la section de l'outil d'étiquetage de classification des images (étiquette unique), ajoutez deux étiquettes : normale et anomalie.

2. Attendez que le personnel ait fini de classer vos images.

3. Si vous créez un jeu de données pour un modèle de segmentation d'image, procédez comme suit. Sinon, passez à l'étape 4.

   a. Dans la console Amazon SageMaker AI Ground Truth, ouvrez la page des tâches d'étiquetage.

   b. Choisissez le job que vous avez créé précédemment. Cela active le menu Actions.

   c. Dans le menu Actions, choisissez Chain (Chaîner). La page des détails de la tâche s'ouvre.

   d. Dans le type de tâche, choisissez la segmentation sémantique.

   e. Choisissez Suivant.

   f. Dans la section Outil d'étiquetage de segmentation sémantique, ajoutez des étiquettes d'anomalie pour chaque type d'anomalie que vous souhaitez que votre modèle détecte.

   g. Sélectionnez Create (Créer).

   h. Attendez que le personnel étiquette vos images.

4. Ouvrez la console Ground Truth et ouvrez la page des tâches d'étiquetage.

5. Si vous créez un modèle de classification d'images, choisissez la tâche que vous avez créée à l'étape 1. Si vous créez un modèle de segmentation d'image, choisissez la tâche créée à l'étape 3.

6. Dans le résumé des tâches d'étiquetage, ouvrez l'emplacement S3 dans Emplacement du jeu de données en sortie. Notez l'emplacement du fichier manifeste, qui devrait êtres3://*output-dataset-location*/manifests/output/output.manifest.

7. Répétez cette procédure si vous souhaitez créer un fichier manifeste pour un ensemble de données de test. Dans le cas contraire, suivez les instructions de la Création du jeu de données section pour créer un ensemble de données avec le fichier manifeste.

Création du jeu de données

Utilisez cette procédure pour créer un ensemble de données dans un projet Lookout for Vision avec le fichier manifeste que vous avez noté à l'étape 6 de Étiqueter des images avec Amazon SageMaker AI Ground Truth. Le fichier manifeste crée le jeu de données d'entraînement pour un seul projet de jeu de données. Si vous souhaitez que votre projet dispose d'un ensemble de données de test distinct, vous pouvez exécuter une autre tâche Amazon SageMaker AI Ground Truth afin de créer un fichier manifeste pour l'ensemble de données de test. Vous pouvez également créer le fichier manifeste vous-même. Vous pouvez également importer des images dans votre ensemble de données de test depuis un compartiment Amazon S3 ou depuis votre ordinateur local. (Les images peuvent avoir besoin d'être étiquetées avant que vous puissiez entraîner le modèle).

Cette procédure part du principe que votre projet ne contient aucun ensemble de données.

Pour créer un jeu de données avec Lookout for Vision (console)

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Choisissez Démarrer.
3. Dans le volet de navigation de gauche, choisissez Projets.
4. Choisissez le projet que vous souhaitez ajouter pour l'utiliser avec le fichier manifeste.
5. Dans la section Fonctionnement, sélectionnez Créer un ensemble de données.
6. Choisissez l'onglet Ensemble de données unique ou l'onglet Ensembles de données d'entraînement et de test séparés et suivez les étapes.

   Single dataset

   1. Choisissez Créer un jeu de données unique.
   2. Dans la section Configuration de la source d'image, choisissez Importer des images étiquetées par SageMaker Ground Truth.
   3. Pour l'emplacement du fichier .manifest, entrez l'emplacement du fichier manifeste que vous avez noté à l'étape 6 de Étiqueter des images avec Amazon SageMaker AI Ground Truth.

Separate training and test datasets

1. Choisissez Créer un ensemble de données d'entraînement et un ensemble de données de test.
2. Dans la section des détails du jeu de données d'entraînement, choisissez Importer des images étiquetées par SageMaker Ground Truth.
3. Dans l'emplacement du fichier .manifest, l'emplacement du fichier manifeste que vous avez noté à l'étape 6 de Étiqueter des images avec Amazon SageMaker AI Ground Truth.
4. Dans la section Détails du jeu de données de test, choisissez Importer des images étiquetées par SageMaker Ground Truth.
5. Dans l'emplacement du fichier .manifest, l'emplacement du fichier manifeste que vous avez noté à l'étape 6 de Étiqueter des images avec Amazon SageMaker AI Ground Truth. N'oubliez pas que vous avez besoin d'un fichier manifeste distinct pour l'ensemble de données de test.

7. Sélectionnez Envoyer.
8. Suivez les étapes décrites dans Entraînement de votre modèle pour entraîner le modèle.

Création d'un fichier manifeste

Vous pouvez créer un ensemble de données 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. Il existe différents formats pour la classification et la segmentation des images. Les fichiers manifestes doivent être codés en UTF-8.

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

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

Vous pouvez créer un fichier manifeste à l'aide de code. Le bloc-notes Python Amazon Lookout for Vision Lab explique comment créer un fichier manifeste de classification d'images pour les exemples d'images du circuit imprimé. Vous pouvez également utiliser l'exemple de code Datasets dans le référentiel d'exemples de AWS code. Vous pouvez facilement créer un fichier manifeste à l'aide d'un fichier CSV (valeurs séparées par des virgules). Pour de plus amples informations, veuillez consulter Création d'un fichier manifeste de classification à partir d'un fichier CSV.

Rubriques

- Définition de lignes JSON pour la classification des images
- Définition de lignes JSON pour la segmentation d'images
- Création d'un fichier manifeste de classification à partir d'un fichier CSV
- Création d'un ensemble de données avec un fichier manifeste (console)
- Création d'un ensemble de données à l'aide d'un fichier manifeste (SDK)

Définition de lignes JSON pour la classification des images

Vous définissez une ligne JSON pour chaque image que vous souhaitez utiliser dans un fichier manifeste Amazon Lookout for Vision. Si vous souhaitez créer un modèle de classification, la ligne JSON doit inclure une classification d'image normale ou anormale. Une ligne JSON est au format SageMaker AI Ground Truth Classification Job Output. Un fichier manifeste est composé d'une ou de plusieurs lignes JSON, une pour chaque image que vous souhaitez importer.

Pour créer un fichier manifeste pour les images classifiées

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://lookoutvision-console-us-east-1-nnnnnnnnnn/gt-job/manifest/
IMG_1133.png","anomaly-label":1,"anomaly-label-metadata":{"confidence":0.95,"job-
name":"labeling-job/testclconsolebucket","class-name":"normal","human-
annotated":"yes","creation-date":"2020-04-15T20:17:23.433061","type":"groundtruth/
image-classification"}}
```

3. Enregistrez le fichier.

> **Note**
>
> 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 de plus amples informations, veuillez consulter Création d'un fichier manifeste.

Lignes JSON de classification

Dans cette section, vous allez apprendre à créer une ligne JSON qui classe une image comme normale ou anormale.

Ligne JSON d'anomalie

La ligne JSON suivante montre une image étiquetée comme une anomalie. Notez que la valeur de `class-name` est`anomaly`.

```
{
    "source-ref": "s3: //bucket/image/anomaly/abnormal-1.jpg",
    "anomaly-label-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/auto-label",
        "class-name": "anomaly",
        "human-annotated": "yes",
        "creation-date": "2020-11-10T03:37:09.600",
        "type": "groundtruth/image-classification"
    },
    "anomaly-label": 1
}
```

Ligne JSON normale

La ligne JSON suivante montre une image étiquetée comme normale. Notez que la valeur de `class-name` est`normal`.

```
{
    "source-ref": "s3: //bucket/image/normal/2020-10-20_12-14-55_613.jpeg",
```

```
    "anomaly-label-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/auto-label",
        "class-name": "normal",
        "human-annotated": "yes",
        "creation-date": "2020-11-10T03:37:09.603",
        "type": "groundtruth/image-classification"
    },
    "anomaly-label": 0
}
```

Clés et valeurs de ligne JSON

Les informations suivantes décrivent les clés et les valeurs d'une ligne JSON Amazon Lookout for Vision.

Source-réf.

(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.

étiquette d'anomalie

(Obligatoire) Attribut de l'étiquette. Utilisez la clé `anomaly-label` ou un autre nom de clé de votre choix. La valeur clé (`0`dans l'exemple précédent) est requise par Amazon Lookout for Vision, mais elle n'est pas utilisée. Le manifeste de sortie créé par Amazon Lookout for Vision convertit la valeur en 1 pour une image anormale et la valeur `0` en pour une image normale. La valeur de `class-name` détermine si l'image est normale ou anormale.

Les métadonnées correspondantes doivent être identifiées par le nom du champ, nom auquel vous devez ajouter -metadata. Par exemple, `"anomaly-label-metadata"`.

anomaly-label-metadata

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

confidence

(Facultatif) Actuellement non utilisé par Amazon Lookout for Vision. Si vous spécifiez une valeur, utilisez une valeur de1.

job-name

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

class-name

(Obligatoire) Si l'image contient un contenu normal, spécifiez`normal`, sinon spécifiez`anomaly`. Si la valeur de `class-name` est une autre valeur, l'image est ajoutée au jeu de données en tant qu'image non étiquetée. Pour étiqueter une image, voirAjouter des images à votre jeu de données.

human-annotated

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

creation-date

(Facultatif) Date et heure de création de l'étiquette en temps universel coordonné (UTC).

type

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

Définition de lignes JSON pour la segmentation d'images

Vous définissez une ligne JSON pour chaque image que vous souhaitez utiliser dans un fichier manifeste Amazon Lookout for Vision. Si vous souhaitez créer un modèle de segmentation, la ligne JSON doit inclure les informations de segmentation et de classification de l'image. Un fichier manifeste est composé d'une ou de plusieurs lignes JSON, une pour chaque image que vous souhaitez importer.

Pour créer un fichier manifeste pour les images segmentées

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://path-to-image","anomaly-label":1,"anomaly-label-metadata":
{"class-name":"anomaly","creation-date":"2021-10-12T14:16:45.668","human-
annotated":"yes","job-name":"labeling-job/classification-job","type":"groundtruth/
image-classification","confidence":1},"anomaly-mask-ref":"s3://path-to-
```

```
image","anomaly-mask-ref-metadata":{"internal-color-map":{"0":{"class-
name":"BACKGROUND","hex-color":"#ffffff","confidence":0.0},"1":{"class-
name":"scratch","hex-color":"#2ca02c","confidence":0.0},"2":{"class-
name":"dent","hex-color":"#1f77b4","confidence":0.0}},"type":"groundtruth/
semantic-segmentation","human-annotated":"yes","creation-
date":"2021-11-23T20:31:57.758889","job-name":"labeling-job/segmentation-job"}}
```

3. Enregistrez le fichier.

> **Note**
>
> 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 de plus amples informations, veuillez consulter Création d'un fichier manifeste.

Lignes JSON de segmentation

Dans cette section, vous apprendrez à créer une ligne JSON qui inclut des informations de segmentation et de classification pour une image.

La ligne JSON suivante montre une image contenant des informations de segmentation et de classification. `anomaly-label-metadata`contient des informations de classification. `anomaly-mask-ref`et `anomaly-mask-ref-metadata` contiennent des informations de segmentation.

```
{
    "source-ref": "s3://path-to-image",
    "anomaly-label": 1,
    "anomaly-label-metadata": {
        "class-name": "anomaly",
        "creation-date": "2021-10-12T14:16:45.668",
        "human-annotated": "yes",
        "job-name": "labeling-job/classification-job",
        "type": "groundtruth/image-classification",
        "confidence": 1
    },
    "anomaly-mask-ref": "s3://path-to-image",
    "anomaly-mask-ref-metadata": {
        "internal-color-map": {
            "0": {
                "class-name": "BACKGROUND",
```

```
                "hex-color": "#ffffff",
                "confidence": 0.0
            },
            "1": {
                "class-name": "scratch",
                "hex-color": "#2ca02c",
                "confidence": 0.0
            },
            "2": {
                "class-name": "dent",
                "hex-color": "#1f77b4",
                "confidence": 0.0
            }
        },
        "type": "groundtruth/semantic-segmentation",
        "human-annotated": "yes",
        "creation-date": "2021-11-23T20:31:57.758889",
        "job-name": "labeling-job/segmentation-job"
    }
}
```

Clés et valeurs de ligne JSON

Les informations suivantes décrivent les clés et les valeurs d'une ligne JSON Amazon Lookout for Vision.

Source-réf.

(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.

étiquette d'anomalie

(Obligatoire) Attribut de l'étiquette. Utilisez la clé `anomaly-label` ou un autre nom de clé de votre choix. La valeur clé (1dans l'exemple précédent) est requise par Amazon Lookout for Vision, mais elle n'est pas utilisée. Le manifeste de sortie créé par Amazon Lookout for Vision convertit la valeur en 1 pour une image anormale et la valeur `0` en pour une image normale. La valeur de `class-name` détermine si l'image est normale ou anormale.

Les métadonnées correspondantes doivent être identifiées par le nom du champ, nom auquel vous devez ajouter -metadata. Par exemple, `"anomaly-label-metadata"`.

anomaly-label-metadata

(Obligatoire) Métadonnées relatives à l'attribut de l'étiquette. Contient des informations de classification. 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

(Facultatif) Actuellement non utilisé par Amazon Lookout for Vision. Si vous spécifiez une valeur, utilisez une valeur de1.

job-name

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

class-name

(Obligatoire) Si l'image contient un contenu normal, spécifiez`normal`, sinon spécifiez`anomaly`. Si la valeur de `class-name` est une autre valeur, l'image est ajoutée au jeu de données en tant qu'image non étiquetée. Pour étiqueter une image, voirAjouter des images à votre jeu de données.

human-annotated

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

creation-date

(Facultatif) Date et heure de création de l'étiquette en temps universel coordonné (UTC).

type

(Obligatoire) Type de traitement à appliquer à l'image. Utilisez la valeur`"groundtruth/image-classification"`.

anomaly-mask-ref

(Obligatoire) Emplacement de l'image du masque sur Amazon S3. `anomaly-mask-ref`Utilisez-le comme nom de clé ou utilisez le nom de clé de votre choix. La clé doit se terminer par`-ref`. L'image du masque doit contenir des masques colorés pour chaque type `internal-color-map` d'anomalie. Le format est `"s3://`*`BUCKET`*`/`*`OBJECT_PATH`*`"`. Les images d'un jeu de données importé doivent être stockées dans le même compartiment Amazon S3. L'image du masque doit être au format PNG (Portable Network Graphic).

anomaly-mask-ref-metadata

(Obligatoire) Métadonnées de segmentation pour l'image. `anomaly-mask-ref-metadata`Utilisez-le comme nom de clé ou utilisez le nom de clé de votre choix. Le nom de la clé doit se terminer par-`ref-metadata`.

internal-color-map

(Obligatoire) Une carte de couleurs correspondant à des types d'anomalies individuels. Les couleurs doivent correspondre à celles de l'image du masque (`anomaly-mask-ref`).

key

(Obligatoire) La clé d'accès à la carte. L'entrée `0` doit contenir le nom de classe BACKGROUND qui représente les zones situées en dehors des anomalies sur l'image.

class-name

(Obligatoire) Le nom du type d'anomalie, tel qu'une égratignure ou une bosse.

couleur hexadécimale

(Obligatoire) La couleur hexadécimale du type d'anomalie, telle que. `#2ca02c` La couleur doit correspondre à une couleur de`anomaly-mask-ref`. La valeur du type `BACKGROUND` d'anomalie est toujours`#ffffff`.

confidence

(Obligatoire) Actuellement non utilisé par Amazon Lookout for Vision, mais une valeur flottante est requise.

human-annotated

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

creation-date

(Facultatif) Date et heure UTC (temps universel coordonné) auxquelles les informations de segmentation ont été créées.

type

(Obligatoire) Type de traitement à appliquer à l'image. Utilisez la valeur`"groundtruth/semantic-segmentation"`.

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

Cet exemple de script Python simplifie la création d'un fichier manifeste de classification en utilisant un fichier CSV (valeurs séparées par des virgules) pour étiqueter les images. C'est vous qui créez le fichier CSV.

Un fichier manifeste décrit les images utilisées pour entraîner un modèle Un fichier manifeste est composé d'une ou plusieurs lignes JSON. Chaque ligne JSON décrit une seule image. Pour de plus amples informations, veuillez consulter Définition de lignes JSON pour la classification des images.

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 de votre fichier CSV inclut l'emplacement S3 d'une image et la classification des anomalies pour l'image (`normalouanomaly`). Chaque ligne correspond à une ligne JSON dans le fichier manifeste.

Par exemple, le fichier CSV suivant décrit certaines des images contenues dans les exemples d'images.

```
s3://s3bucket/circuitboard/train/anomaly/train-anomaly_1.jpg,anomaly
s3://s3bucket/circuitboard/train/anomaly/train-anomaly_10.jpg,anomaly
s3://s3bucket/circuitboard/train/anomaly/train-anomaly_11.jpg,anomaly
s3://s3bucket/circuitboard/train/normal/train-normal_1.jpg,normal
s3://s3bucket/circuitboard/train/normal/train-normal_10.jpg,normal
s3://s3bucket/circuitboard/train/normal/train-normal_11.jpg,normal
```

Le script génère des lignes JSON pour chaque ligne. Par exemple, voici la ligne JSON pour la première ligne (`s3://s3bucket/circuitboard/train/anomaly/train-anomaly_1.jpg,anomaly`).

```
{"source-ref": "s3://s3bucket/csv_test/train_anomaly_1.jpg","anomaly-label":
 1,"anomaly-label-metadata": {"confidence": 1,"job-name": "labeling-job/anomaly-
classification","class-name": "anomaly","human-annotated": "yes","creation-date":
 "2022-02-04T22:47:07","type": "groundtruth/image-classification"}}
```

Si votre fichier CSV n'inclut pas le chemin Amazon S3 pour les images, utilisez l'argument de ligne de `--s3-path` commande pour spécifier le chemin Amazon S3 vers les images.

Avant de créer le fichier manifeste, le script vérifie la présence d'images dupliquées dans le fichier CSV ainsi que les classifications d'images qui ne le sont pas `normal` ou`anomaly`. Si des doublons ou des erreurs de classification d'images sont détectées, le script effectue les opérations suivantes :

- Enregistre la première entrée d'image valide pour toutes les images d'un fichier CSV dédupliqué.
- Enregistre les occurrences de doublons d'une image dans le fichier d'erreurs.
- Enregistre les classifications d'images qui ne figurent pas `anomaly` dans le fichier d'erreurs `normal` ou qui ne le sont pas.
- Ne crée pas de fichier manifeste.

Le fichier d'erreurs inclut le numéro de ligne où une image dupliquée ou une erreur de classification est détectée dans le fichier CSV d'entrée. Utilisez le fichier CSV d'erreurs pour mettre à jour le fichier CSV d'entrée, puis réexécutez le script. Vous pouvez également utiliser le fichier CSV d'erreurs pour mettre à jour le fichier CSV dédupliqué, qui contient uniquement des entrées d'image uniques et des images sans erreur de classification des images. Réexécutez le script avec le fichier CSV dédupliqué mis à jour.

Si aucun doublon ou aucune erreur n'est détecté dans le fichier CSV d'entrée, le script supprime le fichier image CSV dédupliqué et le fichier d'erreurs, 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. Le script a été testé avec Python version 3.7.

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
Nom de l’image ou chemin Amazon S3 de l’image. Par exemple, `s3://s3bucket/circuitboard/train/anomaly/train-anomaly_10.jpg` . Vous ne pouvez pas avoir à la fois des images avec le chemin Amazon S3 et des images sans ce chemin.	Classification des anomalies pour l'image (`normal`ou`anomaly`).

   Par exemple : `s3://s3bucket/circuitboard/train/anomaly/image_10.jpg,anomaly` ou `image_11.jpg,normal`

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.
   - (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
"""
Purpose
Shows how to create an Amazon Lookout for Vision manifest file from a CSV file.
The CSV file format is image location,anomaly classification (normal or anomaly)
For example:
s3://s3bucket/circuitboard/train/anomaly/train_11.jpg,anomaly
s3://s3bucket/circuitboard/train/normal/train_1.jpg,normal

If necessary, use the bucket argument to specify the Amazon S3 bucket folder for
 the images.
"""

from datetime import datetime, timezone
import argparse
import logging
import csv
import os
import json

logger = logging.getLogger(__name__)


def check_errors(csv_file):
    """
    Checks for duplicate images and incorrect classifications in a CSV file.
    If duplicate images or invalid anomaly assignments are found, an errors CSV
 file
    and deduplicated CSV file are created. Only the first
    occurrence of a duplicate is recorded. Other duplicates are recorded in the
 errors file.
    :param csv_file: The source CSV file
```

```
    :return: True if errors or duplicates are found, otherwise false.
    """

    logger.info("Checking %s.", csv_file)

    errors_found = False
    errors_file = f"{os.path.splitext(csv_file)[0]}_errors.csv"
    deduplicated_file = f"{os.path.splitext(csv_file)[0]}_deduplicated.csv"

    with open(csv_file, 'r', encoding="UTF-8") as input_file,\
            open(deduplicated_file, 'w', encoding="UTF-8") as dedup,\
            open(errors_file, 'w', encoding="UTF-8") as errors:

        reader = csv.reader(input_file, delimiter=',')
        dedup_writer = csv.writer(dedup)
        error_writer = csv.writer(errors)
        line = 1
        entries = set()
        for row in reader:

            # Skip empty lines.
            if not ''.join(row).strip():
                continue

            # Record any incorrect classifications.
            if not row[1].lower() == "normal" and not row[1].lower() == "anomaly":
                error_writer.writerow(
                    [line, row[0], row[1], "INVALID_CLASSIFICATION"])
                errors_found = True

            # Write first image entry to dedup file and record duplicates.
            key = row[0]
            if key not in entries:
                dedup_writer.writerow(row)
                entries.add(key)
            else:
                error_writer.writerow([line, row[0], row[1], "DUPLICATE"])
                errors_found = True
            line += 1

    if errors_found:
        logger.info("Errors found check %s.", errors_file)
    else:
        os.remove(errors_file)
```

```
        os.remove(deduplicated_file)

    return errors_found


def create_manifest_file(csv_file, manifest_file, s3_path):
    """
    Read a CSV file and create an Amazon Lookout for Vision 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 Amazon S3 path to the folder that contains the images.
    """
    logger.info("Processing CSV file %s.", csv_file)

    image_count = 0
    anomalous_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 the CSV file.
        for row in image_classifications:
            # Skip empty lines.
            if not ''.join(row).strip():
                continue

            source_ref = str(s3_path) + row[0]
            classification = 0

            if row[1].lower() == 'anomaly':
                classification = 1
                anomalous_count += 1

           # Create the JSON line.
            json_line = {}
            json_line['source-ref'] = source_ref
            json_line['anomaly-label'] = str(classification)

            metadata = {}
            metadata['confidence'] = 1
```

```
            metadata['job-name'] = "labeling-job/anomaly-classification"
            metadata['class-name'] = row[1]
            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['anomaly-label-metadata'] = metadata

            output_file.write(json.dumps(json_line))
            output_file.write('\n')
            image_count += 1

    logger.info("Finished creating manifest file %s.\n"
                "Images: %s\nAnomalous: %s",
                manifest_file,
                image_count,
                anomalous_count)
    return image_count, anomalous_count


def add_arguments(parser):
    """
    Add 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 Amazon S3 bucket and folder path for the images."
        " If not supplied, column 1 is assumed to include the Amazon 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 = ""

        csv_file = args.csv_file
        csv_file_no_extension = os.path.splitext(csv_file)[0]
        manifest_file = csv_file_no_extension + '.manifest'

        # Create manifest file if there are no duplicate images.
        if check_errors(csv_file):
            print(f"Issues found. Use {csv_file_no_extension}_errors.csv "\
                "to view duplicates and errors.")
            print(f"{csv_file}_deduplicated.csv contains the first"\
                "occurrence of a duplicate.\n"
                  "Update as necessary with the correct information.")
            print(f"Re-run the script with
{csv_file_no_extension}_deduplicated.csv")
        else:
            print('No duplicates found. Creating manifest file.')

            image_count, anomalous_count = create_manifest_file(csv_file,
manifest_file, s3_path)

            print(f"Finished creating manifest file: {manifest_file} \n")

            normal_count = image_count-anomalous_count
            print(f"Images processed: {image_count}")
            print(f"Normal: {normal_count}")
            print(f"Anomalous: {anomalous_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. En cas de duplication d'images ou d'erreurs de classification :

a. Utilisez le fichier d'erreurs pour mettre à jour le fichier CSV dédupliqué ou le fichier CSV d'entrée.

b. Réexécutez le script avec le fichier CSV dédupliqué mis à jour ou le fichier CSV d'entrée mis à jour.

5. Si vous prévoyez d'utiliser un ensemble de données de test, répétez les étapes 1 à 4 pour créer un fichier manifeste pour votre ensemble de données de test.
6. Si nécessaire, copiez les images de votre ordinateur vers le chemin du compartiment Amazon S3 que vous avez spécifié dans la colonne 1 du fichier CSV (ou indiqué dans la ligne de `--s3-path` commande). Pour copier les images, entrez la commande suivante à l'invite de commande.

```
aws s3 cp --recursive your-local-folder/ s3://your-target-S3-location/
```

7. Suivez les instructions de l'adresse Création d'un ensemble de données avec un fichier manifeste (console) pour créer un jeu de données. Si vous utilisez le AWS SDK, consultezCréation d'un ensemble de données à l'aide d'un fichier manifeste (SDK).

Création d'un ensemble de données avec un fichier manifeste (console)

La procédure suivante explique comment créer un ensemble de données d'entraînement ou de test en important un fichier manifeste au format SageMaker AI stocké dans un compartiment Amazon S3.

Après avoir créé le jeu de données, vous pouvez y ajouter d'autres images ou ajouter des étiquettes aux images. Pour de plus amples informations, veuillez consulter Ajouter des images à votre jeu de données.

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

1. Créez ou utilisez un fichier manifeste au format AI Ground Truth SageMaker compatible avec Amazon Lookout for Vision existant. Pour de plus amples informations, veuillez consulter Création d'un fichier manifeste.
2. Connectez-vous à la console Amazon S3 AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/s3/.
3. Dans un compartiment Amazon S3, créez un dossier contenant votre fichier manifeste.
4. Téléchargez votre fichier manifeste dans le dossier que vous venez de créer.

5. Dans le compartiment Amazon S3, créez un dossier pour stocker vos images.
6. Chargez vos images dans le dossier que vous venez de créer.

   > **Important**
   >
   > La valeur du `source-ref` champ de chaque ligne JSON doit correspondre aux images du dossier.

7. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
8. Choisissez Démarrer.
9. Dans le volet de navigation de gauche, choisissez Projets.
10. Choisissez le projet que vous souhaitez ajouter pour l'utiliser avec le fichier manifeste.
11. Dans la section Fonctionnement, sélectionnez Créer un ensemble de données.
12. Choisissez l'onglet Ensemble de données unique ou l'onglet Ensembles de données d'entraînement et de test séparés et suivez les étapes.

    Single dataset

    1. Choisissez Créer un jeu de données unique.
    2. Dans la section Configuration de la source d'image, choisissez Importer des images étiquetées par SageMaker Ground Truth.
    3. Pour l'emplacement du fichier .manifest, entrez l'emplacement de votre fichier manifeste.

    Separate training and test datasets

    1. Choisissez Créer un ensemble de données d'entraînement et un ensemble de données de test.
    2. Dans la section des détails du jeu de données d'entraînement, choisissez Importer des images étiquetées par SageMaker Ground Truth.
    3. Dans l'emplacement du fichier .manifest, entrez l'emplacement de votre fichier manifeste d'entraînement.
    4. Dans la section Détails du jeu de données de test, choisissez Importer des images étiquetées par SageMaker Ground Truth.

5. Dans l'emplacement du fichier .manifest, entrez l'emplacement de votre fichier manifeste de test.

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

13. Sélectionnez Envoyer.
14. Suivez les étapes décrites dans Entraînement de votre modèle pour entraîner le modèle.

Amazon Lookout for Vision crée un ensemble de données dans le dossier du compartiment Amazon `datasets` S3. Votre `.manifest` fichier d'origine reste inchangé.

Création d'un ensemble de données à l'aide d'un fichier manifeste (SDK)

Vous utilisez cette CreateDatasetopération pour créer les ensembles de données associés à un projet Amazon Lookout for Vision.

Si vous souhaitez utiliser un seul ensemble de données pour l'entraînement et les tests, créez un ensemble de données unique dont la `DatasetType` valeur est définie sur`train`. Pendant l'entraînement, l'ensemble de données est divisé en interne pour former un ensemble de données d'entraînement et de test. Vous n'avez pas accès aux ensembles de données d'entraînement et de test divisés. Si vous souhaitez un ensemble de données de test distinct, effectuez un deuxième appel à `CreateDataset` avec le jeu de `DatasetType` valeurs`test`. Pendant l'entraînement, les ensembles de données d'entraînement et de test sont utilisés pour entraîner et tester le modèle.

Vous pouvez éventuellement utiliser le `DatasetSource` paramètre pour spécifier l'emplacement d'un fichier manifeste au format SageMaker AI Ground Truth utilisé pour remplir l'ensemble de données. Dans ce cas, l'appel à `CreateDataset` est asynchrone. Pour vérifier le statut actuel, appelez `DescribeDataset`. Pour de plus amples informations, veuillez consulter Afficher vos ensembles de données. Si une erreur de validation se produit lors de l'importation, la valeur de `Status` est définie sur CREATE_FAILED et le message d'état (`StatusMessage`) est défini.

> **Tip**
>
> Si vous créez un ensemble de données à partir de l'exemple de jeu de données Getting Started, utilisez le fichier manifeste (`getting-started/dataset-files/manifests/`

`train.manifest`) dans lequel le script créeÉtape 1 : créer le fichier manifeste et télécharger des images.
Si vous créez un jeu de données à partir des exemples d'images du circuit imprimé, deux options s'offrent à vous :

1. Créez le fichier manifeste à l'aide du code. Le bloc-notes Python Amazon Lookout for Vision Lab explique comment créer le fichier manifeste pour les exemples d'images du circuit imprimé. Vous pouvez également utiliser l'exemple de code Datasets dans le référentiel d'exemples de AWS code.
2. Si vous avez déjà utilisé la console Amazon Lookout for Vision pour créer un ensemble de données avec les exemples d'images du circuit imprimé, réutilisez les fichiers manifestes créés pour vous par Amazon Lookout for Vision. Les emplacements des fichiers manifestes de formation et de test sonts3://*bucket*/datasets/*project name*/*train or test*/manifests/output/output.manifest.

Si vous ne le spécifiez pasDatasetSource, un ensemble de données vide est créé. Dans ce cas, l'appel à `CreateDataset` est synchrone. Plus tard, vous pouvez étiqueter des images dans le jeu de données en appelant UpdateDatasetEntries. Pour obtenir un exemple de code, consultez Ajout d'autres images (kit SDK).

Si vous souhaitez remplacer un ensemble de données, supprimez d'abord le jeu de données existant par, DeleteDatasetpuis créez un nouveau jeu de données du même type en appelant`CreateDataset`. Pour de plus amples informations, veuillez consulter Suppression d'un jeu de données.

Après avoir créé les ensembles de données, vous pouvez créer le modèle. Pour de plus amples informations, veuillez consulter Entraînement d'un modèle (kit SDK).

Vous pouvez afficher les images étiquetées (lignes JSON) dans un ensemble de données en appelant ListDatasetEntries. Vous pouvez ajouter des images étiquetées en appelant`UpdateDatasetEntries`.

Pour consulter les informations relatives aux ensembles de données de test et d'entraînement, consultezAfficher vos ensembles de données.

Pour créer un ensemble de données (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 créer un ensemble de données.

   CLI

   Remplacez les valeurs suivantes :

   - `project-name`au nom du projet auquel vous souhaitez associer le jeu de données.
   - `dataset-type`au type de jeu de données que vous souhaitez créer (`train`ou`test`).
   - `dataset-source`à l'emplacement Amazon S3 du fichier manifeste.
   - `Bucket`au nom du compartiment Amazon S3 qui contient le fichier manifeste.
   - `Key`au chemin et au nom du fichier manifeste dans le compartiment Amazon S3.

```
aws lookoutvision create-dataset --project-name project\
  --dataset-type train or test\
  --dataset-source '{ "GroundTruthManifest": { "S3Object": { "Bucket": "bucket",
 "Key": "manifest file" } } }' \
  --profile lookoutvision-access
```

   Python

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

```
    @staticmethod
    def create_dataset(lookoutvision_client, project_name, manifest_file,
 dataset_type):
        """
        Creates a new Lookout for Vision dataset

        :param lookoutvision_client: A Lookout for Vision Boto3 client.
        :param project_name: The name of the project in which you want to
                             create a dataset.
        :param bucket: The bucket that contains the manifest file.
        :param manifest_file: The path and name of the manifest file.
        :param dataset_type: The type of the dataset (train or test).
```

```
        """
        try:
            bucket, key = manifest_file.replace("s3://", "").split("/", 1)
            logger.info("Creating %s dataset type...", dataset_type)
            dataset = {
                "GroundTruthManifest": {"S3Object": {"Bucket": bucket, "Key":
 key}}
            }
            response = lookoutvision_client.create_dataset(
                ProjectName=project_name,
                DatasetType=dataset_type,
                DatasetSource=dataset,
            )
            logger.info("Dataset Status: %s", response["DatasetMetadata"]
["Status"])
            logger.info(
                "Dataset Status Message: %s",
                response["DatasetMetadata"]["StatusMessage"],
            )
            logger.info("Dataset Type: %s", response["DatasetMetadata"]
["DatasetType"])

            # Wait until either created or failed.
            finished = False
            status = ""
            dataset_description = {}
            while finished is False:
                dataset_description = lookoutvision_client.describe_dataset(
                    ProjectName=project_name, DatasetType=dataset_type
                )
                status = dataset_description["DatasetDescription"]["Status"]

                if status == "CREATE_IN_PROGRESS":
                    logger.info("Dataset creation in progress...")
                    time.sleep(2)
                elif status == "CREATE_COMPLETE":
                    logger.info("Dataset created.")
                    finished = True
                else:
                    logger.info(
                        "Dataset creation failed: %s",
                        dataset_description["DatasetDescription"]
["StatusMessage"],
                    )
```

```
                    finished = True

            if status != "CREATE_COMPLETE":
                message = dataset_description["DatasetDescription"]
["StatusMessage"]
                logger.exception("Couldn't create dataset: %s", message)
                raise Exception(f"Couldn't create dataset: {message}")

        except ClientError:
            logger.exception("Service error: Couldn't create dataset.")
            raise
```

Java V2

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

```
/**
 * Creates an Amazon Lookout for Vision dataset from a manifest file.
 * Returns after Lookout for Vision creates the dataset.
 *
 * @param lfvClient    An Amazon Lookout for Vision client.
 * @param projectName  The name of the project in which you want to create a
 *                     dataset.
 * @param datasetType  The type of dataset that you want to create (train or
 *                     test).
 * @param bucket       The S3 bucket that contains the manifest file.
 * @param manifestFile The name and location of the manifest file within the S3
 *                     bucket.
 * @return DatasetDescription The description of the created dataset.
 */
public static DatasetDescription createDataset(LookoutVisionClient lfvClient,
                String projectName,
                String datasetType,
                String bucket,
                String manifestFile)
                throws LookoutVisionException, InterruptedException {

        logger.log(Level.INFO, "Creating {0} dataset for project {1}",
                        new Object[] { projectName, datasetType });
```

```
        // Build the request. If no bucket supplied, setup for empty dataset
creation.
        CreateDatasetRequest createDatasetRequest = null;

        if (bucket != null && manifestFile != null) {

                InputS3Object s3Object = InputS3Object.builder()
                                .bucket(bucket)
                                .key(manifestFile)
                                .build();

                DatasetGroundTruthManifest groundTruthManifest =
DatasetGroundTruthManifest.builder()
                                .s3Object(s3Object)
                                .build();

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

                createDatasetRequest = CreateDatasetRequest.builder()
                                .projectName(projectName)
                                .datasetType(datasetType)
                                .datasetSource(datasetSource)
                                .build();
        } else {
                createDatasetRequest = CreateDatasetRequest.builder()
                                .projectName(projectName)
                                .datasetType(datasetType)
                                .build();
        }

        lfvClient.createDataset(createDatasetRequest);

        DatasetDescription datasetDescription = null;

        boolean finished = false;

        // Wait until dataset is created, or failure occurs.
        while (!finished) {

                datasetDescription = describeDataset(lfvClient, projectName,
datasetType);
```

```
                switch (datasetDescription.status()) {
                        case CREATE_COMPLETE:
                                logger.log(Level.INFO, "{0}dataset created for
project {1}",
                                                new Object[] { datasetType,
projectName });
                                finished = true;
                                break;
                        case CREATE_IN_PROGRESS:
                                logger.log(Level.INFO, "{0} dataset creating for
project {1}",
                                                new Object[] { datasetType,
projectName });

                                TimeUnit.SECONDS.sleep(5);

                                break;

                        case CREATE_FAILED:
                                logger.log(Level.SEVERE,
                                                "{0} dataset creation failed for
project {1}. Error {2}",
                                                new Object[] { datasetType,
projectName,

datasetDescription.statusAsString() });
                                finished = true;
                                break;
                        default:
                                logger.log(Level.SEVERE, "{0} error when
creating {1} dataset for project {2}",
                                                new Object[] { datasetType,
projectName,

datasetDescription.statusAsString() });
                                finished = true;
                                break;

                }
        }

        logger.log(Level.INFO, "Dataset info. Status: {0}\n Message: {1} }",
                        new Object[] { datasetDescription.statusAsString(),
                                        datasetDescription.statusMessage() });
```

```
        return datasetDescription;

}
```

3. Entraînez votre modèle en suivant les étapes décrites dansEntraînement d'un modèle (kit SDK).

Étiquetage des images

Vous pouvez utiliser la console Amazon Lookout for Vision pour ajouter ou modifier les étiquettes attribuées aux images de votre ensemble de données. Si vous utilisez le SDK, les étiquettes font partie du fichier manifeste que vous fournissez. CreateDataset Vous pouvez mettre à jour les libellés d'une image en appelant UpdateDatasetEntries. Pour obtenir un exemple de code, consultez Ajout d'autres images (kit SDK).

Choix du type de modèle

Les étiquettes que vous attribuez aux images déterminent le type de modèle créé par Lookout for Vision. Si votre projet possède un ensemble de données de test distinct, étiquetez les images de la même manière.

Modèle de classification d'images

Pour créer un modèle de classification d'images, vous devez classer chaque image comme normale ou anormale. Pour chaque image, faites-leClassification des images (console).

Modèle de segmentation d'image

Pour créer un modèle de segmentation d'image, vous devez classer chaque image comme normale ou anormale. Pour chaque image anormale, vous spécifiez également un masque de pixels pour chaque zone anormale de l'image et une étiquette d'anomalie pour le type d'anomalie dans le masque de pixels. Par exemple, le masque bleu de l'image suivante indique l'emplacement d'une anomalie de rayure sur une voiture. Vous pouvez spécifier plusieurs types d'étiquette d'anomalie dans une image. Pour chaque image, faites-leSegmentation d'images (console).

Classification des images (console)

La console Lookout for Vision vous permet de classer les images d'un jeu de données comme normales ou comme anormales. Les images non classées ne sont pas utilisées pour entraîner votre modèle.

Si vous créez un modèle de segmentation d'image, ignorez cette procédureSegmentation d'images (console), qui inclut les étapes de classification des images.

> **Note**
>
> Si vous venez de terminerCréation de votre jeu de données, la console devrait actuellement afficher le tableau de bord de votre modèle et vous n'avez pas besoin de suivre les étapes 1 à 4.

Pour classer vos images (console)

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Dans le volet de navigation de gauche, choisissez Projets.
3. Sur la page Projets, choisissez le projet que vous voulez utiliser.
4. Dans le volet de navigation de gauche de votre projet, sélectionnez Dataset.
5. Si vous avez des ensembles de données d'entraînement et de test distincts, choisissez l'onglet correspondant au jeu de données que vous souhaitez utiliser.
6. Choisissez Commencer à étiqueter.
7. Choisissez Sélectionner toutes les images de cette page.

8. Si les images sont normales, choisissez Classer comme normal, sinon sélectionnez Classer comme anomalie. Une étiquette apparaît sous chaque photo.
9. Si vous devez modifier le libellé d'une image, procédez comme suit :

    a. Choisissez Anomalie ou Normal sous l'image.

    b. Si vous ne parvenez pas à déterminer l'étiquette correcte pour une image, agrandissez l'image en la sélectionnant dans la galerie.

> **Note**
>
> Vous pouvez filtrer les étiquettes des images en choisissant l'étiquette ou l'état d'étiquette souhaité dans la section Filtres.

10. Répétez les étapes 7 à 9 sur chaque page si nécessaire jusqu'à ce que toutes les images du jeu de données soient correctement étiquetées.
11. Sélectionnez Enregistrer les modifications.
12. Une fois que vous avez fini d'étiqueter vos images, vous pouvez entraîner votre modèle.

Segmentation d'images (console)

Si vous créez un modèle de segmentation d'image, vous devez classer les images comme normales ou comme anormales. Vous devez également ajouter des informations de segmentation aux images anormales. Pour spécifier les informations de segmentation, vous devez d'abord définir des étiquettes d'anomalie pour chaque type d'anomalie, telle qu'une bosse ou une égratignure, que vous souhaitez que votre modèle détecte. Vous devez ensuite spécifier un masque d'anomalie et une étiquette d'anomalie pour chaque anomalie sur les images anormales de votre jeu de données.

> **Note**
>
> Si vous créez un modèle de classification d'images, vous n'avez pas besoin de segmenter les images ni de spécifier des étiquettes d'anomalie.

Rubriques

- Spécification des étiquettes d'anomalies
- Étiqueter une image

- Segmenter une image à l'aide de l'outil d'annotation

Spécification des étiquettes d'anomalies

Vous définissez une étiquette d'anomalie pour chaque type d'anomalie figurant dans les images du jeu de données.

Spécifier les étiquettes d'anomalie

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Dans le volet de navigation de gauche, choisissez Projets.
3. Sur la page Projets, choisissez le projet que vous voulez utiliser.
4. Dans le volet de navigation de gauche de votre projet, sélectionnez Dataset.
5. Dans Étiquettes d'anomalies, choisissez Ajouter des étiquettes d'anomalies. Si vous avez déjà ajouté une étiquette d'anomalie, choisissez Gérer.
6. Dans la boîte de dialogue , procédez comme suit :
    a. Entrez l'étiquette d'anomalie que vous souhaitez ajouter et choisissez Ajouter une étiquette d'anomalie.
    b. Répétez l'étape précédente jusqu'à ce que vous ayez saisi toutes les étiquettes d'anomalie que vous souhaitez que votre modèle détecte.
    c. (Facultatif) Cliquez sur l'icône d'édition pour modifier le nom de l'étiquette.
    d. (Facultatif) Cliquez sur l'icône de suppression pour supprimer une nouvelle étiquette d'anomalie. Vous ne pouvez pas supprimer les types d'anomalies que votre ensemble de données utilise actuellement.
7. Choisissez Confirmer pour ajouter les nouvelles étiquettes d'anomalies à l'ensemble de données.

Après avoir spécifié les étiquettes des anomalies, étiquetez les images en Étiqueter une image procédant.

Étiqueter une image

Pour étiqueter une image en vue de sa segmentation, classez-la comme normale ou comme anormale. Utilisez ensuite l'outil d'annotation pour segmenter l'image en dessinant des masques qui recouvrent étroitement les zones correspondant à chaque type d'anomalie présente dans l'image.

Pour étiqueter une image

1. Si vous avez des ensembles de données d'entraînement et de test distincts, choisissez l'onglet correspondant au jeu de données que vous souhaitez utiliser.
2. Si ce n'est pas déjà fait, spécifiez les types d'anomalies pour votre ensemble de données en Spécification des étiquettes d'anomalies procédant.
3. Choisissez Commencer à étiqueter.
4. Choisissez Sélectionner toutes les images de cette page.
5. Si les images sont normales, choisissez Classer comme normal, sinon sélectionnez Classer comme anomalie.
6. Pour modifier le libellé d'une seule image, choisissez Normal ou Anomalie sous l'image.

> **Note**
>
> Vous pouvez filtrer les étiquettes des images en choisissant l'étiquette ou l'état d'étiquette souhaité dans la section Filtres. Vous pouvez trier par indice de confiance dans la section Options de tri.

7. Pour chaque image anormale, choisissez l'image pour ouvrir l'outil d'annotation. Ajoutez des informations de segmentation en procédantSegmenter une image à l'aide de l'outil d'annotation.
8. Sélectionnez Enregistrer les modifications.
9. Une fois que vous avez fini d'étiqueter vos images, vous pouvez entraîner votre modèle.

Segmenter une image à l'aide de l'outil d'annotation

L'outil d'annotation permet de segmenter une image en marquant les zones anormales à l'aide d'un masque.

Pour segmenter une image à l'aide de l'outil d'annotation

1. Ouvrez l'outil d'annotation en sélectionnant l'image dans la galerie de jeux de données. Si nécessaire, choisissez Commencer l'étiquetage pour passer en mode d'étiquetage.
2. Dans la section Étiquettes d'anomalies, choisissez l'étiquette d'anomalie que vous souhaitez marquer. Si nécessaire, choisissez Ajouter des étiquettes d'anomalie pour ajouter une nouvelle étiquette d'anomalie.

3. Choisissez un outil de dessin au bas de la page et dessinez des masques qui recouvrent étroitement les zones anormales pour l'étiquette d'anomalie. L'image suivante est un exemple de masque qui couvre étroitement une anomalie.

Voici un exemple de masque médiocre qui ne couvre pas parfaitement une anomalie.

4. Si vous avez d'autres images à segmenter, choisissez Next et répétez les étapes 2 et 3.

5. Choisissez Soumettre et fermez pour terminer la segmentation des images.

Entraînement de votre modèle

Après avoir créé vos ensembles de données et étiqueté les images, vous pouvez entraîner votre modèle. Dans le cadre du processus de formation, un ensemble de données de test est utilisé. Si vous avez un seul projet de jeu de données, les images du jeu de données sont automatiquement divisées en un ensemble de données de test et un ensemble de données d'apprentissage dans le cadre du processus de formation. Si votre projet comporte un ensemble de données d'entraînement et un ensemble de données de test, ils sont utilisés pour entraîner et tester séparément l'ensemble de données.

Une fois l'entraînement terminé, vous pouvez évaluer les performances du modèle et apporter les améliorations nécessaires. Pour de plus amples informations, veuillez consulter Améliorer votre modèle Amazon Lookout for Vision.

Pour entraîner votre modèle, Amazon Lookout for Vision crée une copie de vos images d'entraînement et de test source. Par défaut, les images copiées sont chiffrées à l'aide d'une clé détenue et gérée par AWS. Vous pouvez également choisir d'utiliser votre propre clé AWS Key Management Service (KMS). Pour plus d'informations, consultez Concepts d'AWS Key Management Service. Vos images source ne sont pas affectées.

Vous pouvez attribuer des métadonnées à votre modèle sous forme de balises. Pour de plus amples informations, veuillez consulter Modèles de balisage.

Chaque fois que vous entraînez un modèle, une nouvelle version du modèle est créée. Si vous n'avez plus besoin de la version d'un modèle, vous pouvez le supprimer. Pour de plus amples informations, veuillez consulter Suppression d'un modèle.

Le temps nécessaire pour entraîner avec succès votre modèle vous est facturé. Pour plus d'informations, consultez la section Heures de formation.

Pour afficher les modèles existants dans un projet,Visualisation de vos modèles.

> **Note**
>
> Si vous venez de terminer Création de votre jeu de données ouAjouter des images à votre jeu de données. La console devrait actuellement afficher le tableau de bord de votre modèle et vous n'avez pas besoin de suivre les étapes 1 à 4.

Rubriques

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

Entraînement d'un modèle (console)

La procédure suivante explique comment entraîner votre modèle à l'aide de la console.

Pour entraîner votre modèle (console)

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Dans le volet de navigation de gauche, choisissez Projets.
3. Sur la page Projets, choisissez le projet qui contient le modèle que vous souhaitez entraîner.
4. Sur la page des détails du projet, choisissez Train model. Le bouton Train model est disponible si vous disposez de suffisamment d'images étiquetées pour entraîner le modèle. Si le bouton n'est pas disponible, ajoutez d'autres images jusqu'à ce que vous ayez suffisamment d'images étiquetées.
5. (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.
6. (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.

7. Choisissez Train model (Entraîner un modèle).
8. Dans la boîte de dialogue Voulez-vous entraîner votre modèle ?, choisissez Entraîner un modèle.
9. Dans la vue Modèles, vous pouvez voir que l'entraînement a commencé et vous pouvez vérifier l'état actuel en consultant la `Status` colonne correspondant à la version du modèle. L'entraînement d'un modèle peut nécessiter un certain temps.
10. Lorsque l'entraînement est terminé, vous pouvez évaluer ses performances. Pour de plus amples informations, veuillez consulter Améliorer votre modèle Amazon Lookout for Vision.

Entraînement d'un modèle (kit SDK)

Vous utilisez cette CreateModelopération pour démarrer l'entraînement, le test et l'évaluation d'un modèle. Amazon Lookout for Vision entraîne le modèle à l'aide de l'ensemble de données de formation et de test associé au projet. Pour de plus amples informations, veuillez consulter Création d'un projet (kit SDK).

Chaque fois que vous appelez`CreateModel`, une nouvelle version du modèle est créée. Le formulaire de réponse `CreateModel` inclut la version du modèle.

Vous êtes facturé pour chaque modèle de formation réussie. Utilisez le paramètre `ClientToken` d'entrée pour éviter les frais dus à des répétitions inutiles ou accidentelles de l'entraînement du modèle par vos utilisateurs. `ClientToken`est un paramètre d'entrée idempotent qui garantit que l'entraînement n'est pas terminé une `CreateModel` seule fois pour un ensemble spécifique de paramètres. Un appel répété `CreateModel` avec la même `ClientToken` valeur garantit que l'entraînement n'est pas répété. Si vous ne fournissez aucune valeur pour`ClientToken`, le SDK AWS que vous utilisez insère une valeur pour vous. Cela empêche les nouvelles tentatives après une erreur réseau de démarrer plusieurs tâches de formation, mais vous devrez fournir votre propre valeur pour vos propres cas d'utilisation. Pour de plus amples informations, veuillez consulter CreateModel.

L'entraînement prend un certain temps. Pour vérifier l'état actuel, appelez `DescribeModel` et transmettez le nom du projet (spécifié dans l'appel à`CreateProject`) et la version du modèle. Le `status` champ indique l'état actuel de l'entraînement du modèle. Pour obtenir un exemple de code, consultez Affichage de vos modèles (SDK).

Si la formation est réussie, vous pouvez évaluer le modèle. Pour de plus amples informations, veuillez consulter Améliorer votre modèle Amazon Lookout for Vision.

Pour afficher les modèles que vous avez créés dans un projet, appelez`ListModels`. Pour obtenir un exemple de code, consultez Visualisation de vos modèles.

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 modèle.

   CLI

   Remplacez les valeurs suivantes :

   - `project-name`au nom du projet qui contient le modèle que vous souhaitez créer.
   - `output-config`à l'endroit où vous souhaitez enregistrer les résultats de l'entraînement. Remplacez les valeurs suivantes :
     - `output bucket`avec le nom du compartiment Amazon S3 dans lequel Amazon Lookout for Vision enregistre les résultats de formation.
     - `output folder`avec le nom du dossier dans lequel vous souhaitez enregistrer les résultats de l'entraînement.
     - `Key`avec le nom d'une clé de tag.
     - `Value`avec une valeur à associer`tag_key`.

   ```
   aws lookoutvision create-model --project-name "project name"\
     --output-config '{ "S3Location": { "Bucket": "output bucket", "Prefix":
    "output folder" } }'\
     --tags '[{"Key":"Key","Value":"Value"}]' \
     --profile lookoutvision-access
   ```

   Python

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

   ```
       @staticmethod
   ```

```
    def create_model(
        lookoutvision_client,
        project_name,
        training_results,
        tag_key=None,
        tag_key_value=None,
    ):
        """
        Creates a version of a Lookout for Vision model.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project in which you want to create
 a
                             model.
        :param training_results: The Amazon S3 location where training results
 are stored.
        :param tag_key: The key for a tag to add to the model.
        :param tag_key_value - A value associated with the tag_key.
        return: The model status and version.
        """
        try:
            logger.info("Training model...")
            output_bucket, output_folder = training_results.replace("s3://",
 "").split(
                "/", 1
            )
            output_config = {
                "S3Location": {"Bucket": output_bucket, "Prefix": output_folder}
            }
            tags = []
            if tag_key is not None:
                tags = [{"Key": tag_key, "Value": tag_key_value}]

            response = lookoutvision_client.create_model(
                ProjectName=project_name, OutputConfig=output_config, Tags=tags
            )

            logger.info("ARN: %s", response["ModelMetadata"]["ModelArn"])
            logger.info("Version: %s", response["ModelMetadata"]
["ModelVersion"])
            logger.info("Started training...")

            print("Training started. Training might take several hours to
 complete.")
```

```
            # Wait until training completes.
            finished = False
            status = "UNKNOWN"
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name,
                    ModelVersion=response["ModelMetadata"]["ModelVersion"],
                )
                status = model_description["ModelDescription"]["Status"]

                if status == "TRAINING":
                    logger.info("Model training in progress...")
                    time.sleep(600)
                    continue

                if status == "TRAINED":
                    logger.info("Model was successfully trained.")
                else:
                    logger.info(
                        "Model training failed: %s ",
                        model_description["ModelDescription"]["StatusMessage"],
                    )
                finished = True
        except ClientError:
            logger.exception("Couldn't train model.")
            raise
        else:
            return status, response["ModelMetadata"]["ModelVersion"]
```

Java V2

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

```
/**
 * Creates an Amazon Lookout for Vision model. The function returns after model
 * training completes. Model training can take multiple hours to complete.
 * You are charged for the amount of time it takes to successfully train a
 model.
 * Returns after Lookout for Vision creates the dataset.
 *
```

```
     * @param lfvClient   An Amazon Lookout for Vision client.
     * @param projectName The name of the project in which you want to create a
     *                    model.
     * @param description A description for the model.
     * @param bucket      The S3 bucket in which Lookout for Vision stores the
     *                    training results.
     * @param folder      The location of the training results within the S3
     *                    bucket.
     * @return ModelDescription The description of the created model.
     */
    public static ModelDescription createModel(LookoutVisionClient lfvClient, String
     projectName,
                    String description, String bucket, String folder)
                    throws LookoutVisionException, InterruptedException {

            logger.log(Level.INFO, "Creating model for project: {0}.", new Object[]
     { projectName });

            // Setup input parameters.
            S3Location s3Location = S3Location.builder()
                            .bucket(bucket)
                            .prefix(folder)
                            .build();

            OutputConfig config = OutputConfig.builder()
                            .s3Location(s3Location)
                            .build();

            CreateModelRequest createModelRequest = CreateModelRequest.builder()
                            .projectName(projectName)
                            .description(description)
                            .outputConfig(config)
                            .build();

            // Create and train the model.
            CreateModelResponse response =
     lfvClient.createModel(createModelRequest);

            String modelVersion = response.modelMetadata().modelVersion();
            boolean finished = false;
            DescribeModelResponse descriptionResponse = null;

            // Wait until training finishes or fails.
```

```
        do {
                DescribeModelRequest describeModelRequest =
DescribeModelRequest.builder()
                                .projectName(projectName)
                                .modelVersion(modelVersion)
                                .build();

                descriptionResponse =
lfvClient.describeModel(describeModelRequest);

                switch (descriptionResponse.modelDescription().status()) {
                        case TRAINED:
                                logger.log(Level.INFO, "Model training completed
for project {0} version {1}.",
                                                new Object[] { projectName,
modelVersion });
                                finished = true;
                                break;

                        case TRAINING:
                                logger.log(Level.INFO,
                                                "Model training in progress for
project {0} version {1}.",
                                                new Object[] { projectName,
modelVersion });
                                TimeUnit.SECONDS.sleep(60);

                                break;

                        case TRAINING_FAILED:
                                logger.log(Level.SEVERE,
                                                "Model training failed for for
project {0} version {1}.",
                                                new Object[] { projectName,
modelVersion });
                                finished = true;
                                break;

                        default:
                                logger.log(Level.SEVERE,
                                                "Unexpected error when training
model project {0} version {1}: {2}.",
                                                new Object[] { projectName,
modelVersion,
```

```
 descriptionResponse.modelDescription()

 .status() });
                                finished = true;
                                break;

                }
        } while (!finished);

        return descriptionResponse.modelDescription();

}
```

3. Lorsque l'entraînement est terminé, vous pouvez évaluer ses performances. Pour de plus amples informations, veuillez consulter Améliorer votre modèle Amazon Lookout for Vision.

Modèle de résolution des problèmes

Des problèmes liés à votre fichier manifeste ou à vos images d'entraînement peuvent entraîner l'échec de l'entraînement du modèle. Avant de recycler votre modèle, vérifiez les problèmes potentiels suivants.

Les couleurs des étiquettes d'anomalies ne correspondent pas à la couleur des anomalies dans l'image du masque

Si vous entraînez un modèle de segmentation d'image, la couleur de l'étiquette d'anomalie dans le fichier manifeste doit correspondre à celle de l'image du masque. La ligne JSON d'une image dans le fichier manifeste contient des métadonnées (`internal-color-map`) qui indiquent à Amazon Lookout for Vision quelle couleur correspond à une étiquette d'anomalie. Par exemple, la couleur de l'étiquette d'`scratch`anomalie dans la ligne JSON suivante est`#2ca02c`.

```
{
    "source-ref": "s3://path-to-image",
    "anomaly-label": 1,
    "anomaly-label-metadata": {
        "class-name": "anomaly",
        "creation-date": "2021-10-12T14:16:45.668",
        "human-annotated": "yes",
```

```
        "job-name": "labeling-job/classification-job",
        "type": "groundtruth/image-classification",
        "confidence": 1
    },
    "anomaly-mask-ref": "s3://path-to-image",
    "anomaly-mask-ref-metadata": {
        "internal-color-map": {
            "0": {
                "class-name": "BACKGROUND",
                "hex-color": "#ffffff",
                "confidence": 0.0
            },
            "1": {
                "class-name": "scratch",
                "hex-color": "#2ca02c",
                "confidence": 0.0
            },
            "2": {
                "class-name": "dent",
                "hex-color": "#1f77b4",
                "confidence": 0.0
            }
        },
        "type": "groundtruth/semantic-segmentation",
        "human-annotated": "yes",
        "creation-date": "2021-11-23T20:31:57.758889",
        "job-name": "labeling-job/segmentation-job"
    }
}
```

Si les couleurs de l'image du masque ne correspondent pas aux valeurs indiquéeshex-color, l'apprentissage échoue et vous devez mettre à jour le fichier manifeste.

Pour mettre à jour les valeurs de couleur dans un fichier manifeste

1. À l'aide d'un éditeur de texte, ouvrez le fichier manifeste que vous avez utilisé pour créer le jeu de données.
2. Pour chaque ligne JSON (image), vérifiez que les couleurs (`hex-color`) du `internal-color-map` champ correspondent aux couleurs des étiquettes d'anomalie de l'image du masque.

Vous pouvez obtenir l'emplacement de l'image du masque à partir du *anomaly-mask*-ref champ. Téléchargez l'image sur votre ordinateur et utilisez le code suivant pour obtenir les couleurs d'une image.

```
from PIL import Image
img = Image.open('path to local copy of mask file')
colors = img.convert('RGB').getcolors() #this converts the mode to RGB
for color in colors:
    print('#%02x%02x%02x' % color[1])
```

3. Pour chaque image dont l'attribution de couleur est incorrecte, mettez à jour le hex-color champ dans la ligne JSON de l'image.
4. Enregistrez le fichier manifeste de mise à jour.
5. Supprimez le jeu de données existant du projet.
6. Créez un nouvel ensemble de données dans le projet avec le fichier manifeste mis à jour.
7. Entraînez le modèle.

Pour les étapes 5 et 6, vous pouvez également mettre à jour des images individuelles dans le jeu de données en appelant l'UpdateDatasetEntriesopération et en fournissant des lignes JSON mises à jour pour les images que vous souhaitez mettre à jour. Pour obtenir un exemple de code, consultez Ajout d'autres images (kit SDK).

Les images du masque ne sont pas au format PNG

Si vous entraînez un modèle de segmentation d'image, les images du masque doivent être au format PNG. Si vous créez un jeu de données à partir d'un fichier manifeste, assurez-vous que les images de masque auxquelles vous faites référence *anomaly-mask*-ref sont au format PNG. Si les images du masque ne sont pas au format PNG, vous devez les convertir au format PNG. Il ne suffit pas de renommer l'extension d'un fichier image en.png.

Les images de masque que vous créez dans la console Amazon Lookout for Vision ou à l'aide SageMaker d'une tâche AI Ground Truth sont créées au format PNG. Il n'est pas nécessaire de modifier le format de ces images.

Pour corriger les images de masque au format autre que PNG dans un fichier manifeste

1. À l'aide d'un éditeur de texte, ouvrez le fichier manifeste que vous avez utilisé pour créer le jeu de données.
2. Pour chaque ligne JSON (image), assurez-vous que l'image *anomaly-mask*`-ref` fait référence à une image au format PNG. Pour de plus amples informations, veuillez consulter Création d'un fichier manifeste.
3. Enregistrez le fichier manifeste mis à jour.
4. Supprimez le jeu de données existant du projet.
5. Créez un nouvel ensemble de données dans le projet avec le fichier manifeste mis à jour.
6. Entraînez le modèle.

Les étiquettes de segmentation ou de classification sont inexactes ou manquantes

Des étiquettes manquantes ou inexactes peuvent entraîner l'échec de la formation ou créer un modèle peu performant. Nous vous recommandons d'étiqueter toutes les images de votre jeu de données. Si vous n'étiquetez pas toutes les images et que l'apprentissage du modèle échoue ou que votre modèle fonctionne mal, ajoutez d'autres images.

Vérifiez les éléments suivants :

- Si vous créez un modèle de segmentation, les masques doivent couvrir étroitement les anomalies présentes sur les images de votre jeu de données. Pour vérifier les masques de votre jeu de données, visualisez les images dans la galerie de jeux de données du projet. Si nécessaire, redessinez les masques d'image. Pour de plus amples informations, veuillez consulter Segmentation d'images (console).

- Assurez-vous que les images anormales présentes dans les images de votre jeu de données sont classées. Si vous créez un modèle de segmentation d'image, assurez-vous que les images anormales comportent des étiquettes d'anomalie et des masques d'image.

  Il est important de se souvenir du type de modèle (segmentation ou classification) que vous créez. Un modèle de classification ne nécessite pas de masques d'image sur les images

anormales. N'ajoutez pas de masques aux images de jeux de données destinées à un modèle de classification.

Pour mettre à jour les étiquettes manquantes

1. Ouvrez la galerie de jeux de données du projet.
2. Filtrez les images non étiquetées pour voir quelles images n'ont pas d'étiquette.
3. Effectuez l'une des actions suivantes :
   - Si vous créez un modèle de classification d'images, classez chaque image non étiquetée.
   - Si vous créez un modèle de segmentation d'image, classez et segmentez chaque image non étiquetée.
4. Si vous créez un modèle de segmentation d'image, ajoutez des masques à toutes les images anormales classées pour lesquelles il manque des masques.
5. Entraînez le modèle.

Si vous choisissez de ne pas corriger les étiquettes incorrectes ou manquantes, nous vous recommandons d'ajouter d'autres images étiquetées ou de supprimer les images concernées du jeu de données. Vous pouvez en ajouter d'autres depuis la console ou en utilisant l'UpdateDatasetEntriesopération. Pour de plus amples informations, veuillez consulter Ajouter des images à votre jeu de données.

Si vous choisissez de supprimer les images, vous devez recréer le jeu de données sans les images concernées, car vous ne pouvez pas supprimer d'image d'un ensemble de données. Pour de plus amples informations, veuillez consulter Supprimer des images de votre jeu de données.

Améliorer votre modèle Amazon Lookout for Vision

Pendant l'entraînement, Lookout for Vision teste votre modèle à l'aide du jeu de données de test et utilise les résultats pour créer des indicateurs de performance. Vous pouvez utiliser des indicateurs de performance pour évaluer les performances de votre modèle. Si nécessaire, vous pouvez prendre des mesures pour améliorer vos ensembles de données, puis réentraîner votre modèle.

Si vous êtes satisfait des performances de votre modèle, vous pouvez commencer à l'utiliser. Pour de plus amples informations, veuillez consulter Exécution de votre modèle Amazon Lookout for Vision entraîné.

Rubriques

- Étape 1 : Évaluez les performances de votre modèle
- Étape 2 : Améliorez votre modèle
- Consultation des métriques de performances
- Vérification de votre modèle à l'aide d'une tâche de détection d'essai

Étape 1 : Évaluez les performances de votre modèle

Vous pouvez accéder aux indicateurs de performance depuis la console et depuis l'DescribeModelopération. Amazon Lookout for Vision fournit des mesures de performance récapitulatives pour le jeu de données de test et les résultats prévus pour toutes les images individuelles. Si votre modèle est un modèle de segmentation, la console affiche également des mesures récapitulatives pour chaque étiquette d'anomalie.

Pour consulter les indicateurs de performance et tester les prédictions d'image dans la console, voirAffichage des indicateurs de performance (console). Pour plus d'informations sur l'accès aux mesures de performance et le test des prédictions d'image avec l'`DescribeModel`opération, consultezAffichage des indicateurs de performance (SDK).

Métriques de classification des images

Amazon Lookout for Vision fournit les statistiques récapitulatives suivantes pour les classifications effectuées par un modèle lors des tests :

- Précision

- Rappel
- Score de F1

Métriques du modèle de segmentation d'images

Si le modèle est un modèle de segmentation d'image, Amazon Lookout for Vision fournit des mesures de classification des images récapitulatives et des mesures de performance récapitulatives pour chaque étiquette d'anomalie :

- Score de F1
- Intersection moyenne au-dessus de l'Union (IoU)

Précision

La métrique de précision répond à la question suivante : lorsque le modèle prédit qu'une image contient une anomalie, à quelle fréquence cette prédiction est-elle correcte ?

La précision est une mesure utile dans les situations où le coût d'un faux positif est élevé. Par exemple, le coût du retrait d'une pièce de machine non défectueuse d'une machine assemblée.

Amazon Lookout for Vision fournit une valeur métrique de précision récapitulative pour l'ensemble de données de test.

La précision est la fraction des anomalies correctement prédites (vrais positifs) par rapport à toutes les anomalies prédites (vrais et faux positifs). La formule de précision est la suivante.

Valeur de précision = vrais positifs/(vrais positifs + faux positifs)

Les valeurs de précision possibles sont comprises entre 0 et 1. La console Amazon Lookout for Vision affiche la précision sous forme de pourcentage (0 à 100).

Une valeur de précision plus élevée indique qu'un plus grand nombre d'anomalies prédites sont correctes. Supposons, par exemple, que votre modèle prédit que 100 images sont anormales. Si 85 des prédictions sont correctes (les vrais positifs) et 15 sont incorrectes (les faux positifs), la précision est calculée comme suit :

85 vrais positifs/(85 vrais positifs + 15 faux positifs) = valeur de précision de 0,85

Toutefois, si le modèle ne prédit correctement que 40 images sur 100 prédictions d'anomalies, la valeur de précision obtenue est inférieure à 0,40 (c'est-à-dire 40/(40 + 60) = 0,40). Dans ce cas, votre

modèle fait plus de prédictions incorrectes que de prédictions correctes. Pour résoudre ce problème, pensez à apporter des améliorations à votre modèle. Pour de plus amples informations, veuillez consulter Étape 2 : Améliorez votre modèle.

Pour plus d'informations, consultez Précision et rappel.

Rappel

La métrique de rappel répond à la question suivante : sur le nombre total d'images anormales dans l'ensemble de données de test, combien sont correctement prédites comme anormales ?

La métrique de rappel est utile dans les situations où le coût d'un faux négatif est élevé. Par exemple, lorsque le coût de ne pas retirer une pièce défectueuse est élevé. Amazon Lookout for Vision fournit une valeur de métrique de rappel récapitulative pour l'ensemble de données de test.

Le rappel est la fraction des images de test anormales qui ont été détectées correctement. Il s'agit d'une mesure de la fréquence à laquelle le modèle peut prédire correctement une image anormale, alors qu'elle est réellement présente dans les images de votre jeu de données de test. La formule de rappel est calculée comme suit :

Valeur de rappel = vrais positifs/(vrais positifs + faux négatifs)

La plage de rappel est comprise entre 0 et 1. La console Amazon Lookout for Vision affiche le rappel sous forme de pourcentage (0 à 100).

Une valeur de rappel plus élevée indique qu'un plus grand nombre d'images anormales sont correctement identifiées. Supposons, par exemple, que le jeu de données de test contienne 100 images anormales. Si le modèle détecte correctement 90 des 100 images anormales, le rappel est le suivant :

90 vrais positifs/(90 vrais positifs + 10 faux négatifs) = 0,90 valeur de rappel

Une valeur de rappel de 0,90 indique que votre modèle prédit correctement la plupart des images anormales du jeu de données de test. Si le modèle ne prédit correctement que 20 des images anormales, le rappel est inférieur à 0,20 (c'est-à-dire 20/(20 + 80) = 0,20).

Dans ce cas, vous devriez envisager d'apporter des améliorations à votre modèle. Pour de plus amples informations, veuillez consulter Étape 2 : Améliorez votre modèle.

Pour plus d'informations, consultez Précision et rappel.

Score de F1

Amazon Lookout for Vision fournit un score de performance moyen du modèle pour le jeu de données de test. Plus précisément, les performances du modèle pour la classification des anomalies sont mesurées par la métrique de score F1, qui est la moyenne harmonique des scores de précision et de rappel.

Le score F1 est une mesure agrégée qui prend en compte à la fois la précision et le rappel. 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. Par exemple, pour un modèle avec une précision de 0,9 et un rappel de 1,0, le score F1 est de 0,947.

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, ce qui indique des problèmes avec le modèle.

Pour plus d'informations, consultez Score F1.

Intersection moyenne au-dessus de l'Union (IoU)

Pourcentage moyen de chevauchement entre les masques d'anomalies dans les images de test et les masques d'anomalies que le modèle prédit pour les images de test. Amazon Lookout for Vision renvoie l'IoU moyen pour chaque étiquette d'anomalie et n'est renvoyé que par les modèles de segmentation d'image.

Une faible valeur en pourcentage indique que le modèle ne correspond pas exactement aux masques prévus pour une étiquette avec les masques des images de test.

L'image suivante présente un faible Iou. Le masque orange est la prédiction du modèle et ne recouvre pas étroitement le masque bleu qui représente le masque dans une image de test.

L'image suivante a un Iou plus élevé. Le masque bleu (image de test) est étroitement recouvert par le masque orange (masque prédit).

Résultats des tests

Pendant le test, le modèle prédit la classification de chaque image de test dans le jeu de données de test. Le résultat de chaque prédiction est comparé à l'étiquette (normale ou anormale) de l'image de test correspondante comme suit :

- Prédire correctement qu'une image est anormale est considéré comme un véritable positif.
- Prédire à tort qu'une image est anormale est considéré comme un faux positif.
- Prédire correctement qu'une image est normale est considéré comme un véritable négatif.
- Prédire à tort qu'une image est normale est considéré comme un faux négatif.

Si le modèle est un modèle de segmentation, le modèle prédit également des masques et des étiquettes d'anomalies pour la localisation des anomalies sur l'image de test.

Amazon Lookout for Vision utilise les résultats des comparaisons pour générer les indicateurs de performance.

Étape 2 : Améliorez votre modèle

Les indicateurs de performance peuvent indiquer que vous pouvez améliorer votre modèle. Par exemple, si le modèle ne détecte pas toutes les anomalies dans le jeu de données de test, votre modèle présente un faible taux de rappel (c'est-à-dire que la métrique de rappel a une faible valeur). Si vous devez améliorer votre modèle, considérez les points suivants :

- Vérifiez que les images des jeux de données d'entraînement et de test sont correctement étiquetées.

- Réduisez la variabilité des conditions de capture d'image telles que l'éclairage et la pose des objets, et entraînez votre modèle sur des objets du même type.
- Assurez-vous que vos images ne présentent que le contenu requis. Par exemple, si votre projet détecte des anomalies dans des pièces de machines, assurez-vous qu'aucun autre objet ne figure dans vos images.
- Ajoutez d'autres images étiquetées à vos ensembles de données de train et de test. Si la mémorisation et la précision de votre jeu de données de test sont excellentes, mais que le modèle est peu performant lors du déploiement, votre jeu de données de test n'est peut-être pas suffisamment représentatif et vous devez l'étendre.
- Si votre jeu de données de test se traduit par une mémorisation et une précision médiocres, déterminez dans quelle mesure les anomalies et les conditions de capture d'image des ensembles de données d'entraînement et de test correspondent. Si vos images d'entraînement ne sont pas représentatives des anomalies et des conditions attendues, mais que les images des images de test le sont, ajoutez des images à l'ensemble de données d'entraînement avec les anomalies et conditions attendues. Si les images du jeu de données de test ne sont pas dans les conditions attendues, mais que les images d'entraînement le sont, mettez à jour le jeu de données de test.

  Pour de plus amples informations, veuillez consulter Ajout d’autres images. Une autre méthode pour ajouter des images étiquetées à votre jeu de données d'entraînement consiste à exécuter une tâche de détection d'essai et à vérifier les résultats. Vous pouvez ensuite ajouter les images vérifiées à l'ensemble de données d'entraînement. Pour de plus amples informations, veuillez consulter Vérification de votre modèle à l'aide d'une tâche de détection d'essai.
- Assurez-vous de disposer d'images normales et anormales suffisamment diverses dans votre ensemble de données d'entraînement et de test. Les images doivent représenter le type d'images normales et anormales que rencontrera votre modèle. Par exemple, lorsque vous analysez des circuits imprimés, vos images normales doivent représenter les variations de position et de soudure des composants, tels que les résistances et les transistors. Les images anormales doivent représenter les différents types d'anomalies que le système peut rencontrer, telles que des composants égarés ou manquants.
- Si votre modèle présente un IOU moyen faible pour les types d'anomalies détectés, vérifiez les sorties du masque du modèle de segmentation. Dans certains cas d'utilisation, tels que les rayures, le modèle peut produire des rayures très proches de celles de la base sur les images de test, mais avec un faible chevauchement des pixels. Par exemple, deux lignes parallèles distantes de 1 pixel. Dans ces cas, l'IOU moyen n'est pas un indicateur fiable pour mesurer le succès des prédictions.

- Si la taille de l'image est petite ou si la résolution de l'image est faible, envisagez de capturer des images à une résolution supérieure. Les dimensions de l'image peuvent aller de 64 x 64 pixels à 4096 pixels x 4096 pixels.
- Si la taille de l'anomalie est faible, pensez à diviser les images en vignettes distinctes et à utiliser les images en mosaïque pour l'entraînement et les tests. Cela permet au modèle de voir les défauts à une taille plus grande sur une image.

Après avoir amélioré votre jeu de données d'entraînement et de test, réentraînez et réévaluez votre modèle. Pour de plus amples informations, veuillez consulter Entraînement de votre modèle.

Si les mesures indiquent que les performances de votre modèle sont acceptables, vous pouvez vérifier ses performances en ajoutant les résultats d'une tâche de détection d'essai au jeu de données de test. Après le réentraînement, les indicateurs de performance devraient confirmer les indicateurs de performance de l'entraînement précédent. Pour de plus amples informations, veuillez consulter Vérification de votre modèle à l'aide d'une tâche de détection d'essai.

Consultation des métriques de performances

Vous pouvez obtenir des indicateurs de performance à partir de la console et en appelant l'`DescribeModel`opération.

Rubriques

- Affichage des indicateurs de performance (console)
- Affichage des indicateurs de performance (SDK)

Affichage des indicateurs de performance (console)

Une fois l'entraînement terminé, la console affiche les indicateurs de performance.

La console Amazon Lookout for Vision affiche les indicateurs de performance suivants pour les classifications effectuées lors des tests :

- Précision
- Rappel
- Score de F1

Si le modèle est un modèle de segmentation, la console affiche également les mesures de performance suivantes pour chaque étiquette d'anomalie :

- Le nombre d'images de test où l'étiquette d'anomalie a été détectée.
- Score de F1
- Intersection moyenne au-dessus de l'Union (IoU)

La section de présentation des résultats des tests vous indique le total des prédictions correctes et incorrectes pour les images du jeu de données de test. Vous pouvez également consulter les attributions d'étiquettes prévues et réelles pour les images individuelles dans le jeu de données de test.

La procédure suivante montre comment obtenir les mesures de performance à partir de la vue de liste des modèles d'un projet.

Pour consulter les indicateurs de performance (console)

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Choisissez Démarrer.
3. Dans le volet de navigation de gauche, choisissez Projets.
4. Dans la vue des projets, choisissez le projet qui contient la version du modèle que vous souhaitez consulter.
5. Dans le volet de navigation de gauche, sous le nom du projet, sélectionnez Modèles.
6. Dans la vue de liste des modèles, choisissez les versions du modèle que vous souhaitez consulter.
7. Sur la page des détails du modèle, consultez les mesures de performance dans l'onglet Mesures de performance.
8. Remarques :

    a. La section Mesures de performance du modèle contient les mesures globales du modèle (précision, rappel, score F1) pour les prédictions de classification établies par le modèle pour les images de test.

    b. Si le modèle est un modèle de segmentation d'image, la section Performances par étiquette contient le nombre d'images de test où l'étiquette d'anomalie a été détectée. Vous

pouvez également consulter les statistiques (score F1, IoU moyen) pour chaque étiquette d'anomalie.

c. La section Vue d'ensemble des résultats des tests fournit les résultats de chaque image de test utilisée par Lookout for Vision pour évaluer le modèle. Elle comprend les éléments suivants :

- Le nombre total de prédictions de classification correctes (vrais positifs) et incorrectes (faux négatifs) (normales ou anormales) pour toutes les images de test.
- La prédiction de classification pour chaque image de test. Si Correct apparaît sous une image, la classification prévue correspond à la classification réelle de l'image. Sinon, le modèle n'a pas correctement classé l'image.
- Avec un modèle de segmentation d'image, vous pouvez voir les étiquettes d'anomalie attribuées par le modèle à l'image et les masques sur l'image qui correspondent aux couleurs des étiquettes d'anomalie.

Affichage des indicateurs de performance (SDK)

Vous pouvez utiliser cette DescribeModelopération pour obtenir les mesures de performance récapitulatives (classification) du modèle, le manifeste d'évaluation et les résultats de l'évaluation d'un modèle.

Obtenir les indicateurs de performance récapitulatifs

Les mesures de performance récapitulatives pour les prédictions de classification effectuées par le modèle lors des tests (PrécisionRappel,, etScore de F1) sont renvoyées dans le `Performance` champ renvoyé par un appel à`DescribeModel`.

```
        "Performance": {
            "F1Score": 0.8,
            "Recall": 0.8,
            "Precision": 0.9
        },
```

Le `Performance` champ n'inclut pas les mesures de performance des étiquettes d'anomalies renvoyées par un modèle de segmentation. Vous pouvez les obtenir `EvaluationResult` sur le terrain. Pour de plus amples informations, veuillez consulter Révision du résultat de l'évaluation.

Pour plus d'informations sur les mesures de performance récapitulatives, voirÉtape 1 : Évaluez les performances de votre modèle. Pour obtenir un exemple de code, consultez Affichage de vos modèles (SDK).

Utilisation du manifeste d'évaluation

Le manifeste d'évaluation fournit des mesures de prédiction de test pour les images individuelles utilisées pour tester un modèle. Pour chaque image du jeu de données de test, une ligne JSON contient les informations de test d'origine (vérité fondamentale) et la prédiction du modèle pour l'image. Amazon Lookout for Vision stocke le manifeste d'évaluation dans un compartiment Amazon S3. Vous pouvez obtenir l'emplacement à partir du `EvaluationManifest` champ dans la réponse de `DescribeModel` l'opération.

```
"EvaluationManifest": {
            "Bucket": "lookoutvision-us-east-1-nnnnnnnnnn",
            "Key": "my-sdk-project-model-output/EvaluationManifest-my-sdk-
project-1.json"
        }
```

Le format du nom de fichier est`EvaluationManifest-`*`project name`*`.json`. Pour obtenir un exemple de code, consultez Visualisation de vos modèles.

Dans l'exemple de ligne JSON suivant, `class-name` il s'agit de la vérité fondamentale du contenu de l'image. Dans cet exemple, l'image contient une anomalie. Le `confidence` champ indique le niveau de confiance d'Amazon Lookout for Vision dans les prévisions.

```
{
    "source-ref"*: "s3://customerbucket/path/to/image.jpg",
    "source-ref-metadata": {
        "creation-date": "2020-05-22T21:33:37.201882"
    },

    // Test dataset ground truth
    "anomaly-label": 1,
    "anomaly-label-metadata": {
        "class-name": "anomaly",
        "type": "groundtruth/image-classification",
        "human-annotated": "yes",
        "creation-date": "2020-05-22T21:33:37.201882",
        "job-name": "labeling-job/anomaly-detection"
```

```
    },
    // Anomaly label detected by Lookout for Vision
    "anomaly-label-detected": 1,
    "anomaly-label-detected-metadata": {
        "class-name": "anomaly",
        "confidence": 0.9,
        "type": "groundtruth/image-classification",
        "human-annotated": "no",
        "creation-date": "2020-05-22T21:33:37.201882",
        "job-name": "training-job/anomaly-detection",
        "model-arn": "lookoutvision-some-model-arn",
        "project-name": "lookoutvision-some-project-name",
        "model-version": "lookoutvision-some-model-version"
    }
}
```

Révision du résultat de l'évaluation

Le résultat de l'évaluation contient les mesures de performance agrégées suivantes (classification) pour l'ensemble complet des images de test :

- Précision
- Rappel
- Courbe ROC (non montrée dans la console)
- Précision moyenne (non affichée dans la console)
- Score de F1

Le résultat de l'évaluation inclut également le nombre d'images utilisées pour l'entraînement et le test du modèle.

Si le modèle est un modèle de segmentation, le résultat de l'évaluation inclut également les mesures suivantes pour chaque étiquette d'anomalie trouvée dans l'ensemble de données de test :

- Précision
- Rappel
- Score de F1
- Intersection moyenne au-dessus de l'Union (IoU)

Amazon Lookout for Vision stocke le résultat de l'évaluation dans un compartiment Amazon S3. Vous pouvez obtenir l'emplacement en vérifiant la valeur de `EvaluationResult` dans la réponse de `DescribeModel` l'opération.

```
"EvaluationResult": {
            "Bucket": "lookoutvision-us-east-1-nnnnnnnnnn",
            "Key": "my-sdk-project-model-output/EvaluationResult-my-sdk-project-1.json"
        }
```

Le format du nom de fichier est`EvaluationResult-`*`project name`*`.json`. Pour obtenir un exemple, consultez Visualisation de vos modèles.

Le schéma suivant montre le résultat de l'évaluation.

```
   {
       "Version": 1,
       "EvaluationDetails":
       {
           "ModelArn": "string", // The Amazon Resource Name (ARN) of the model
version.
           "EvaluationEndTimestamp": "string",   // The UTC date and time that
evaluation finished.
           "NumberOfTrainingImages": int,        // The number of images that were
successfully used for training.
           "NumberOfTestingImages": int          // The number of images that were
successfully used for testing.
       },
       "AggregatedEvaluationResults":
       {
           "Metrics":
           {                       //Classification metrics.
               "ROCAUC": float,             // ROC area under the curve.
               "AveragePrecision": float,  // The average precision of the model.
               "Precision": float,         // The overall precision of the model.
               "Recall": float,            // The overall recall of the model.
               "F1Score": float,           // The overal F1 score for the model.

               "PixelAnomalyClassMetrics":     //Segmentation metrics.
               [
                   {
                       "Precision": float,       // The precision for the anomaly
label.
                       "Recall": float,          // The recall for the anomaly label.
```

```
                        "F1Score": float,         // The F1 score for the anomaly
label.
                        "AIOU" : float,           // The average Intersection Over
Union for the anomaly label.
                        "ClassName": "string"   // The anomaly label.
                    }
                ]
            }
        }
    }
```

Vérification de votre modèle à l'aide d'une tâche de détection d'essai

Si vous souhaitez vérifier ou améliorer la qualité de votre modèle, vous pouvez exécuter une tâche de détection d'essai. Une tâche de détection d'essai détecte les anomalies dans les nouvelles images que vous fournissez.

Vous pouvez vérifier les résultats de détection et ajouter les images vérifiées à votre jeu de données. Si vous disposez d'ensembles de données d'entraînement et de test distincts, les images vérifiées sont ajoutées au jeu de données d'entraînement.

Vous pouvez vérifier des images provenant de votre ordinateur local ou des images situées dans un compartiment Amazon S3. Si vous souhaitez ajouter des images vérifiées à l'ensemble de données, les images situées dans un compartiment S3 doivent se trouver dans le même compartiment S3 que les images de votre ensemble de données.

> **Note**
>
> Pour exécuter une tâche de détection d'essai, assurez-vous que la gestion des versions est activée dans votre compartiment S3. Pour plus d'informations, consultez la section Utilisation du versionnement. Le bucket de console est créé avec le versionnement activé.

Par défaut, vos images sont chiffrées à l'aide d'une clé détenue et gérée par AWS. Vous pouvez également choisir d'utiliser votre propre clé AWS Key Management Service (KMS). Pour plus d'informations, consultez Concepts d'AWS Key Management Service.

Rubriques

- Exécution d'une tâche de détection d'essai
- Vérification des résultats de détection des essais
- Corriger les étiquettes de segmentation avec l'outil d'annotation

Exécution d'une tâche de détection d'essai

Procédez comme suit pour exécuter une tâche de détection d'essai.

Pour exécuter une version d'essai de détection (console)

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Choisissez Démarrer.
3. Dans le volet de navigation de gauche, choisissez Projets.
4. Dans la vue des projets, choisissez le projet qui contient la version du modèle que vous souhaitez consulter.
5. Dans le volet de navigation de gauche, sous le nom du projet, sélectionnez Trial detections.
6. Dans la vue des détections d'essai, choisissez l'option Exécuter la détection d'essai.
7. Sur la page Exécuter la détection d'essai, entrez le nom de votre tâche de détection d'essai dans Nom de la tâche.
8. Dans Choisir un modèle, choisissez la version du modèle que vous souhaitez utiliser.
9. Importez les images en fonction de leur source comme suit :

   - Si vous importez vos images sources depuis un compartiment Amazon S3, entrez l'URI S3.

     > **Tip**
     >
     > Si vous utilisez les exemples d'images Getting Started, utilisez le dossier extra_images. L'URI Amazon S3 est `s3://your bucket/circuitboard/extra_images`.

   - Si vous téléchargez des images depuis votre ordinateur, ajoutez-les après avoir sélectionné Détecter les anomalies.
10. (Facultatif) Si vous souhaitez utiliser votre propre clé de chiffrement AWS KMS, procédez comme suit :

a. Pour le chiffrement des données d'image, choisissez Personnaliser les paramètres de chiffrement (avancés).

b. Dans encryption.aws_kms_key, entrez le nom de ressource Amazon (ARN) de votre clé ou choisissez une clé AWS KMS existante. Pour créer une clé, choisissez Créer une clé AWS KMS.

11. Choisissez Détecter les anomalies, puis sélectionnez Exécuter la détection des essais pour démarrer la tâche de détection des essais.
12. Vérifiez l'état actuel dans la vue des détections d'essais. La détection de l'essai peut prendre un certain temps.

Vérification des résultats de détection des essais

La vérification des résultats d'une détection d'essai peut vous aider à améliorer votre modèle.

Si les indicateurs de performance sont médiocres, améliorez votre modèle en exécutant une détection d'essai, puis en ajoutant des images vérifiées au jeu de données (ensemble de données d'entraînement, si vous avez des ensembles de données distincts).

Si les indicateurs de performance du modèle sont bons, mais que les résultats d'une détection d'essai sont médiocres, vous pouvez améliorer votre modèle en ajoutant des images vérifiées à l'ensemble de données (ensemble de données d'apprentissage). Si vous disposez d'un ensemble de données de test distinct, pensez à ajouter d'autres images à l'ensemble de données de test.

Après avoir ajouté des images vérifiées à votre jeu de données, réentraînez et réévaluez votre modèle. Pour de plus amples informations, veuillez consulter Entraînement de votre modèle.

Pour vérifier les résultats d'un essai de détection

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Dans le volet de navigation de gauche, choisissez Projets.
3. Sur la page Projets, choisissez le projet que vous voulez utiliser. Le tableau de bord de votre projet s'affiche.
4. Dans le volet de navigation de gauche, choisissez Trial detections.
5. Choisissez la détection d'essai que vous souhaitez vérifier.
6. Sur la page de détection des essais, choisissez Vérifier les prédictions de la machine.

7. Choisissez Sélectionner toutes les images de cette page.
8. Si les prévisions sont correctes, choisissez Vérifier comme étant correctes. Sinon, sélectionnez Vérifier comme incorrect. La prédiction et le score de confiance des prédictions sont affichés sous chaque image.
9. Si vous devez modifier le libellé d'une image, procédez comme suit :

    a. Choisissez Correct ou Incorrect sous l'image.

    b. Si vous ne parvenez pas à déterminer l'étiquette correcte pour une image, agrandissez l'image en la sélectionnant dans la galerie.

> **Note**
>
> Vous pouvez filtrer les étiquettes des images en choisissant l'étiquette ou l'état d'étiquette souhaité dans la section Filtres. Vous pouvez trier par indice de confiance dans la section Options de tri.

10. Si votre modèle est un modèle de segmentation et que le masque ou l'étiquette d'anomalie d'une image est incorrect, choisissez Zone anormale sous l'image et ouvrez l'outil d'annotation. Mettez à jour les informations de segmentation en procédantCorriger les étiquettes de segmentation avec l'outil d'annotation.
11. Répétez les étapes 7 à 10 sur chaque page si nécessaire jusqu'à ce que toutes les images aient été vérifiées.
12. Choisissez Ajouter des images vérifiées à l'ensemble de données. Si vous avez des ensembles de données distincts, les images sont ajoutées au jeu de données d'entraînement.
13. Entraînez votre modèle. Pour de plus amples informations, veuillez consulter the section called “Entraînement de votre modèle”.

Corriger les étiquettes de segmentation avec l'outil d'annotation

L'outil d'annotation permet de segmenter une image en marquant les zones anormales à l'aide d'un masque.

Pour corriger les étiquettes de segmentation d'une image à l'aide de l'outil d'annotation

1. Ouvrez l'outil d'annotation en sélectionnant une zone anormale sous une image dans la galerie de jeux de données.

2. Si l'étiquette d'anomalie d'un masque n'est pas correcte, choisissez le masque, puis choisissez la bonne étiquette d'anomalie sous Étiquettes d'anomalie. Si nécessaire, choisissez Ajouter une étiquette d'anomalie pour ajouter une nouvelle étiquette d'anomalie.
3. Si le masque n'est pas correct, choisissez un outil de dessin en bas de page et dessinez des masques qui recouvrent étroitement les zones anormales pour l'étiquette d'anomalie. L'image suivante est un exemple de masque qui couvre étroitement une anomalie.

Voici un exemple de masque médiocre qui ne couvre pas parfaitement une anomalie.

4. Si vous avez d'autres images à corriger, choisissez Next et répétez les étapes 2 et 3.
5. Choisissez Soumettre et fermer pour terminer la mise à jour des images.

Exécution de votre modèle Amazon Lookout for Vision entraîné

Pour détecter des anomalies dans les images avec votre modèle, vous devez d'abord démarrer votre modèle avec l'StartModelopération. La console Amazon Lookout for Vision AWS CLI fournit des commandes que vous pouvez utiliser pour démarrer et arrêter votre modèle. Cette section contient un exemple de code que vous pouvez utiliser.

Après le démarrage de votre modèle, vous pouvez utiliser cette `DetectAnomalies` opération pour détecter des anomalies dans une image. Pour de plus amples informations, veuillez consulter Détecter des anomalies dans une image.

Rubriques

- Unités d’inférence
- Zones de disponibilité
- Démarrage de votre modèle Amazon Lookout for Vision
- Arrêter votre modèle Amazon Lookout for Vision

Unités d’inférence

Lorsque vous démarrez votre modèle, Amazon Lookout for Vision fournit au moins une ressource de calcul, appelée unité d'inférence. Vous spécifiez le nombre d'unités d'inférence à utiliser dans le paramètre `MinInferenceUnits` d'entrée de l'`StartModel`API. L'allocation par défaut pour un modèle est de 1 unité d'inférence.

> **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 la page de tarification d'Amazon Lookout for Vision. Si vous n'arrêtez pas explicitement votre modèle en appelant StopModel, 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 :

- Algorithme utilisé par Lookout for Vision pour entraîner le modèle. Lorsque vous entraînez un modèle, plusieurs modèles sont entraînés. Lookout for Vision sélectionne le modèle présentant les meilleures performances en fonction de la taille du jeu de données et de sa composition d'images normales et anormales.
- Les images à haute résolution nécessitent plus de temps pour l'analyse.
- Les images de petite taille (mesurées en MBs) sont analysées plus rapidement que les images plus grandes.

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, Amazon Lookout for Vision peut automatiquement ajuster le nombre d'unités d'inférence utilisées par votre modèle. À mesure que la demande augmente, Amazon Lookout for Vision ajoute des unités d'inférence supplémentaires au modèle et les supprime lorsque la demande diminue.

Pour permettre à Lookout for Vision de redimensionner 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, Lookout for Vision ne redimensionnera pas automatiquement votre modèle, en utilisant 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 Lookout for Vision. Spécifiez plutôt le paramètre d'entrée `MaxInferenceUnits` de l'opération `StartModel`.

Lookout for Vision fournit 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 auxquelles Lookout for Vision augmente ou diminue.
`InServiceInferenceUnits`	Le nombre d'unités d'inférence utilisées par le modèle.

Si `DesiredInferenceUnits` =`InServiceInferenceUnits`, Lookout for Vision ne redimensionne pas actuellement le nombre d'unités d'inférence.

Si `DesiredInferenceUnits` >`InServiceInferenceUnits`, Lookout for Vision passe à la valeur `DesiredInferenceUnits` de.

Si `DesiredInferenceUnits` <`InServiceInferenceUnits`, Lookout for Vision est réduit à la valeur `DesiredInferenceUnits` de.

Pour plus d'informations concernant les métriques renvoyées par Lookout for Vision et les dimensions de filtrage, consultez la section Surveillance de Lookout for Vision avec Amazon. CloudWatch

Pour connaître le nombre maximum d'unités d'inférence que vous avez demandées pour un modèle, appelez DescribeModel et vérifiez le champ `MaxInferenceUnits` dans la réponse.

Zones de disponibilité

Amazon Lookout for Vision distribue des unités d'inférence dans plusieurs zones de disponibilité au sein AWS d'une région afin d'améliorer 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 DetectAnomaliessont 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 la zone de disponibilité AWS réparée, les unités d'inférence sont redémarrées et leur capacité maximale est rétablie.

En cas de défaillance d'une seule unité d'inférence, Amazon Lookout for Vision 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 de votre modèle Amazon Lookout for Vision

Avant de pouvoir utiliser un modèle Amazon Lookout for Vision pour détecter des anomalies, vous devez d'abord démarrer le modèle. Vous démarrez un modèle en appelant l'StartModelAPI et en transmettant ce qui suit :

- ProjectName— Le nom du projet qui contient le modèle que vous souhaitez démarrer.
- ModelVersion— La version du modèle que vous souhaitez démarrer.

- MinInferenceUnits— Le nombre minimum d'unités d'inférence. Pour de plus amples informations, veuillez consulter Unités d’inférence.
- (Facultatif) MaxInferenceUnits— Le nombre maximum d'unités d'inférence qu'Amazon Lookout for Vision peut utiliser pour redimensionner automatiquement le modèle. Pour de plus amples informations, veuillez consulter Mise à l’échelle automatique d’unités d’inférence.

La console Amazon Lookout for Vision fournit un exemple de code que vous pouvez utiliser pour démarrer et arrêter un modèle.

> **Note**
>
> La durée de fonctionnement de votre modèle vous est facturée. Pour arrêter un modèle en cours d’exécution, consultez Arrêter votre modèle Amazon Lookout for Vision.
> Vous pouvez utiliser le AWS SDK pour visualiser les modèles en cours d'exécution dans toutes les AWS régions dans lesquelles Lookout for Vision est disponible. Pour un exemple de code, voir find_running_models.py.

Rubriques

- Démarrage de votre modèle (console)
- Démarrage de votre modèle Amazon Lookout for Vision (SDK)

Démarrage de votre modèle (console)

La console Amazon Lookout for Vision fournit AWS CLI une commande que vous pouvez utiliser pour démarrer un modèle. Une fois le modèle démarré, vous pouvez commencer à détecter des anomalies dans les images. Pour de plus amples informations, veuillez consulter Détecter des anomalies dans une image.

Pour démarrer un modèle (console)

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. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
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. Sur la page de détails du modèle, choisissez Utiliser le modèle, puis choisissez Intégrer l'API au cloud.

   > **Tip**
   >
   > Si vous souhaitez déployer votre modèle sur un appareil périphérique, choisissez Create model packaging job. Pour de plus amples informations, veuillez consulter Emballage de votre modèle Amazon Lookout for Vision.

8. Sous les commandes de l'AWS CLI, copiez la commande AWS CLI qui appelle`start-model`.
9. À l'invite de commande, entrez la commande `start-model` que vous avez copiée à l'étape précédente. Si vous utilisez le `lookoutvision` profil pour obtenir des informations d'identification, ajoutez le `--profile lookoutvision-access` paramètre.
10. Dans la console, sélectionnez Modèles dans la page de navigation de gauche.
11. Consultez la colonne État pour connaître l'état actuel du modèle. Lorsque le statut est hébergé, vous pouvez utiliser le modèle pour détecter des anomalies dans les images. Pour de plus amples informations, veuillez consulter Détecter des anomalies dans une image.

Démarrage de votre modèle Amazon Lookout for Vision (SDK)

Vous démarrez un modèle en appelant l'StartModelopération.

Le démarrage d'un modèle peut prendre un certain temps. Vous pouvez vérifier l'état actuel en appelant DescribeModel. Pour de plus amples informations, veuillez consulter Visualisation de vos modèles.

Pour démarrer votre modèle (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 les valeurs suivantes :

- `project-name`au nom du projet contenant le modèle que vous souhaitez démarrer.
- `model-version`à la version du modèle que vous souhaitez démarrer.
- `--min-inference-units`au nombre d'unités d'inférence que vous souhaitez utiliser.
- (Facultatif) `--max-inference-units` jusqu'au nombre maximum d'unités d'inférence qu'Amazon Lookout for Vision peut utiliser pour redimensionner automatiquement le modèle.

```
aws lookoutvision start-model --project-name "project name"\
  --model-version model version\
  --min-inference-units minimum number of units\
  --max-inference-units max number of units \
  --profile lookoutvision-access
```

Python

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

```
    @staticmethod
    def start_model(
            lookoutvision_client, project_name, model_version,
min_inference_units, max_inference_units = None):
        """
        Starts the hosting of a Lookout for Vision model.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name:  The name of the project that contains the version
of the
                              model that you want to start hosting.
        :param model_version: The version of 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: (Optional) The maximum number of inference
units that
```

```
        Lookout for Vision can use to automatically scale the model.
        """
        try:
            logger.info(
                "Starting model version %s for project %s", model_version,
project_name)

            if max_inference_units is None:
                lookoutvision_client.start_model(
                    ProjectName = project_name,
                    ModelVersion = model_version,
                    MinInferenceUnits = min_inference_units)

            else:
                lookoutvision_client.start_model(
                    ProjectName = project_name,
                    ModelVersion = model_version,
                    MinInferenceUnits = min_inference_units,
                    MaxInferenceUnits = max_inference_units)

            print("Starting hosting...")

            status = ""
            finished = False

            # Wait until hosted or failed.
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name, ModelVersion=model_version)
                status = model_description["ModelDescription"]["Status"]

                if status == "STARTING_HOSTING":
                    logger.info("Host starting in progress...")
                    time.sleep(10)
                    continue

                if status == "HOSTED":
                    logger.info("Model is hosted and ready for use.")
                    finished = True
                    continue

                logger.info("Model hosting failed and the model can't be used.")
                finished = True
```

```
            if status != "HOSTED":
                logger.error("Error hosting model: %s", status)
                raise Exception(f"Error hosting model: {status}")
        except ClientError:
            logger.exception("Couldn't host model.")
            raise
```

Java V2

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

```
/**
 * Starts hosting an Amazon Lookout for Vision model. Returns when the model has
 * started or if hosting fails. You are charged for the amount of time that a
 * model is hosted. To stop hosting a model, use the StopModel operation.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that contains the model that you
 *                    want to host.
 * @modelVersion The version of the model that you want to host.
 * @minInferenceUnits The number of inference units to use for hosting.
 * @maxInferenceUnits The maximum number of inference units that Lookout for
 *                    Vision can use for automatically scaling the model. If the
 *                    value is null, automatic scaling doesn't happen.
 * @return ModelDescription The description of the model, which includes the
 *         model hosting status.
 */
public static ModelDescription startModel(LookoutVisionClient lfvClient, String
 projectName, String modelVersion,
        Integer minInferenceUnits, Integer maxInferenceUnits) throws
 LookoutVisionException, InterruptedException {

    logger.log(Level.INFO, "Starting Model version {0} for project {1}.",
            new Object[] { modelVersion, projectName });

    StartModelRequest startModelRequest = null;

    if (maxInferenceUnits == null) {

        startModelRequest =
 StartModelRequest.builder().projectName(projectName).modelVersion(modelVersion)
                .minInferenceUnits(minInferenceUnits).build();
```

```
   } else {
       startModelRequest =
StartModelRequest.builder().projectName(projectName).modelVersion(modelVersion)

.minInferenceUnits(minInferenceUnits).maxInferenceUnits(maxInferenceUnits).build();
   }

   // Start hosting the model.
   lfvClient.startModel(startModelRequest);

   DescribeModelRequest describeModelRequest =
DescribeModelRequest.builder().projectName(projectName)
           .modelVersion(modelVersion).build();

   ModelDescription modelDescription = null;

   boolean finished = false;
   // Wait until model is hosted or failure occurs.
   do {

       modelDescription =
lfvClient.describeModel(describeModelRequest).modelDescription();

       switch (modelDescription.status()) {

       case HOSTED:
           logger.log(Level.INFO, "Model version {0} for project {1} is
running.",
                   new Object[] { modelVersion, projectName });
           finished = true;
           break;

       case STARTING_HOSTING:
           logger.log(Level.INFO, "Model version {0} for project {1} is
starting.",
                   new Object[] { modelVersion, projectName });

           TimeUnit.SECONDS.sleep(60);

           break;
       case HOSTING_FAILED:
           logger.log(Level.SEVERE, "Hosting failed for model version {0} for
project {1}.",
                   new Object[] { modelVersion, projectName });
```

```
            finished = true;
            break;

        default:
            logger.log(Level.SEVERE, "Unexpected error when hosting model
 version {0} for project {1}: {2}.",
                    new Object[] { projectName, modelVersion,
 modelDescription.status() });
            finished = true;
            break;

        }

    } while (!finished);

    logger.log(Level.INFO, "Finished starting model version {0} for project {1}
 status: {2}",
            new Object[] { modelVersion, projectName,
 modelDescription.statusMessage() });

    return modelDescription;

}
```

3. Si le résultat du code est le cas`Model is hosted and ready for use`, vous pouvez utiliser le modèle pour détecter des anomalies dans les images. Pour de plus amples informations, veuillez consulter Détecter des anomalies dans une image.

Arrêter votre modèle Amazon Lookout for Vision

Pour arrêter un modèle en cours d'exécution, vous devez lancer l'`StopModel`opération et transmettre ce qui suit :

- Projet : nom du projet qui contient le modèle que vous souhaitez arrêter.
- ModelVersion— Version du modèle que vous souhaitez arrêter.

La console Amazon Lookout for Vision fournit un exemple de code que vous pouvez utiliser pour arrêter un modèle.

> **Note**
>
> La durée de fonctionnement de votre modèle vous est facturée.

Rubriques

- Arrêt de votre modèle (console)
- Arrêter votre modèle Amazon Lookout for Vision (SDK)

Arrêt de votre modèle (console)

Procédez comme indiqué dans la procédure suivante pour arrêter votre modèle d'utiliser la console.

Pour arrêter votre modèle (console)

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. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
3. Choisissez Démarrer.
4. Dans le volet de navigation de gauche, choisissez Projets.
5. Sur la page Ressources des projets, choisissez le projet contenant le modèle en cours d'exécution que vous souhaitez arrêter.
6. Dans la section Modèles, choisissez le modèle que vous souhaitez arrêter.
7. Sur la page de détails du modèle, choisissez Utiliser le modèle, puis choisissez Intégrer l'API au cloud.
8. Sous les commandes de l'AWS CLI, copiez la commande AWS CLI qui appelle`stop-model`.
9. À l'invite de commande, entrez la commande `stop-model` que vous avez copiée à l'étape précédente. Si vous utilisez le `lookoutvision` profil pour obtenir des informations d'identification, ajoutez le `--profile lookoutvision-access` paramètre.
10. Sur la console, sélectionnez Modèles dans la page de navigation de gauche.
11. Consultez la colonne État pour connaître l'état actuel du modèle. Le modèle s'arrête lorsque la valeur de la colonne Status est Training complete.

Arrêter votre modèle Amazon Lookout for Vision (SDK)

Vous arrêtez un modèle en appelant l'StopModelopération.

L'arrêt d'un modèle peut prendre un certain temps. Pour vérifier le statut actuel, utilisez `DescribeModel`.

Pour arrêter votre modèle (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 les valeurs suivantes :

   - `project-name`au nom du projet contenant le modèle que vous souhaitez arrêter.
   - `model-version`à la version du modèle que vous souhaitez arrêter.

   ```
   aws lookoutvision stop-model --project-name "project name"\
       --model-version model version \
     --profile lookoutvision-access
   ```

   Python

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

   ```
       @staticmethod
       def stop_model(lookoutvision_client, project_name, model_version):
           """
           Stops a running Lookout for Vision Model.

           :param lookoutvision_client: A Boto3 Lookout for Vision client.
           :param project_name: The name of the project that contains the version
   of
                                the model that you want to stop hosting.
           :param model_version:  The version of the model that you want to stop
   hosting.
   ```

```
        """
        try:
            logger.info("Stopping model version %s for %s", model_version,
project_name)
            response = lookoutvision_client.stop_model(
                ProjectName=project_name, ModelVersion=model_version
            )
            logger.info("Stopping hosting...")

            status = response["Status"]
            finished = False

            # Wait until stopped or failed.
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name, ModelVersion=model_version
                )
                status = model_description["ModelDescription"]["Status"]

                if status == "STOPPING_HOSTING":
                    logger.info("Host stopping in progress...")
                    time.sleep(10)
                    continue

                if status == "TRAINED":
                    logger.info("Model is no longer hosted.")
                    finished = True
                    continue

                logger.info("Failed to stop model: %s ", status)
                finished = True

            if status != "TRAINED":
                logger.error("Error stopping model: %s", status)
                raise Exception(f"Error stopping model: {status}")
        except ClientError:
            logger.exception("Couldn't stop hosting model.")
            raise
```

Java V2

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

```
/**
 * Stops the hosting an Amazon Lookout for Vision model. Returns when model has
 * stopped or if hosting fails.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that contains the model that you
 *                    want to stop hosting.
 * @modelVersion The version of the model that you want to stop hosting.
 * @return ModelDescription The description of the model, which includes the
 *         model hosting status.
 */

public static ModelDescription stopModel(LookoutVisionClient lfvClient, String
 projectName,
                String modelVersion) throws LookoutVisionException,
 InterruptedException {

        logger.log(Level.INFO, "Stopping Model version {0} for project {1}.",
                        new Object[] { modelVersion, projectName });

        StopModelRequest stopModelRequest = StopModelRequest.builder()
                        .projectName(projectName)
                        .modelVersion(modelVersion)
                        .build();

        // Stop hosting the model.

        lfvClient.stopModel(stopModelRequest);

        DescribeModelRequest describeModelRequest =
 DescribeModelRequest.builder()
                        .projectName(projectName)
                        .modelVersion(modelVersion)
                        .build();

        ModelDescription modelDescription = null;

        boolean finished = false;
```

```
        // Wait until model is stopped or failure occurs.
        do {

                modelDescription =
lfvClient.describeModel(describeModelRequest).modelDescription();

                switch (modelDescription.status()) {

                        case TRAINED:
                                logger.log(Level.INFO, "Model version {0} for
project {1} has stopped.",
                                                new Object[] { modelVersion,
projectName });
                                finished = true;
                                break;

                        case STOPPING_HOSTING:
                                logger.log(Level.INFO, "Model version {0} for
project {1} is stopping.",
                                                new Object[] { modelVersion,
projectName });

                                TimeUnit.SECONDS.sleep(60);

                                break;

                        default:
                                logger.log(Level.SEVERE,
                                                "Unexpected error when stopping
model version {0} for project {1}: {2}.",
                                                new Object[] { projectName,
modelVersion,

modelDescription.status() });
                                finished = true;
                                break;

                }

        } while (!finished);

        logger.log(Level.INFO, "Finished stopping model version {0} for project
{1} status: {2}",
```

```
                        new Object[] { modelVersion, projectName,
 modelDescription.statusMessage() });

        return modelDescription;

}
```

Détecter des anomalies dans une image

Pour détecter des anomalies dans une image à l'aide d'un modèle Amazon Lookout for Vision entraîné, vous devez lancer DetectAnomaliesl'opération. Le résultat de `DetectAnomalies` inclut une prédiction booléenne qui classe l'image comme contenant une ou plusieurs anomalies et une valeur de confiance pour la prédiction. Si le modèle est un modèle de segmentation d'image, le résultat inclut également un masque coloré indiquant les positions des différents types d'anomalies.

Les images que vous fournissez `DetectAnomalies` doivent avoir les mêmes dimensions de largeur et de hauteur que les images que vous avez utilisées pour entraîner le modèle.

`DetectAnomalies`accepte les images au format PNG ou JPG. Nous recommandons que les images soient au même format de codage et de compression que ceux utilisés pour entraîner le modèle. Par exemple, si vous entraînez le modèle avec des images au format PNG, appelez `DetectAnomalies` avec des images au format PNG.

Avant d'appeler`DetectAnomalies`, vous devez d'abord démarrer l'`StartModel`opération sur votre modèle. Pour de plus amples informations, veuillez consulter Démarrage de votre modèle Amazon Lookout for Vision. Vous êtes facturé en fonction de la durée, en minutes, d'exécution d'un modèle et du nombre d'unités de détection d'anomalies utilisées par votre modèle. Si vous n'utilisez aucun modèle, utilisez l'`StopModel`opération pour arrêter votre modèle. Pour de plus amples informations, veuillez consulter Arrêter votre modèle Amazon Lookout for Vision.

Rubriques

- Appel DetectAnomalies
- Comprendre la réponse de DetectAnomalies
- Déterminer si une image est anormale
- Affichage des informations de classification et de segmentation
- Trouver des anomalies avec une AWS Lambda fonction

Appel DetectAnomalies

Pour appeler`DetectAnomalies`, spécifiez les éléments suivants :

- Projet : nom du projet qui contient le modèle que vous souhaitez utiliser.
- ModelVersion— La version du modèle que vous souhaitez utiliser.

- ContentType— Le type d'image que vous souhaitez analyser. Les valeurs valides sont `image/png` (images au format PNG) et `image/jpeg` (images au format JPG).
- Body — Les octets binaires non codés qui représentent l'image.

  L'image doit avoir les mêmes dimensions que les images utilisées pour entraîner le modèle.

L'exemple suivant montre comment appeler`DetectAnomalies`. Vous pouvez utiliser la réponse de fonction des exemples Python et Java pour appeler des fonctionsDéterminer si une image est anormale.

AWS CLI

Cette AWS CLI commande affiche la sortie JSON pour l'opération `DetectAnomalies` CLI. Modifiez les valeurs des paramètres d'entrée suivants :

- `project name`avec le nom du projet que vous souhaitez utiliser.
- `model version`avec la version du modèle que vous souhaitez utiliser.
- `content type`avec le type d'image que vous souhaitez utiliser. Les valeurs valides sont `image/png` (images au format PNG) et `image/jpeg` (images au format JPG).
- `file name`avec le chemin et le nom de fichier de l'image que vous souhaitez utiliser. Assurez-vous que le type de fichier correspond à la valeur de`content-type`.

```
aws lookoutvision detect-anomalies  --project-name project name\
    --model-version model version\
    --content-type content type\
    --body file name \
    --profile lookoutvision-access
```

Python

Pour l'exemple de code complet, voir GitHub.

```
    def detect_anomalies(lookoutvision_client, project_name, model_version, photo):
        """
        Calls DetectAnomalies using the supplied project, model version, and image.
        :param lookoutvision_client: A Lookout for Vision Boto3 client.
        :param project: The project that contains the model that you want to use.
        :param model_version: The version of the model that you want to use.
```

```
        :param photo: The photo that you want to analyze.
        :return: The DetectAnomalyResult object that contains the analysis results.
        """

        image_type = imghdr.what(photo)
        if image_type == "jpeg":
            content_type = "image/jpeg"
        elif image_type == "png":
            content_type = "image/png"
        else:
            logger.info("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
        with open(photo, "rb") as image:
            response = lookoutvision_client.detect_anomalies(
                ProjectName=project_name,
                ContentType=content_type,
                Body=image.read(),
                ModelVersion=model_version)

        return response['DetectAnomalyResult']
```

Java V2

```
  public static DetectAnomalyResult detectAnomalies(LookoutVisionClient lfvClient,
String projectName,
            String modelVersion,
            String photo) throws IOException, LookoutVisionException {
        /**
         * Creates an Amazon Lookout for Vision dataset from a manifest file.
         * Returns after Lookout for Vision creates the dataset.
         *
         * @param lfvClient    An Amazon Lookout for Vision client.
         * @param projectName  The name of the project in which you want to create a
         *                     dataset.
         * @param modelVersion The version of the model that you want to use.
         *
         * @param photo        The photo that you want to analyze.
         *
         * @return DetectAnomalyResult The analysis result from DetectAnomalies.
```

```
         */

        logger.log(Level.INFO, "Processing local file: {0}", photo);

        // Get image bytes.

        InputStream sourceStream = new FileInputStream(new File(photo));
        SdkBytes imageSDKBytes = SdkBytes.fromInputStream(sourceStream);
        byte[] imageBytes = imageSDKBytes.asByteArray();

        // Get the image type. Can be image/jpeg or image/png.
        String contentType = getImageType(imageBytes);

        // Detect anomalies in the supplied image.
        DetectAnomaliesRequest request =
DetectAnomaliesRequest.builder().projectName(projectName)
                .modelVersion(modelVersion).contentType(contentType).build();

        DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
                RequestBody.fromBytes(imageBytes));

        /*
         * Tip: You can also use the following to analyze a local file.
         * Path path = Paths.get(photo);
         * DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
path);
         */
        DetectAnomalyResult result = response.detectAnomalyResult();

        String prediction = "Prediction: Normal";

        if (Boolean.TRUE.equals(result.isAnomalous())) {
            prediction = "Prediction: Anomalous";
        }

        // Convert confidence to percentage.
        NumberFormat defaultFormat = NumberFormat.getPercentInstance();
        defaultFormat.setMinimumFractionDigits(1);
        String confidence = String.format("Confidence: %s",
defaultFormat.format(result.confidence()));

        // Log classification result.
        String photoPath = "File: " + photo;
        String[] imageLines = { photoPath, prediction, confidence };
```

```
        logger.log(Level.INFO, "Image: {0}\nAnomalous: {1}\nConfidence {2}",
imageLines);

        return result;

    }

    // Gets the image mime type. Supported formats are image/jpeg and image/png.
    private static String getImageType(byte[] image) throws IOException {

        InputStream is = new BufferedInputStream(new ByteArrayInputStream(image));
        String mimeType = URLConnection.guessContentTypeFromStream(is);

        logger.log(Level.INFO, "Image type: {0}", mimeType);

        if (mimeType.equals("image/jpeg") || mimeType.equals("image/png")) {
            return mimeType;
        }
        // Not a supported file type.
        logger.log(Level.SEVERE, "Unsupported image type: {0}", mimeType);
        throw new IOException(String.format("Wrong image type. %s format isn't
supported.", mimeType));
    }
```

Comprendre la réponse de DetectAnomalies

La réponse de `DetectAnomalies` varie en fonction du type de modèle que vous entraînez (modèle de classification ou modèle de segmentation). Dans les deux cas, la réponse est un DetectAnomalyResultobjet.

Modèle de classification

Si votre modèle est unModèle de classification d'images, le formulaire de réponse `DetectAnomalies` contient les éléments suivants :

- IsAnomalous— Indicateur booléen indiquant que l'image contient une ou plusieurs anomalies.
- Confiance : confiance d'Amazon Lookout for Vision quant à l'exactitude de la prédiction `IsAnomalous` des anomalies (). `Confidence`est une valeur à virgule flottante comprise entre 0 et 1. Plus la valeur est élevée, plus le niveau de confiance est élevé.
- Source — Informations sur l'image transmise à`DetectAnomalies`.

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.9996867775917053
    }
}
```

Vous déterminez si une image présente une anomalie en vérifiant le `IsAnomalous` champ et en confirmant que la `Confidence` valeur est suffisamment élevée pour répondre à vos besoins.

Si vous trouvez que les valeurs de confiance renvoyées par `DetectAnomalies` sont trop faibles, pensez à entraîner à nouveau le modèle. Pour obtenir un exemple de code, consultez Classification.

Modèle de segmentation

Si votre modèle est unModèle de segmentation d'image, la réponse inclut des informations de classification et des informations de segmentation, telles qu'un masque d'image et des types d'anomalies. Les informations de classification sont calculées séparément des informations de segmentation et vous ne devez pas supposer qu'il existe une relation entre elles. Si la réponse ne contient pas d'informations de segmentation, vérifiez que la dernière version du AWS SDK est installée (AWS Command Line Interface si vous utilisez le AWS CLI). Pour un exemple de code, voir Segmentation etAffichage des informations de classification et de segmentation.

- IsAnomalous(classification) — Indicateur booléen qui classe l'image comme normale ou anormale.
- Confiance (classification) : confiance d'Amazon Lookout for Vision quant à l'exactitude de la classification de l'image `IsAnomalous` (). `Confidence`est une valeur à virgule flottante comprise entre 0 et 1. Plus la valeur est élevée, plus le niveau de confiance est élevé.
- Source — Informations sur l'image transmise à`DetectAnomalies`.
- AnomalyMask(segmentation) — Masque de pixels couvrant les anomalies détectées dans l'image analysée. L'image peut présenter plusieurs anomalies. La couleur d'une carte de masque indique le type d'anomalie. Les couleurs du masque correspondent aux couleurs attribuées aux types d'anomalies dans le jeu de données d'apprentissage. Pour trouver le type d'anomalie à partir d'une couleur de masque, vérifiez `Color` le `PixelAnomaly` champ de chaque anomalie renvoyée dans la `Anomalies` liste. Pour obtenir un exemple de code, consultez Affichage des informations de classification et de segmentation.

- Anomalies (segmentation) : liste des anomalies détectées dans l'image. Chaque anomalie inclut le type d'anomalie (`Name`) et les informations relatives aux pixels (`PixelAnomaly`). `TotalPercentageArea`est la zone en pourcentage de l'image couverte par l'anomalie. `Color`est la couleur du masque pour l'anomalie.

  Le premier élément de la liste est toujours un type d'anomalie représentant l'arrière-plan de l'image (`BACKGROUND`) et ne doit pas être considéré comme une anomalie. Amazon Lookout for Vision ajoute automatiquement le type d'anomalie d'arrière-plan à la réponse. Il n'est pas nécessaire de déclarer un type d'anomalie d'arrière-plan dans votre ensemble de données.


```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.9996814727783203,
        "Anomalies": [
            {
                "Name": "background",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.998999834060669,
                    "Color": "#FFFFFF"
                }
            },
            {
                "Name": "scratch",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.0004034999874420464,
                    "Color": "#7ED321"
                }
            },
            {
                "Name": "dent",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.0005966666503809392,
                    "Color": "#4DD8FF"
                }
            }
        ],
        "AnomalyMask": "iVBORw0....."
```

```
    }
}
```

Déterminer si une image est anormale

Vous pouvez déterminer si une image est anormale de différentes manières. La méthode que vous choisissez dépend de votre cas d'utilisation et du type de votre modèle. Les solutions possibles sont les suivantes.

Rubriques

- Classification
- Segmentation

Classification

`IsAnomalous`classe une image comme anormale, utilise le `Confidence` champ pour déterminer si l'image est réellement anormale. Une valeur plus élevée indique une plus grande confiance. Par exemple, vous pouvez décider qu'un produit est défectueux uniquement si le niveau de confiance est supérieur à 80 %. Vous pouvez classer les images analysées par des modèles de classification ou par des modèles de segmentation d'images.

Python

Pour l'exemple de code complet, voir GitHub.

```
   def reject_on_classification(image, prediction, confidence_limit):
       """
       Returns True if the anomaly confidence is greater than or equal to
       the supplied confidence limit.
       :param image: The name of the image file that was analyzed.
       :param prediction: The DetectAnomalyResult object returned from
DetectAnomalies
       :param confidence_limit: The minimum acceptable confidence. Float value
between 0 and 1.
       :return: True if the error condition indicates an anomaly, otherwise False.
       """

       reject = False
```

```
        logger.info("Checking classification for %s", image)

        if prediction['IsAnomalous'] and prediction['Confidence'] >=
confidence_limit:
            reject = True
            reject_info=(f"Rejected: Anomaly confidence
({prediction['Confidence']:.2%}) is greater"
                    f" than limit ({confidence_limit:.2%})")
            logger.info("%s", reject_info)

        if not reject:
            logger.info("No anomalies found.")
        return reject
```

Java V2

```
   public static boolean rejectOnClassification(String image, DetectAnomalyResult
prediction, float minConfidence) {
        /**
         * Rejects an image based on its anomaly classification and prediction
         * confidence
         *
         * @param image         The file name of the analyzed image.
         * @param prediction    The prediction for an image analyzed with
         *                      DetectAnomalies.
         * @param minConfidence The minimum acceptable confidence for the prediction
         *                      (0-1).
         *
         * @return boolean True if the image is anomalous, otherwise False.
         */

        Boolean reject = false;

        logger.log(Level.INFO, "Checking classification for {0}", image);

        String[] logParameters = { prediction.confidence().toString(),
String.valueOf(minConfidence) };

        if (Boolean.TRUE.equals(prediction.isAnomalous()) && prediction.confidence()
>= minConfidence) {
            logger.log(Level.INFO, "Rejected: Anomaly confidence {0} is greater than
confidence limit {1}",
                    logParameters);
```

```
            reject = true;
        }
        if (Boolean.FALSE.equals(reject))
            logger.log(Level.INFO, ": No anomalies found.");

        return reject;

    }
```

Segmentation

Si votre modèle est un modèle de segmentation d'image, vous pouvez utiliser les informations de segmentation pour déterminer si une image contient des anomalies. Vous pouvez également utiliser un modèle de segmentation d'image pour classer les images. Par exemple du code permettant d'obtenir et d'afficher des masques d'image, voir Affichage des informations de classification et de segmentation

Zone d'anomalie

Utilisez le pourcentage de couverture (`TotalPercentageArea`) d'une anomalie sur l'image. Par exemple, vous pouvez décider qu'un produit est défectueux si la zone d'une anomalie est supérieure à 1 % de l'image.

Python

Pour l'exemple de code complet, voir GitHub.

```
   def reject_on_coverage(image, prediction, confidence_limit, anomaly_label,
coverage_limit):
        """
        Checks if the coverage area of an anomaly is greater than the coverage limit
and if
        the prediction confidence is greater than the confidence limit.
        :param image: The name of the image file that was analyzed.
        :param prediction: The DetectAnomalyResult object returned from
DetectAnomalies
        :param confidence_limit: The minimum acceptable confidence (float 0-1).
        :anomaly_label: The anomaly label for the type of anomaly that you want to
check.
        :coverage_limit: The maximum acceptable percentage coverage of an anomaly
(float 0-1).
```

```
        :return: True if the error condition indicates an anomaly, otherwise False.
        """

        reject = False

        logger.info("Checking coverage for %s", image)

        if prediction['IsAnomalous'] and prediction['Confidence'] >=
 confidence_limit:
            for anomaly in prediction['Anomalies']:
                if (anomaly['Name'] == anomaly_label and
                        anomaly['PixelAnomaly']['TotalPercentageArea'] >
 (coverage_limit)):
                    reject = True
                    reject_info=(f"Rejected: Anomaly confidence
 ({prediction['Confidence']:.2%}) "
                        f"is greater than limit ({confidence_limit:.2%}) and
 {anomaly['Name']} "
                        f"coverage ({anomaly['PixelAnomaly']
['TotalPercentageArea']:.2%}) "
                        f"is greater than limit ({coverage_limit:.2%})")

                    logger.info("%s", reject_info)

        if not reject:
            logger.info("No anomalies found.")

        return reject
```

Java V2

```
   public static Boolean rejectOnCoverage(String image, DetectAnomalyResult
 prediction, float minConfidence,
            String anomalyType, float maxCoverage) {
        /**
         * Rejects an image based on a maximum allowable coverage area for an
 anomaly
         * type.
         *
         * @param image         The file name of the analyzed image.
         * @param prediction    The prediction for an image analyzed with
         *                      DetectAnomalies.
```

```
        * @param minConfidence The minimum acceptable confidence for the prediction
        *                      (0-1).
        * @param anomalyTypes  The anomaly type to check.
        * @param maxCoverage   The maximum allowable coverage area of the anomaly
type.
        *                      (0-1).
        *
        * @return boolean True if the coverage area of the anomaly type exceeds the
        *         maximum allowed, otherwise False.
        */

       Boolean reject = false;

       logger.log(Level.INFO, "Checking coverage for {0}", image);

       if (Boolean.TRUE.equals(prediction.isAnomalous()) && prediction.confidence()
>= minConfidence) {
           for (Anomaly anomaly : prediction.anomalies()) {

               if (Objects.equals(anomaly.name(), anomalyType)
                       && anomaly.pixelAnomaly().totalPercentageArea() >=
maxCoverage) {

                   String[] logParameters = { prediction.confidence().toString(),
                           String.valueOf(minConfidence),

String.valueOf(anomaly.pixelAnomaly().totalPercentageArea()),
                           String.valueOf(maxCoverage) };
                   logger.log(Level.INFO,
                           "Rejected: Anomaly confidence {0} is greater than
confidence limit {1} and " +
                                   "{2} anomaly type coverage is higher than
coverage limit {3}\n",
                           logParameters);
                   reject = true;

               }
           }
       }

       if (Boolean.FALSE.equals(reject))
           logger.log(Level.INFO, ": No anomalies found.");

       return reject;
```

```
    }
```

Nombre de types d'anomalies

Utilisez le nombre des différents types d'anomalies (Name) présents sur l'image. Par exemple, vous pouvez décider qu'un produit est défectueux s'il existe plus de deux types d'anomalies.

Python

Pour l'exemple de code complet, voir GitHub.

```
   def reject_on_anomaly_types(image, prediction, confidence_limit,
anomaly_types_limit):
       """
       Checks if the number of anomaly types is greater than than the anomaly types
       limit and if the prediction confidence is greater than the confidence limit.
       :param image: The name of the image file that was analyzed.
       :param prediction: The DetectAnomalyResult object returned from
DetectAnomalies
       :param confidence: The minimum acceptable confidence. Float value between 0
and 1.
       :param anomaly_types_limit: The maximum number of allowable anomaly types
(Integer).
       :return: True if the error condition indicates an anomaly, otherwise False.
       """

       logger.info("Checking number of anomaly types for %s",image)

       reject = False

       if prediction['IsAnomalous'] and prediction['Confidence'] >=
confidence_limit:

           anomaly_types = {anomaly['Name'] for anomaly in prediction['Anomalies']\
               if anomaly['Name'] != 'background'}

           if len (anomaly_types) > anomaly_types_limit:
                   reject = True
                   reject_info = (f"Rejected: Anomaly confidence
({prediction['Confidence']:.2%}) "
                   f"is greater than limit ({confidence_limit:.2%}) and "
                   f"the number of anomaly types ({len(anomaly_types)-1}) is "
```

```
                    f"greater than the limit ({anomaly_types_limit})")

                    logger.info("%s", reject_info)

        if not reject:
            logger.info("No anomalies found.")
        return reject
```

Java V2

```
   public static Boolean rejectOnAnomalyTypeCount(String image, DetectAnomalyResult
prediction,
           float minConfidence, Integer maxAnomalyTypes) {

       /**
        * Rejects an image based on a maximum allowable number of anomaly types.
        *
        * @param image           The file name of the analyzed image.
        * @param prediction      The prediction for an image analyzed with
        *                        DetectAnomalies.
        * @param minConfidence   The minimum acceptable confidence for the
predictio
        *                        (0-1).
        * @param maxAnomalyTypes The maximum allowable number of anomaly types.
        *
        * @return boolean True if the image contains more than the maximum allowed
        *         anomaly types, otherwise False.
        */

       Boolean reject = false;

       logger.log(Level.INFO, "Checking coverage for {0}", image);

       Set<String> defectTypes = new HashSet<>();

       if (Boolean.TRUE.equals(prediction.isAnomalous()) && prediction.confidence()
>= minConfidence) {
           for (Anomaly anomaly : prediction.anomalies()) {
               defectTypes.add(anomaly.name());
           }
           // Reduce defect types by one to account for 'background' anomaly type.
           if ((defectTypes.size() - 1) > maxAnomalyTypes) {
               String[] logParameters = { prediction.confidence().toString(),
```

```
                        String.valueOf(minConfidence),
                        String.valueOf(defectTypes.size()),
                        String.valueOf(maxAnomalyTypes) };
                logger.log(Level.INFO, "Rejected: Anomaly confidence {0} is >=
minimum confidence {1} and " +
                        "the number of anomaly types {2} > the allowable number of
anomaly types {3}\n", logParameters);
                reject = true;
            }

        }

        if (Boolean.FALSE.equals(reject))
            logger.log(Level.INFO, ": No anomalies found.");

        return reject;
    }
```

Affichage des informations de classification et de segmentation

Cet exemple montre l'image analysée et superpose les résultats de l'analyse. Si la réponse inclut un masque d'anomalie, le masque est affiché dans les couleurs des types d'anomalies associés.

Pour afficher les informations de classification et de segmentation des images

1. Si vous ne l'avez pas déjà fait, effectuez les opérations suivantes :

   a. 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.

   b. Entraînez votre modèle.

   c. Démarrez votre modèle.

2. Assurez-vous que l'utilisateur qui appelle `DetectAnomalies` a accès à la version du modèle que vous souhaitez utiliser. Pour de plus amples informations, veuillez consulter Configuration des autorisations du kit SDK.

3. Utilisez le code suivant.

Python

L'exemple de code suivant détecte des anomalies dans une image que vous fournissez. L'exemple utilise les options de ligne de commande suivantes :

- `project`— le nom du projet que vous souhaitez utiliser.
- `version`— la version du modèle, dans le projet, que vous souhaitez utiliser.
- `image`— le chemin et le fichier d'un fichier image local (format JPEG ou PNG).

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Shows how to detect and show anomalies in an image using a trained Amazon
 Lookout
for Vision model. The script displays the analysed image and overlays mask and
 analysis
output.
"""

import argparse
import logging
import io
import boto3
from PIL import Image, ImageDraw, ImageFont

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class ShowAnomalies:
    """
    Class to detect and show anomalies in an image analyzed by detect_anomalies.
    """



    @staticmethod
    def draw_line(draw, text, fnt,  y_coordinate, color):
        """
        Draws a line of text on the supplied drawing surface.
        :param draw: The surface on which to draw the text.
```

```
        :param text: The text to draw in the drawing surface.
        :param fnt: The font for the text.
        :param y_coordinate: The y position for the text.
        :param color: The color for the text.
        :returns The y coordinate for the next line of text.
        """
        text_width, text_height = draw.textsize(text, fnt)
        draw.rectangle([(10, y_coordinate), (text_width + 10,
                                             y_coordinate + text_height)],
 fill="black")
        draw.text((10, y_coordinate), text, fill=color, font=fnt)

        y_coordinate += text_height

        return y_coordinate

    @staticmethod
    def draw_analysis_text(image, analysis):
        """
        Draws classification and segmentation info onto supplied image
        overlay analysis results on an image analyzed by detect_anomalies.
        :param analysis: The response from a call to detect_anomalies.
        :returns Nothing
        """

        ## Calculate a reasonable font size based on image width.
        font_size = int(image.size[0]/32)

        fnt = ImageFont.truetype('/Library/Fonts/Tahoma.ttf', font_size)

        draw = ImageDraw.Draw(image)

        y_coordinate = 0

        # Draw classification information.
        prediction = "Anomalous" if analysis["DetectAnomalyResult"]
["IsAnomalous"] \
            else "Normal"

        confidence = analysis["DetectAnomalyResult"]["Confidence"]
        found_anomalies = analysis["DetectAnomalyResult"]['Anomalies']
        segmentation_info = False

        logger.info("Prediction: %s", format(prediction))
```

```
        logger.info("Confidence: %s", format(confidence))

        y_coordinate = 0
        y_coordinate = ShowAnomalies.draw_line(
            draw, "Classification", fnt, y_coordinate, "white")
        y_coordinate = ShowAnomalies.draw_line(
            draw, f"  Prediction: {prediction}", fnt, y_coordinate, "white")
        y_coordinate = ShowAnomalies.draw_line(
            draw, f"  Confidence: {confidence:.2%}", fnt, y_coordinate, "white")

        # Draw segmentation information, if present.
        if (len(found_anomalies)) > 1:
            logger.info("Anomalies:")

            y_coordinate = ShowAnomalies.draw_line(
                draw, "Segmentation:", fnt, y_coordinate, "white")
            for i in range(1, len(found_anomalies)):

                # Only display info if more than 0% coverage found.
                percent_coverage = found_anomalies[i]['PixelAnomaly']
['TotalPercentageArea']
                if percent_coverage > 0:
                    segmentation_info = True
                    logger.info("  %s", found_anomalies[i]['Name'])
                    logger.info("    Color: %s",
                                found_anomalies[i]['PixelAnomaly']['Color'])
                    logger.info("    Area: %s", percent_coverage)
                    y_coordinate = ShowAnomalies.draw_line(
                        draw,
                        f"  Anomaly: {found_anomalies[i]['Name']}. Area:
 {percent_coverage:.2%}",
                        fnt,
                        y_coordinate,
                        found_anomalies[i]['PixelAnomaly']['Color'])

            if not segmentation_info:
                y_coordinate = ShowAnomalies.draw_line(
                    draw, "No segmentation information found.", fnt,
 y_coordinate, "white")




    @staticmethod
```

```
   def show_anomaly_prediction(lookoutvision_client, project_name,
model_version, photo):
       """
       Detects anomalies in an image (jpg/png) by using your Amazon Lookout for
Vision
       model. Displays the image and overlays prediction information text.
       :param lookoutvision_client: An Amazon Lookout for Vision Boto3 client.
       :param project_name: The name of the project that contains the model
that
       you want to use.
       :param model_version: The version of the model that you want to use.
       :param photo: The path and name of the image in which you want to detect
       anomalies.
       """
       try:

           logger.info("Detecting anomalies in %s", photo)

           image = Image.open(photo)
           image_type = Image.MIME[image.format]

           # Check that image type is valid.
           if image_type not in ("image/jpeg", "image/png"):
               logger.info("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()

           # Analyze the image.
           response = lookoutvision_client.detect_anomalies(
               ProjectName=project_name,
               ContentType=image_type,
               Body=image_bytes,
               ModelVersion=model_version
           )

           # Overlay mask onto analyzed image.
           image_mask_bytes = response["DetectAnomalyResult"]["AnomalyMask"]
```

```
            image_mask = Image.open(io.BytesIO(image_mask_bytes))

            final_img = Image.blend(image, image_mask, 0.5) \
                if response["DetectAnomalyResult"]["IsAnomalous"] else image

            # Overlay analysis output on image.
            ShowAnomalies.draw_analysis_text(final_img, response)

            final_img.show()

        except ClientError as err:
            logger.info(format(err))
            raise


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

    parser.add_argument(
        "project", help="The project containing the model that you want to use."
    )
    parser.add_argument(
        "version", help="The version of the model that you want to use."
    )
    parser.add_argument(
        "image",
        help="The file that you want to analyze. "
        "Supply a local file path.",
    )


def main():
    """
    Entrypoint for anomaly detection example.
    """

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

        session = boto3.Session(
```

```
            profile_name='lookoutvision-access')

        lookoutvision_client = session.client("lookoutvision")

        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)

        add_arguments(parser)

        args = parser.parse_args()

        # Analyze the image and show results.
        ShowAnomalies.show_anomaly_prediction(
            lookoutvision_client, args.project, args.version, args.image
        )

    except ClientError as err:
        print("A service error occured: " +
              format(err.response["Error"]["Message"]))
    except FileNotFoundError as err:
        print("The supplied file couldn't be found: " + err.filename)
    except ValueError as err:
        print("A value error occured. " + format(err))
    else:
        print("Successfully completed analysis.")


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

Java 2

L'exemple de code suivant détecte des anomalies dans une image que vous fournissez. L'exemple utilise les options de ligne de commande suivantes :

- `project`— le nom du projet que vous souhaitez utiliser.
- `version`— la version du modèle, dans le projet, que vous souhaitez utiliser.
- `image`— le chemin et le fichier d'un fichier image local (format JPEG ou PNG).

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

```
*/


package com.example.lookoutvision;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.lookoutvision.LookoutVisionClient;
import software.amazon.awssdk.services.lookoutvision.model.Anomaly;
import
 software.amazon.awssdk.services.lookoutvision.model.DetectAnomaliesRequest;
import
 software.amazon.awssdk.services.lookoutvision.model.DetectAnomaliesResponse;
import software.amazon.awssdk.services.lookoutvision.model.DetectAnomalyResult;
import
 software.amazon.awssdk.services.lookoutvision.model.LookoutVisionException;

import java.io.BufferedInputStream;
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.net.URLConnection;

import java.text.NumberFormat;
import java.awt.*;
import java.awt.font.LineMetrics;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import javax.swing.*;

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

// Finds anomalies on a supplied image.
public class ShowAnomalies extends JPanel {
/**
 * Finds and displays anomalies on a supplied image.
 */

    private static final long serialVersionUID = 1L;
```

```
   private transient BufferedImage image;
   private transient BufferedImage maskImage;
   private transient Dimension dimension;
   public static final Logger logger =
Logger.getLogger(ShowAnomalies.class.getName());

   // Constructor. Finds anomalies in a local image file.
   public ShowAnomalies(LookoutVisionClient lfvClient, String projectName,
String modelVersion,
           String photo) throws IOException, LookoutVisionException {

       logger.log(Level.INFO, "Processing local file: {0}", photo);

       maskImage = null;

       // Get image bytes and buffered image.
       InputStream sourceStream = new FileInputStream(new File(photo));
       SdkBytes imageSDKBytes = SdkBytes.fromInputStream(sourceStream);
       byte[] imageBytes = imageSDKBytes.asByteArray();
       ByteArrayInputStream inputStream = new
ByteArrayInputStream(imageSDKBytes.asByteArray());
       image = ImageIO.read(inputStream);

       // Get the image type. Can be image/jpeg or image/png.
       String contentType = getImageType(imageBytes);

       // Set the size of the window that shows the image.
       setWindowDimensions();

       // Detect anomalies in the supplied image.
       DetectAnomaliesRequest request =
DetectAnomaliesRequest.builder().projectName(projectName)
               .modelVersion(modelVersion).contentType(contentType).build();

       DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
               RequestBody.fromBytes(imageBytes));

       /*
        * Tip: You can also use the following to analyze a local file.
        * Path path = Paths.get(photo);
        * DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
path);
        */
       DetectAnomalyResult result = response.detectAnomalyResult();
```

```
        if (result.anomalyMask() != null){
            SdkBytes maskSDKBytes = result.anomalyMask();

            ByteArrayInputStream maskInputStream = new
ByteArrayInputStream(maskSDKBytes.asByteArray());
            maskImage = ImageIO.read(maskInputStream);
        }

        drawImageInfo(result);

    }

    // 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;
        dimension.height = (int) dimension.getHeight() / 2;

        if (image.getWidth() < dimension.width || image.getHeight() <
dimension.height) {
            dimension.width = image.getWidth();
            dimension.height = image.getHeight();
        }
        setPreferredSize(dimension);

    }

    private int drawLine(Graphics2D g2d, String line, FontMetrics metrics, int
yPos, Color color) {
    /**
    * Draws a line of text at the spsecified y position and color.
    * confidence
    *
    * @param g2D The Graphics2D object for the image.
    * @param line The line of text to draw.
    * @param metrics The font information to use.
    * @param yPos The y position for the line of text.
    *
    * @return  The yPos for the next line of text.
    */
```

```
        int indent = 10;

        // Get text height, width, and descent.
        int textWidth = metrics.stringWidth(line);
        LineMetrics lm = metrics.getLineMetrics(line, g2d);
        int textHeight = (int) lm.getHeight();
        int descent = (int) lm.getDescent();

        int y2Pos = (yPos + textHeight) - descent;

        // Draw black rectangle.
        g2d.setColor(Color.BLACK);
        g2d.fillRect(indent, yPos, textWidth, textHeight);

        // Draw text.
        g2d.setColor(color);
        g2d.drawString(line, indent, y2Pos);

        yPos += textHeight;

        return yPos;

    }

    public void drawImageInfo(DetectAnomalyResult result) {
    /**
     * Draws the results from DetectAnomalies onto the output image.
     *
     * @param result The response from a call to
     *               DetectAnomalies.
     *
     */

        // Set up drawing.
        Graphics2D g2d = image.createGraphics();

        if (result.anomalyMask() != null){
            Composite composite = g2d.getComposite();
            g2d.setComposite(AlphaComposite.SrcOver.derive(0.5f));
            int x = (image.getWidth() - maskImage.getWidth()) / 2;
            int y = (image.getHeight() - maskImage.getHeight()) / 2;
            g2d.drawImage(maskImage, x, y, null);
            // Set composite for overlaying text.
            g2d.setComposite(composite);
```

```
        }

        //Calculate font size based on arbitary 32 pixel image width.
        int fontSize = (image.getWidth() / 32);


        g2d.setFont(new Font("Tahoma", Font.PLAIN, fontSize));
        Font font = g2d.getFont();
        FontMetrics metrics = g2d.getFontMetrics(font);

        // Get classification information.

        String prediction = "Prediction: Normal";

        if (Boolean.TRUE.equals(result.isAnomalous())) {
            prediction = "Prediction: Anomalous";
        }

        // Convert prediction to percentage.
        NumberFormat defaultFormat = NumberFormat.getPercentInstance();
        defaultFormat.setMinimumFractionDigits(1);
        String confidence = String.format("Confidence: %s",
defaultFormat.format(result.confidence()));

        // Draw classification information.
        int yPos = 0;

        yPos = drawLine(g2d, "Classification:", metrics, yPos, Color.WHITE);
        yPos = drawLine(g2d, prediction, metrics, yPos, Color.WHITE);
        yPos = drawLine(g2d, confidence, metrics, yPos, Color.WHITE);

        // Draw segmentation info.
        yPos = drawLine(g2d, "Segmentation:", metrics, yPos, Color.WHITE);

        // Ignore background label, so size must be > 1
        if (result.anomalies().size() > 1) {
            for (Anomaly anomaly : result.anomalies()) {
                if (anomaly.name().equals("background"))
                    continue;
                String label = String.format("Anomaly: %s. Area: %s",
anomaly.name(),

defaultFormat.format(anomaly.pixelAnomaly().totalPercentageArea()));
```

```
                Color anomalyColor =
Color.decode((anomaly.pixelAnomaly().color()));
                yPos = drawLine(g2d, label, metrics, yPos, anomalyColor);

            }

        } else {
            drawLine(g2d, "None found.", metrics, yPos, Color.WHITE);

        }

        g2d.dispose();

    }

    @Override
    public void paintComponent(Graphics g)
    /**
     * Draws the image and analysis results.
     *
     * @param g The Graphics context object for drawing.
     *          DetectAnomalies.
     *
     */
    {

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

        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

    // Gets the image mime type. Supported formats are image/jpeg and image/png.

    private String getImageType(byte[] image) throws IOException
    /**
     * Gets the file type of a supplied image. Raises an exception if the image
     * isn't compatible with with Amazon Lookout for Vision.
     *
     * @param image The image that you want to check.
     *
     * @return String The type of the image.
     */
```

```
   {
       InputStream is = new BufferedInputStream(new
ByteArrayInputStream(image));
       String mimeType = URLConnection.guessContentTypeFromStream(is);

       logger.log(Level.INFO, "Image type: {0}", mimeType);

       if (mimeType.equals("image/jpeg") || mimeType.equals("image/png")) {
           return mimeType;
       }
       // Not a supported file type.
       logger.log(Level.SEVERE, "Unsupported image type: {0}", mimeType);
       throw new IOException(String.format("Wrong image type. %s format isn't
supported.", mimeType));
   }

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

       String photo = null;
       String projectName = null;
       String modelVersion = null;

       final String USAGE = "\n" +
               "Usage:\n" +
               "    DetectAnomalies <project> <version> <image> \n\n" +
               "Where:\n" +
               "    project - The Lookout for Vision project.\n\n" +
               "    version - The version of the model within the project.\n\n"
+
               "    image - The path and filename of a local image. \n\n";

       try {

           if (args.length != 3) {
               System.out.println(USAGE);
               System.exit(1);
           }

           projectName = args[0];
           modelVersion = args[1];
           photo = args[2];
           ShowAnomalies panel = null;
```

```
            // Get the Lookout for Vision client.
            LookoutVisionClient lfvClient = LookoutVisionClient.builder()

 .credentialsProvider(ProfileCredentialsProvider.create("lookoutvision-access"))
                .build();

            // Create frame and panel.
            JFrame frame = new JFrame(photo);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            panel = new ShowAnomalies(lfvClient, projectName, modelVersion,
 photo);

            frame.setContentPane(panel);
            frame.pack();
            frame.setVisible(true);


        } catch (LookoutVisionException lfvError) {
            logger.log(Level.SEVERE, "Lookout for Vision client error: {0}:
 {1}",
                    new Object[] { lfvError.awsErrorDetails().errorCode(),
                            lfvError.awsErrorDetails().errorMessage() });
            System.out.println(String.format("lookout for vision client error:
 %s", lfvError.getMessage()));
            System.exit(1);

        } catch (FileNotFoundException fileError) {
            logger.log(Level.SEVERE, "Could not find file: {0}",
 fileError.getMessage());
            System.out.println(String.format("Could not find file: %s",
 fileError.getMessage()));
            System.exit(1);

        } catch (IOException ioError) {
            logger.log(Level.SEVERE, "IO error {0}", ioError.getMessage());
            System.out.println(String.format("IO error: %s",
 ioError.getMessage()));
            System.exit(1);
        }

    }
}
```

4. Si vous n'avez pas l'intention de continuer à utiliser votre modèle, arrêtez-le.

Trouver des anomalies avec 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 DetectAnomalies. La fonction analyse une image fournie et renvoie une classification pour détecter la présence d'anomalies dans cette 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)
- Étape 2 : (Facultatif) Créer une couche (console)
- Étape 3 : Ajouter le code Python (console)
- Étape 4 : Essayer votre fonction Lambda

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'`DetectAnomalies`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 projet Amazon Lookout for Vision et la version du modèle 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
NOM_PROJET	Le projet Lookout for Vision qui contient le modèle que vous souhaitez utiliser.
MODÈLE_VERSION	Version du modèle que vous souhaitez utiliser.
CONFIDENCE	Valeur minimale (0 à 100) indiquant que le modèle est certain que la prédiction est anormale. Si le niveau de confiance est inférieur, le classement est considéré comme normal.

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": "lookoutvision:DetectAnomalies",
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "DetectAnomaliesAccess"
        }
    ]
}
```

12. Choisissez Suivant.
13. Dans Détails de la politique, entrez le nom de la politique, tel que DetectAnomalies-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'`DetectAnomalies`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 `DetectAnomalies` et renvoie une classification (vrai si l'image est anormale, faux si l'image est normale). 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 Amazon Lookout for Vision
 model.
"""
import base64
import imghdr
from os import environ
from io import BytesIO
import logging
import boto3
```

```
from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


# Get the model and confidence.
project_name = environ['PROJECT_NAME']
model_version = environ['MODEL_VERSION']
min_confidence = int(environ.get('CONFIDENCE', 50))

lookoutvision_client = boto3.client('lookoutvision')


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:

        file_name = ""

        # Determine image source.
        if 'image' in event:
            # Decode the encoded image
            image_bytes = event['image'].encode('utf-8')
            img_b64decoded = base64.b64decode(image_bytes)
            image_type = get_image_type(img_b64decoded)
            image = BytesIO(img_b64decoded)
            file_name = event['filename']

        elif 'S3Object' in event:
            bucket = boto3.resource('s3').Bucket(event['S3Object']['Bucket'])
            image_object = bucket.Object(event['S3Object']['Name'])
            image = image_object.get().get('Body').read()
            image_type = get_image_type(image)
            file_name = f"s3://{event['S3Object']['Bucket']}/{event['S3Object']
['Name']}"
```

```
        else:
            raise ValueError(
                'Invalid image source. Only base 64 encoded image bytes or images
in S3 buckets are supported.')

        # Analyze the image.
        response = lookoutvision_client.detect_anomalies(
            ProjectName=project_name,
            ContentType=image_type,
            Body=image,
            ModelVersion=model_version)

        reject = reject_on_classification(
            response['DetectAnomalyResult'],
confidence_limit=float(environ['CONFIDENCE'])/100)

        status = "anomalous" if reject else "normal"

        lambda_response = {
            "statusCode": 200,
            "body": {
                "Reject": reject,
                "RejectMessage": f"Image {file_name} is {status}."
            }
        }

    except ClientError as err:
        error_message = f"Couldn't analyze {file_name}. " + \
            err.response['Error']['Message']

        lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": err.response['Error']['Code'],
                "ErrorMessage": error_message,
                "Image": file_name
            }
        }
        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),
                "Image": event['filename']
            }
        }
        logger.error("Error function %s: %s",
                     context.invoked_function_arn, format(val_error))

    return lambda_response

def get_image_type(image):
    """
    Gets the format of the image. Raises an error
    if the type is not PNG or JPEG.
    :param image: The image that you want to check.
    :return The type of the image.

    """
    image_type = imghdr.what(None, image)

    if image_type == "jpeg":
        content_type = "image/jpeg"
    elif image_type == "png":
        content_type = "image/png"
    else:
        logger.info("Invalid image type")
        raise ValueError(
            "Invalid file format. Supply a jpeg or png format file.")
    return content_type


def reject_on_classification(prediction, confidence_limit):
    """
    Returns True if the anomaly confidence is greater than or equal to
    the supplied confidence limit.
    :param image: The name of the image file that was analyzed.
    :param prediction: The DetectAnomalyResult object returned from DetectAnomalies
    :param confidence_limit: The minimum acceptable confidence. Float value between
 0 and 1.
    :return: True if the error condition indicates an anomaly, otherwise False.
    """
```

```
    reject = False

    if prediction['IsAnomalous'] and prediction['Confidence'] >= confidence_limit:
        reject = True
        reject_info = (f"Rejected: Anomaly confidence
({prediction['Confidence']:.2%}) is greater"
                       f" than limit ({confidence_limit:.2%})")
        logger.info("%s", reject_info)

    if not reject:
        logger.info("No anomalies found.")
    return reject
```

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. Si vous ne l'avez pas déjà fait, effectuez les opérations suivantes :

a. Assurez-vous que l'utilisateur qui utilise le code client dispose de `lambda:InvokeFunction` l'autorisation requise. Vous pouvez utiliser les autorisations suivantes.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LambdaPermission",
            "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.

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

c. Démarrez le modèle que vous avez spécifié à l'étape 7 de l'Étape 1 : Création d'une AWS Lambda fonction (console).

2. 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: Shows how to call the anomaly detection
AWS Lambda function.
"""
from botocore.exceptions import ClientError

import argparse
import logging
import base64
import json
import boto3
from os import environ

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, "filename": image}

    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()

        # Analyze image and display results.

        result = analyze_image(args.function, args.image)

        status = result['statusCode']
```

```
        if status == 200:
            classification = result['body']
            print(f"classification: {classification['Reject']}")
            print(f"Message: {classification['RejectMessage']}")
        else:
            print(f"Error: {result['statusCode']}")
            print(f"Message: {result['body']}")

    except ClientError as error:
        logging.error(error)
        print(error)


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

3. Exécutez le code. Pour l'argument de ligne de commande, indiquez le nom de la fonction Lambda et le chemin d'accès à l'image locale que vous souhaitez analyser. Par exemple :

```
python client.py function_name /bucket/path/image.jpg
```

   En cas de succès, le résultat est une classification des anomalies détectées dans l'image. Si aucune classification n'est renvoyée, pensez à réduire la valeur de confiance que vous avez définie à l'étape 7 deÉtape 1 : Création d'une AWS Lambda fonction (console).

4. 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.

Utilisation de votre modèle Amazon Lookout for Vision sur un appareil périphérique

Vous pouvez utiliser votre modèle Amazon Lookout for Vision sur des appareils périphériques gérés par AWS IoT Greengrass Version 2. AWS IoT Greengrass est un environnement d'exécution périphérique et un service cloud open source pour l'Internet des objets (IoT). Vous pouvez l'utiliser pour créer, déployer et gérer des applications IoT sur vos appareils. Pour de plus amples informations, veuillez consulter AWS IoT Greengrass.

Vous déployez les mêmes modèles Amazon Lookout for Vision que ceux que vous avez entraînés dans le cloud AWS IoT Greengrass V2 sur des appareils périphériques compatibles. Vous pouvez ensuite utiliser votre modèle déployé pour détecter les anomalies sur site, par exemple dans une usine, sans diffuser continuellement les données vers le cloud. Vous pouvez ainsi minimiser les coûts de bande passante et détecter les anomalies localement grâce à une analyse d'image en temps réel.

> **Tip**
>
> Avant de déployer un modèle Lookout for Vision AWS IoT Greengrass avec, nous vous recommandons de lire AWS IoT Greengrass Version 2 le guide du développeur. Pour plus d'informations, consultez Qu'est-ce qu'AWS IoT Greengrass ? .

Pour utiliser un modèle Lookout for Vision sur AWS IoT Greengrass V2 un appareil principal, vous déployez le modèle et le logiciel de support en tant que composants sur le périphérique principal. Un composant est un module logiciel, tel qu'un modèle Lookout for Vision, qui s'exécute sur un appareil principal de Greengrass. Il existe deux formes de composants. Un composant personnalisé est un composant que vous créez et auquel vous seul pouvez accéder. Il est également connu sous le nom de composant privé. Un composant AWS fourni est un composant prédéfini qui AWS fournit. Il est également connu sous le nom de composant public. Pour de plus amples informations, veuillez consulter https://docs.aws.amazon.com/greengrass/v2/developerguide/public-components.html.

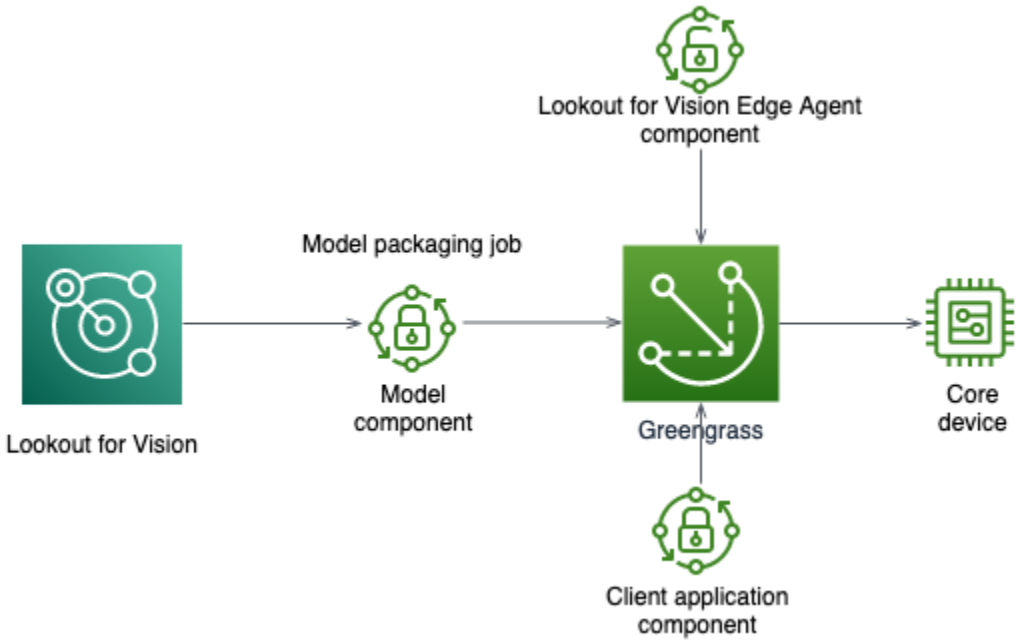
Les composants que vous déployez sur un appareil principal pour un modèle Lookout for Vision et les logiciels associés sont les suivants :

- Composant du modèle. Composant personnalisé qui contient votre modèle Lookout for Vision. Pour créer le composant du modèle, vous utilisez Lookout for Vision pour créer une tâche

d'empaquetage du modèle. Une tâche d'empaquetage de modèles crée un composant pour le modèle et le rend disponible en tant que composant personnalisé au sein de celui-ci AWS IoT Greengrass V2. Pour de plus amples informations, veuillez consulter Emballage de votre modèle Amazon Lookout for Vision.

- Composant d'application client. Composant personnalisé que vous créez et qui implémente le code correspondant aux besoins de votre entreprise. Par exemple, trouver des circuits imprimés anormaux à partir d'images prises après l'assemblage. Pour de plus amples informations, veuillez consulter Écrire le composant de votre application client.
- Composant Amazon Lookout for Vision Edge Agent. Composant AWS fourni qui fournit une API pour l'utilisation et la gestion de votre modèle. Par exemple, le code du composant de votre application cliente peut utiliser l'`DetectAnomalies`API pour détecter des anomalies dans les images. Le composant Lookout for Vision Edge Agent est une dépendance du composant du modèle. Il est automatiquement installé sur le périphérique principal lorsque vous déployez le composant du modèle. Pour de plus amples informations, veuillez consulter Référence de l'API Amazon Lookout for Vision Edge Agent.

Après avoir créé le composant modèle et le composant d'application client, vous pouvez les utiliser AWS IoT Greengrass V2 pour déployer les composants et les dépendances sur le périphérique principal. Pour de plus amples informations, veuillez consulter Déploiement de vos composants sur un appareil.



> **Important**
>
> Les prédictions effectuées par votre modèle `DetectAnomalies` sur un appareil principal peuvent être différentes de celles effectuées à l'aide du même modèle hébergé dans le cloud. Nous vous recommandons de tester votre modèle sur un appareil principal avant de l'utiliser dans un environnement de production.
> Pour réduire les incohérences de prédiction entre les modèles hébergés sur l'appareil et les modèles hébergés dans le cloud, nous vous recommandons d'augmenter le nombre d'images normales et anormales dans votre ensemble de données d'entraînement. Nous ne recommandons pas de réutiliser les images existantes pour augmenter la taille du jeu de données d'entraînement.

Déploiement d'un modèle et d'un composant d'application client sur un périphérique AWS IoT Greengrass Version 2 principal

La procédure de déploiement d'un modèle Amazon Lookout for Vision et d'un composant d'application client sur AWS IoT Greengrass Version 2 un appareil principal est la suivante :

1. Configurez vos principaux appareils avec AWS IoT Greengrass Version 2.
2. Créez une tâche d'empaquetage modèle à l'aide de Lookout for Vision. La tâche crée le composant de votre modèle.
3. Écrivez un composant d'application client. Le composant met en œuvre votre logique métier.
4. Déployez le composant du modèle et le composant de l'application client sur le périphérique principal à l'aide de AWS IoT Greengrass V2.

Une fois les composants et les dépendances déployés sur le périphérique principal, vous pouvez utiliser le modèle sur le périphérique principal.

> **Note**
>
> Vous devez utiliser la même AWS région et le même AWS compte pour créer et déployer votre modèle Lookout for Vision et le même composant d'application client.

AWS IoT Greengrass Version 2 exigences de base en matière d'appareils

Pour utiliser un modèle Amazon Lookout for Vision sur AWS IoT Greengrass Version 2 un appareil principal, votre modèle doit satisfaire à différentes exigences relatives à l'appareil principal.

Rubriques

- Appareils, architectures de puces et systèmes d'exploitation testés
- Mémoire et stockage de l'appareil principal
- Logiciel requis

Appareils, architectures de puces et systèmes d'exploitation testés

Nous nous attendons à ce qu'Amazon Lookout for Vision fonctionne sur le matériel suivant :

- Architectures de processeurs
  - X86_64 (version 64 bits du jeu d'instructions x86)
  - Aarch64 (ARMv8 processeur 64 bits)
- (Inférence accélérée par GPU uniquement) Accélérateur GPU NVIDIA avec une capacité de mémoire suffisante (au moins 6 Go pour un modèle en cours d'exécution).

L'équipe Amazon Lookout for Vision a testé les modèles Lookout for Vision sur les appareils, architectures de puces et systèmes d'exploitation suivants.

Appareils

Appareil	Système d'exploitation	Architecture	Accélérateur	Options du compilateur
jetson_xavier (NVIDIA® Jetson AGX Xavier)	Linux	Aarch64	NVIDIA	`{"gpu-code": "sm_72", "trt-ver": "7.1.3",`

Appareil	Système d'exploitation	Architecture	Accélérateur	Options du compilateur

				`"cuda-ver": "10.2"}`
				
				`{"gpu-code": "sm_72", "trt-ver" : "8.2.1", "cuda-ver": "10.2"}`

g4dn.xlarge (EC2 instances (G4) avec NVIDIA T4 Tensor Core) GPUs	Linux	X86_64/X86-64	NVIDIA	`{"gpu-code": "sm_75", "trt-ver" : "7.1.3", "cuda-ver": "10.2"}`
g5.xlarge (EC2 instances (G5) avec noyau tenseur NVIDIA A10G) GPUs	Linux	X86_64/X86-64	NVIDIA	`{"gpu-code": "sm_80", "trt-ver" : "8.2.0", "cuda-ver": "11.2"}`
c5.2xlarge (instances Amazon EC2 C5)	Linux	X86_64/X86-64	CPU	`{"mcpu": "core-avx2"}`

Mémoire et stockage de l'appareil principal

Pour exécuter un seul modèle et l'agent Amazon Lookout for Vision Edge, les exigences de mémoire et de stockage de votre appareil principal sont les suivantes. Il se peut que vous ayez besoin de plus de mémoire et de stockage pour le composant de votre application cliente.

- Stockage : au moins 1,5 Go.
- Mémoire : au moins 6 Go pour un modèle en cours d'exécution.

Logiciel requis

Un périphérique principal nécessite les logiciels suivants.

Appareils Jetson

Si votre appareil principal est un appareil Jetson, vous devez installer les logiciels suivants sur le périphérique principal.

Logiciels	Versions prises en charge
Kit de développement logiciel Jetpack	4.4 à 4.6.1
Python et environnement virtuel Python pour Lookout for Vision Edge Agent version 1.x	3.8 ou 3.9

Matériel X86

Si votre périphérique principal utilise du matériel x86, les logiciels suivants doivent être installés sur le périphérique principal.

Inférence du processeur

Logiciels	Versions prises en charge
Python et environnement virtuel Python pour Lookout for Vision Edge Agent version 1.x	3.8 ou 3.9

Inférence accélérée par GPU

Les versions du logiciel varient en fonction de la microarchitecture du GPU NVIDIA que vous utilisez.

GPU NVIDIA avec microarchitecture antérieure à Ampere (capacité de calcul inférieure à 8,0)

Logiciel requis pour un GPU NVIDIA doté d'une microarchitecture antérieure à Ampere (capacité de calcul inférieure à 8,0). Le `gpu-code` doit être inférieur à`sm_80`.

Logiciels	Versions prises en charge
NVIDIA CUDA	10.2
NVIDIA TensorRT	Au moins 7.1.3 et moins de 8.0.0
Python et environnement virtuel Python pour Lookout for Vision Edge Agent version 1.x	3.8 ou 3.9

GPU NVIDIA avec microarchitecture Ampere (capacité de calcul 8.0)

Logiciel requis pour un GPU NVIDIA doté de la microarchitecture Ampere (capacité de calcul 8.0). Ils `gpu-code` doivent l'être`sm_80`).

Logiciels	Versions prises en charge
NVIDIA CUDA	11.2
NVIDIA TensorRT	8.2.0
Python et environnement virtuel Python pour Lookout for Vision Edge Agent version 1.x	3.8 ou 3.9

Configuration de votre appareil AWS IoT Greengrass Version 2 principal

Amazon Lookout for Vision AWS IoT Greengrass Version 2 simplifie le déploiement du composant modèle, du composant Amazon Lookout for Vision Edge Agent et du composant d'application client sur votre appareil principal. AWS IoT Greengrass V2 Pour plus d'informations sur les appareils et le

matériel que vous pouvez utiliser, consultezAWS IoT Greengrass Version 2 exigences de base en matière d'appareils.

Configuration de votre appareil principal

Utilisez les informations suivantes pour configurer votre appareil principal.

Pour configurer votre appareil principal

1. Configurez vos bibliothèques de GPU. N'effectuez pas cette étape si vous n'utilisez pas l'inférence accélérée par GPU.

   a. Vérifiez que vous disposez d'un processeur graphique compatible CUDA. Pour plus d'informations, consultez Vérifier que vous disposez d'un GPU compatible CUDA.

   b. Configurez CUDA, cuDNN et TensorRT sur votre appareil en effectuant l'une des opérations suivantes :

      - Si vous utilisez un appareil Jetson, installez les JetPack versions 4.4 à 4.6.1. Pour plus d'informations, consultez la section JetPack Archive.
      - Si vous utilisez du matériel x86 et que la microarchitecture de votre GPU NVIDIA est antérieure à Ampere (capacité de calcul inférieure à 8,0), procédez comme suit :
        1. Configurez la version 10.2 de CUDA en suivant les instructions du Guide d'installation de NVIDIA CUDA pour Linux.
        2. Installez cuDNN en suivant les instructions de la documentation NVIDIA cuDNN.
        3. Configurez TensorRT (version 7.1.3 ou ultérieure, mais antérieure à 8.0.0) en suivant les instructions de la DOCUMENTATION NVIDIA TENSORRT.
      - Si vous utilisez du matériel x86 et que la microarchitecture de votre GPU NVIDIA est Ampère (capacité de calcul 8.0), procédez comme suit :
        1. Configurez CUDA (version 11.2) en suivant les instructions du Guide d'installation NVIDIA CUDA pour Linux.
        2. Installez cuDNN en suivant les instructions de la documentation NVIDIA cuDNN.
        3. Configurez TensorRT (version 8.2.0) en suivant les instructions de la documentation NVIDIA TENSORRT.

2. Installez le logiciel AWS IoT Greengrass Version 2 principal sur votre appareil principal. Pour plus d'informations, consultez la section Installation du logiciel AWS IoT Greengrass Core dans le manuel du AWS IoT Greengrass Version 2 développeur.

3. Pour lire le contenu du compartiment Amazon S3 qui stocke le modèle, attachez une autorisation au rôle IAM (rôle d'échange de jetons) que vous créez lors de la AWS IoT Greengrass Version 2 configuration. Pour plus d'informations, consultez Autoriser l'accès aux compartiments S3 pour les artefacts des composants.
4. À l'invite de commande, entrez la commande suivante pour installer Python et un environnement virtuel Python sur le périphérique principal.

   ```
   sudo apt install python3.8 python3-venv python3.8-venv
   ```

5. Utilisez la commande suivante pour ajouter l'utilisateur Greengrass au groupe vidéo. Cela permet aux composants déployés par Greengrass d'accéder au GPU :

   ```
   sudo usermod -a -G video ggc_user
   ```

6. (Facultatif) Si vous souhaitez appeler l'API Lookout for Vision Edge Agent depuis un autre utilisateur, ajoutez l'utilisateur requis au`ggc_group`. Cela permet à l'utilisateur de communiquer avec l'agent Lookout for Vision Edge via le socket du domaine Unix :

   ```
   sudo usermod -a -G ggc_group $(whoami)
   ```

Emballage de votre modèle Amazon Lookout for Vision

Une tâche d'empaquetage de modèles regroupe un modèle Amazon Lookout for Vision en tant que composant du modèle.

Pour créer une tâche d'empaquetage de modèles, vous choisissez le modèle que vous souhaitez empaqueter et vous définissez les paramètres du composant de modèle créé par la tâche. Vous ne pouvez empaqueter qu'un modèle qui a été correctement entraîné.

Vous pouvez utiliser la console AWS ou le SDK Lookout for Vision pour créer la tâche d'empaquetage du modèle. Vous pouvez également obtenir des informations sur les modèles d'emballage que vous créez. Pour de plus amples informations, veuillez consulter Obtenir des informations sur les travaux de maquette d'emballage. Vous pouvez utiliser AWS IoT Greengrass V2 la console ou le AWS SDK pour déployer les composants sur le périphérique AWS IoT Greengrass Version 2 principal.

Rubriques

- Réglages du package

- Emballage de votre modèle (console)
- Emballage de votre modèle (SDK)
- Obtenir des informations sur les travaux de maquette d'emballage

Réglages du package

Utilisez les informations suivantes pour définir les paramètres de package pour votre modèle de tâche d'emballage.

Pour créer un modèle de tâche d'emballage, voir Emballage de votre modèle (console) ouEmballage de votre modèle (SDK).

Rubriques

- Matériel cible
- Réglages des composants

Matériel cible

Vous pouvez choisir un appareil ou une plate-forme cible pour votre modèle, mais pas les deux. Pour de plus amples informations, veuillez consulter Appareils, architectures de puces et systèmes d'exploitation testés.

Appareil cible

L'appareil cible du modèle, tel que NVIDIA® Jetson AGX Xavier. Il n'est pas nécessaire de spécifier les options du compilateur.

Plateforme cible

Amazon Lookout for Vision prend en charge les configurations de plateforme suivantes :

- Architectures X86_64 (version 64 bits du jeu d'instructions x86) et Aarch64 (ARMv8 processeur 64 bits).
- Système d'exploitation Linux
- Inférence à l'aide d'accélérateurs NVIDIA ou CPU.

Vous devez spécifier les options de compilation appropriées pour votre plate-forme cible.

Options du compilateur

Les options du compilateur vous permettent de spécifier la plate-forme cible pour votre appareil AWS IoT Greengrass Version 2 principal. Actuellement, vous pouvez spécifier les options de compilation suivantes.

Accélérateur NVIDIA

- `gpu-code`— Spécifie le code GPU du périphérique principal qui exécute le composant du modèle.
- `trt-ver`— Spécifie la version de TensorRT au format x.y.z.
- `cuda-ver`— Spécifie la version CUDA au format x.y.

Accélérateur CPU

- (Facultatif) `mcpu` — Spécifie le jeu d'instructions. Par exemple, `core-avx2`. Si vous ne fournissez aucune valeur, Lookout for Vision l'`core-avx2`utilise.

Vous spécifiez les options au format JSON. Par exemple :

```
{"gpu-code": "sm_75", "trt-ver": "7.1.3", "cuda-ver": "10.2"}
```

Pour obtenir plus d’exemples, consultez Appareils, architectures de puces et systèmes d'exploitation testés.

Réglages des composants

La tâche d'empaquetage du modèle crée un composant de modèle qui contient votre modèle. La tâche crée des artefacts qui sont AWS IoT Greengrass V2 utilisés pour déployer le composant du modèle sur le périphérique principal.

Vous ne pouvez pas créer un composant de modèle avec le même nom et la même version de composant qu'un composant existant.

Nom du composant

Nom du composant du modèle créé par Lookout for Vision lors du packaging du modèle. Le nom du composant que vous spécifiez s'affiche dans la AWS IoT Greengrass V2 console. Vous utilisez le nom du composant dans la recette que vous créez pour le composant de l'application client. Pour de plus amples informations, veuillez consulter Création du composant d'application client.

Description du composant

(Facultatif) Description du composant du modèle.

Version du composant

Numéro de version du composant du modèle. Vous pouvez accepter le numéro de version par défaut ou choisir le vôtre. Le numéro de version doit suivre le système de numéro de version sémantique : major.minor.patch. Par exemple, la version 1.0.0 représente la première version majeure d'un composant. Pour plus d'informations, veuillez consulter la rubrique Gestion des versions sémantique 2.0.0. Si vous ne fournissez aucune valeur, Lookout for Vision utilise le numéro de version de votre modèle pour générer une version pour vous.

Emplacement des composants

L'emplacement Amazon S3 où vous souhaitez que la tâche d'empaquetage du modèle enregistre les artefacts des composants du modèle. Le compartiment Amazon S3 doit se trouver dans la même région AWS et dans le même compte AWS que ceux que vous utilisez AWS IoT Greengrass Version 2. Pour créer un compartiment Amazon S3, consultez Création d'un compartiment.

Balises

Vous pouvez identifier, organiser, rechercher et filtrer vos composants à l'aide de balises. Chaque balise est une étiquette composée d'une clé définie par l'utilisateur et d'une valeur. Les balises sont attachées au composant du modèle lorsque la tâche d'empaquetage du modèle crée le composant du modèle dans Greengrass. Un composant est une ressource AWS IoT Greengrass V2. Les balises ne sont associées à aucune de vos ressources Lookout for Vision, telles que vos modèles. Pour plus d'informations, consultez Balisage des ressources AWS.

Emballage de votre modèle (console)

Vous pouvez créer une tâche d'empaquetage de modèles à l'aide de la console Amazon Lookout for Vision.

Pour plus d'informations sur les paramètres du package, consultezRéglages du package.

Pour empaqueter un modèle (console)

1. Créez un compartiment Amazon S3, ou réutilisez un compartiment existant, que Lookout for Vision utilise pour stocker les artefacts de la tâche d'empaquetage (composant du modèle).

2. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
3. Choisissez Démarrer.
4. Dans le volet de navigation de gauche, choisissez Projets.
5. Dans la section Projets, choisissez le projet qui contient le modèle que vous souhaitez empaqueter.
6. Dans le volet de navigation de gauche, sous le nom du projet, choisissez Edge model packages.
7. Dans la section Tâches de maquette d'emballage, choisissez Créer une tâche d'empaquetage modèle.
8. Entrez les paramètres du package. Pour de plus amples informations, veuillez consulter Réglages du package.
9. Choisissez Créer un modèle de tâche d'emballage.
10. Attendez que le travail d'emballage soit terminé. La tâche est terminée lorsque le statut de la tâche est Réussite.
11. Choisissez la tâche d'emballage dans la section Modèles de tâches d'emballage.
12. Choisissez Poursuivre le déploiement dans Greengrass pour poursuivre le déploiement du composant de votre modèle dans. AWS IoT Greengrass Version 2 Pour de plus amples informations, veuillez consulter Déploiement de vos composants sur un appareil.

Emballage de votre modèle (SDK)

Vous empaqueter un modèle en tant que composant de modèle en créant une tâche d'empaquetage de modèle. Pour créer un modèle de tâche d'empaquetage, vous devez appeler l'StartModelPackagingJobAPI. La tâche peut prendre un certain temps. Pour connaître l'état actuel, appelez DescribeModelPackagingJobet vérifiez le `Status` champ dans la réponse.

Pour plus d'informations sur les paramètres du package, consultezRéglages du package.

La procédure suivante explique comment démarrer une tâche d'empaquetage à l'aide de la AWS CLI. Vous pouvez empaqueter le modèle pour une plate-forme ou un équipement cible. Par exemple, le code Java, voir StartModelPackagingJob.

Pour emballer votre modèle (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. Assurez-vous que vous disposez des autorisations appropriées pour démarrer une tâche d'empaquetage de modèles. Pour de plus amples informations, veuillez consulter StartModelPackagingJob.
3. Utilisez les commandes CLI suivantes pour empaqueter votre modèle pour un équipement cible ou une plate-forme cible.

   Target platform

   La commande CLI suivante montre comment empaqueter un modèle pour une plate-forme cible avec un accélérateur NVIDIA.

   Remplacez les valeurs suivantes :

   - `project_name`au nom du projet qui contient le modèle que vous souhaitez empaqueter.
   - `model_version`à la version du modèle que vous souhaitez empaqueter.
   - (Facultatif) `description` à une description de votre modèle d'emballage.
   - `architecture`à l'architecture (`ARM64`ou`X86_64`) du périphérique AWS IoT Greengrass Version 2 principal sur lequel vous exécutez le composant du modèle.
   - `gpu_code`au code gpu du périphérique principal sur lequel vous exécutez le composant du modèle.
   - `trt_ver`à la version de TensorRT que vous avez installée sur votre appareil principal.
   - `cuda_ver`à la version CUDA que vous avez installée sur votre appareil principal.
   - `component_name`au nom du composant du modèle sur lequel vous souhaitez créer AWS IoT Greengrass V2.
   - (Facultatif) `component_version` vers une version pour le composant du modèle créé par la tâche d'empaquetage. Utilisez le format `major.minor.patch`. Par exemple, 1.0.0 représente la première version majeure d'un composant.
   - `bucket`vers le compartiment Amazon S3 dans lequel la tâche d'empaquetage stocke les artefacts des composants du modèle.
   - `prefix`à l'emplacement du compartiment Amazon S3 où la tâche d'empaquetage stocke les artefacts des composants du modèle.
   - (Facultatif) `component_description` à une description du composant du modèle.
   - (Facultatif) `tag_key1` et `tag_key2` aux clés des balises attachées au composant du modèle.

- (Facultatif) `tag_value1` et `tag_value2` aux valeurs clés des balises associées au composant du modèle.

```
aws lookoutvision start-model-packaging-job \
  --project-name project_name \
  --model-version model_version \
  --description="description" \
  --configuration
 "Greengrass={TargetPlatform={Os='LINUX',Arch='architecture',Accelerator='NVIDIA'},Compi
code\": \"gpu_code\", \"trt-ver\": \"trt_ver\", \"cuda-ver\":
 \"cuda_ver\"}',S3OutputLocation={Bucket='bucket',Prefix='prefix'},ComponentName='Compon
 {Key='tag_key2',Value='tag_value2'}]}" \
  --profile lookoutvision-access
```

Par exemple :

```
aws lookoutvision start-model-packaging-job \
  --project-name test-project-01 \
  --model-version 1 \
  --description="Model Packaging Job for G4 Instance using TargetPlatform
 Option" \
  --configuration
 "Greengrass={TargetPlatform={Os='LINUX',Arch='X86_64',Accelerator='NVIDIA'},CompilerOpt
code\": \"sm_75\", \"trt-ver\": \"7.1.3\", \"cuda-ver\":
 \"10.2\"}',S3OutputLocation={Bucket='bucket',Prefix='test-project-01/
folder'},ComponentName='SampleComponentNameX86TargetPlatform',ComponentVersion='0.1.0',C
 is my component',Tags=[{Key='modelKey0',Value='modelValue'},
 {Key='modelKey1',Value='modelValue'}]}" \
  --profile lookoutvision-access
```

Target Device

Utilisez les commandes CLI suivantes pour empaqueter un modèle pour une machine cible.

Remplacez les valeurs suivantes :

- `project_name`au nom du projet qui contient le modèle que vous souhaitez empaqueter.
- `model_version`à la version du modèle que vous souhaitez empaqueter.
- (Facultatif) `description` à une description de votre modèle d'emballage.

- `component_name`au nom du composant du modèle sur lequel vous souhaitez créer AWS IoT Greengrass V2.
- (Facultatif) `component_version` vers une version pour le composant du modèle créé par la tâche d'empaquetage. Utilisez le format `major.minor.patch`. Par exemple, 1.0.0 représente la première version majeure d'un composant.
- `bucket`vers le compartiment Amazon S3 dans lequel la tâche d'empaquetage stocke les artefacts des composants du modèle.
- `prefix`à l'emplacement du compartiment Amazon S3 où la tâche d'empaquetage stocke les artefacts des composants du modèle.
- (Facultatif) `component_description` à une description du composant du modèle.
- (Facultatif) `tag_key1` et `tag_key2` aux clés des balises attachées au composant du modèle.
- (Facultatif) `tag_value1` et `tag_value2` aux valeurs clés des balises associées au composant du modèle.

```
aws lookoutvision start-model-packaging-job \
  --project-name project_name \
  --model-version model_version \
  --description="description" \
  --configuration
 "Greengrass={TargetDevice='jetson_xavier',S3OutputLocation={Bucket='bucket',Prefix='pre
 {Key='tag_key2',Value='tag_value2'}]}" \
  --profile lookoutvision-access
```

Par exemple :

```
aws lookoutvision start-model-packaging-job \
  --project-name project_01 \
  --model-version 1 \
  --description="description" \
  --configuration
 "Greengrass={TargetDevice='jetson_xavier',S3OutputLocation={Bucket='bucket',Prefix='com
 model component',Tags=[{Key='tag_key1',Value='tag_value1'},
 {Key='tag_key2',Value='tag_value2'}]}" \
  --profile lookoutvision-access
```

4. Notez la valeur de JobName dans la réponse. Vous en avez besoin à l'étape suivante. Par exemple :

```
{
    "JobName": "6bcfd0ff-90c3-4463-9a89-6b4be3daf972"
}
```

5. `DescribeModelPackagingJob`Utilisez-le pour obtenir le statut actuel de la tâche. Modifiez les éléments suivants :

   - `project_name`au nom du projet que vous utilisez.
   - `job_name`au nom de la tâche que vous avez noté à l'étape précédente.

```
aws lookoutvision describe-model-packaging-job \
    --project-name project_name \
    --job-name job_name \
    --profile lookoutvision-access
```

   La tâche d'empaquetage du modèle est terminée si la valeur de `Status` est`SUCCEEDED`. Si la valeur est différente, attendez une minute et réessayez.

6. Poursuivez le déploiement à l'aide de AWS IoT Greengrass V2. Pour de plus amples informations, veuillez consulter Déploiement de vos composants sur un appareil.

Obtenir des informations sur les travaux de maquette d'emballage

Vous pouvez utiliser la console AWS et le SDK Amazon Lookout for Vision pour obtenir des informations sur les tâches d'empaquetage de modèles que vous créez.

Rubriques

- Obtenir des informations sur les tâches d'empaquetage des modèles (Console)
- Obtenir des informations sur les tâches d'emballage des modèles (SDK)

Obtenir des informations sur les tâches d'empaquetage des modèles (Console)

Pour obtenir des informations sur les tâches d'empaquetage du modèle (console)

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Choisissez Démarrer.
3. Dans le volet de navigation de gauche, choisissez Projets.
4. Dans la section Projets, choisissez le projet qui contient le modèle de tâche d'empaquetage que vous souhaitez consulter.
5. Dans le volet de navigation de gauche, sous le nom du projet, choisissez Edge model packages.
6. Dans la section Modèle de tâche d'emballage, choisissez le modèle de tâche d'emballage que vous souhaitez consulter. La page de détails de la tâche d'empaquetage du modèle s'affiche.

Obtenir des informations sur les tâches d'emballage des modèles (SDK)

Vous pouvez utiliser le AWS SDK pour répertorier les tâches d'empaquetage de modèles dans un projet et obtenir des informations sur une tâche d'empaquetage de modèle spécifique.

Répertorier les travaux d'emballage sur

Vous pouvez répertorier les tâches d'empaquetage de modèles dans un projet en appelant l'ListModelPackagingJobsAPI. La réponse inclut une liste d'ModelPackagingJobMetadataobjets fournissant des informations sur chaque tâche d'empaquetage de modèles. Un jeton de pagination est également inclus, que vous pouvez utiliser pour obtenir la prochaine série de résultats, si la liste est incomplète.

Pour répertorier vos travaux d'emballage de modèles

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 la commande CLI suivante. Modifiez `project_name` le nom du projet que vous souhaitez utiliser.

```
aws lookoutvision list-model-packaging-jobs \
  --project-name project_name \
  --profile lookoutvision-access
```

Décrire un modèle de travail d'emballage

Utilisez l'DescribeModelPackagingJobAPI pour obtenir des informations sur une tâche d'empaquetage de modèles. La réponse est un ModelPackagingDescriptionobjet qui inclut l'état actuel de la tâche et d'autres informations.

Pour décrire un package

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 la commande CLI suivante. Modifiez les éléments suivants :

   - `project_name`au nom du projet que vous utilisez.
   - `job_name`au nom de la tâche. Vous obtenez le nom de la tâche (`JobName`) lorsque vous appelez StartModelPackagingJob.

```
aws lookoutvision describe-model-packaging-job \
    --project-name project_name \
    --job-name job_name \
    --profile lookoutvision-access
```

Écrire le composant de votre application client

Un composant d'application client est un AWS IoT Greengrass Version 2 composant personnalisé que vous écrivez. Il met en œuvre la logique métier dont vous avez besoin pour utiliser un modèle Amazon Lookout for Vision sur AWS IoT Greengrass Version 2 un appareil principal.

Pour accéder à un modèle, le composant de votre application client utilise le composant Lookout for Vision Edge Agent. Le composant Lookout for Vision Edge Agent fournit une API que vous pouvez utiliser pour analyser des images avec un modèle et gérer les modèles sur un appareil principal.

L'API Lookout for Vision Edge Agent est implémentée à l'aide de gRPC, un protocole permettant d'effectuer des appels de procédure à distance. Pour plus d'informations, consultez gRPC. Pour écrire votre code, vous pouvez utiliser n'importe quel langage supporté par gRPC. Nous fournissons un exemple de code Python. Pour de plus amples informations, veuillez consulter Utilisation d'un modèle dans le composant de votre application client.

> **Note**
>
> Le composant Lookout for Vision Edge Agent dépend du composant modèle que vous déployez. Il est automatiquement déployé sur le périphérique principal lorsque vous déployez le composant modèle sur le périphérique principal.

Pour écrire un composant d'application client, procédez comme suit.

1. Configurez votre environnement pour utiliser gRPC et installez des bibliothèques tierces.
2. Écrivez le code pour utiliser le modèle.
3. Déployez le code en tant que composant personnalisé sur le périphérique principal.

Pour un exemple de composant d'application client qui montre comment effectuer la détection d'anomalies dans un GStreamer pipeline personnalisé, consultez https://github.com/awslabs/aws-greengrass-labs-lookoutvision-gstreamer.

Configuration de votre environnement

Pour écrire du code client, votre environnement de développement se connecte à distance à un appareil AWS IoT Greengrass Version 2 principal sur lequel vous avez déployé un composant du modèle Amazon Lookout for Vision et ses dépendances. Vous pouvez également écrire du code sur un périphérique principal. Pour plus d'informations, consultez les outils de développement AWS IoT Greengrass et Develop AWS IoT Greengrass components.

Votre code client doit utiliser le client gRPC pour accéder à l'agent Amazon Lookout for Vision Edge. Cette section explique comment configurer votre environnement de développement avec gRPC et installer les dépendances tierces nécessaires à l'`DetectAnomalies`exemple de code.

Une fois que vous avez terminé d'écrire votre code client, vous créez un composant personnalisé et vous le déployez sur vos appareils Edge. Pour de plus amples informations, veuillez consulter Création du composant d'application client.

Rubriques

- Configuration de gRPC
- Ajouter des dépendances tierces

Configuration de gRPC

Dans votre environnement de développement, vous avez besoin d'un client gRPC que vous utilisez dans votre code pour appeler l'API Lookout for Vision Edge Agent. Pour ce faire, vous créez un stub gRPC à l'aide d'un fichier de définition de `.proto` service pour l'agent Lookout for Vision Edge.

> **Note**
>
> Vous pouvez également obtenir le fichier de définition du service dans le bundle d'applications Lookout for Vision Edge Agent. Le bundle d'applications est installé lorsque le composant Lookout for Vision Edge Agent est installé en tant que dépendance du composant du modèle. Le bundle d'applications se trouve à l'adresse`/greengrass/v2/packages/artifacts-unarchived/aws.iot.lookoutvision.EdgeAgent/`*`edge_agent_version`*`/lookoutvision_edge_agent`. `edge_agent_version`Remplacez-le par la version de l'agent Lookout for Vision Edge que vous utilisez. Pour obtenir le bundle d'applications, vous devez déployer l'agent Lookout for Vision Edge sur un appareil principal.

Pour configurer gRPC

1. Téléchargez le fichier zip, proto.zip. Le fichier zip contient le fichier de définition du service .proto ()`edge-agent.proto`.
2. Décompressez le contenu.
3. Ouvrez une invite de commande et naviguez jusqu'au dossier qui contient`edge-agent.proto`.
4. Utilisez les commandes suivantes pour générer les interfaces client Python.

   ```
   %%bash
   python3 -m pip install grpcio
   python3 -m pip install grpcio-tools
   python3 -m grpc_tools.protoc --proto_path=. --python_out=. --grpc_python_out=.
    edge-agent.proto
   ```

   Si les commandes aboutissent, les stubs `edge_agent_pb2_grpc.py` `edge_agent_pb2.py` sont créés dans le répertoire de travail.
5. Écrivez le code client qui utilise votre modèle. Pour de plus amples informations, veuillez consulter Utilisation d'un modèle dans le composant de votre application client.

Ajouter des dépendances tierces

L'`DetectAnomalies`exemple de code utilise la bibliothèque Pillow pour travailler avec des images. Pour de plus amples informations, veuillez consulter Détection des anomalies à l'aide d'octets d'image.

Utilisez la commande suivante pour installer la bibliothèque Pillow.

```
python3 -m pip install Pillow
```

Utilisation d'un modèle dans le composant de votre application client

Les étapes d'utilisation d'un modèle issu d'un composant d'application client sont similaires à celles d'un modèle hébergé dans le cloud.

1. Commencez à exécuter le modèle.
2. Détectez les anomalies dans les images.
3. Arrêtez le modèle s'il n'est plus nécessaire.

L'agent Amazon Lookout for Vision Edge fournit une API permettant de démarrer un modèle, de détecter des anomalies dans une image et d'arrêter un modèle. Vous pouvez également utiliser l'API pour répertorier les modèles d'un appareil et obtenir des informations sur un modèle déployé. Pour de plus amples informations, veuillez consulter Référence de l'API Amazon Lookout for Vision Edge Agent.

Vous pouvez obtenir des informations sur les erreurs en vérifiant les codes d'état gRPC. Pour de plus amples informations, veuillez consulter Obtenir des informations sur les erreurs.

Pour écrire votre code, vous pouvez utiliser n'importe quel langage supporté par gRPC. Nous fournissons un exemple de code Python.

Rubriques

- Utilisation du stub dans le composant de votre application cliente
- Démarrage du modèle
- Détecter les anomalies
- Arrêter le modèle
- Répertorier des modèles sur un appareil

- Décrire un modèle
- Obtenir des informations sur les erreurs

Utilisation du stub dans le composant de votre application cliente

Utilisez le code suivant pour configurer l'accès à votre modèle via l'agent Lookout for Vision Edge.

```
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2

# Creating stub.
 with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    # Add additional code that works with Edge Agent in this block to prevent resources
 leakage
```

Démarrage du modèle

Vous démarrez un modèle en appelant l'StartModelAPI. Le démarrage du modèle peut prendre un certain temps. Vous pouvez vérifier l'état actuel en appelantDescribeModel. Le modèle est en cours d'exécution si la valeur du `status` champ est En cours d'exécution.

Exemple de code

*`component_name`*Remplacez-le par le nom du composant de votre modèle.

```
import time

import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2


model_component_name = "component_name"


def start_model_if_needed(stub, model_name):
    # Starting model if needed.
    while True:
```

```
        model_description_response =
 stub.DescribeModel(pb2.DescribeModelRequest(model_component=model_name))
        print(f"DescribeModel() returned {model_description_response}")
        if model_description_response.model_description.status == pb2.RUNNING:
            print("Model is already running.")
            break
        elif model_description_response.model_description.status == pb2.STOPPED:
            print("Starting the model.")
            stub.StartModel(pb2.StartModelRequest(model_component=model_name))
            continue
        elif model_description_response.model_description.status == pb2.FAILED:
            raise Exception(f"model {model_name} failed to start")
        print(f"Waiting for model to start.")
        if model_description_response.model_description.status != pb2.STARTING:
            break
        time.sleep(1.0)


# Creating stub.
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    start_model_if_needed(stub, model_component_name)
```

Détecter les anomalies

Vous utilisez l'DetectAnomaliesAPI pour détecter les anomalies dans une image.

L'`DetectAnomalies`opération s'attend à ce que le bitmap de l'image soit transmis au format RGB888 compressé. Le premier octet représente le canal rouge, le deuxième octet représente le canal vert et le troisième octet représente le canal bleu. Si vous fournissez l'image dans un format différent, tel que BGR, les prédictions DetectAnomalies sont incorrectes.

Par défaut, OpenCV utilise le format BGR pour les bitmaps d'image. Si vous utilisez OpenCV pour capturer des images à des fins d'analyse`DetectAnomalies`, vous devez convertir l'image au format avant de la transmettre RGB888 à. `DetectAnomalies`

Les images que vous fournissez `DetectAnomalies` doivent avoir les mêmes dimensions de largeur et de hauteur que les images que vous avez utilisées pour entraîner le modèle.

Détection des anomalies à l'aide d'octets d'image

Vous pouvez détecter des anomalies dans une image en fournissant l'image sous forme d'octets d'image. Dans l'exemple suivant, les octets de l'image sont extraits d'une image stockée dans le système de fichiers local.

*sample.jpg*Remplacez-le par le nom du fichier image que vous souhaitez analyser. *component_name*Remplacez-le par le nom du composant de votre modèle.

```
import time

from PIL import Image
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2


model_component_name = "component_name"

....
# Detecting anomalies.
def detect_anomalies(stub, model_name, image_path):
    image = Image.open(image_path)
    image = image.convert("RGB")
    detect_anomalies_response = stub.DetectAnomalies(
        pb2.DetectAnomaliesRequest(
            model_component=model_name,
            bitmap=pb2.Bitmap(
                width=image.size[0],
                height=image.size[1],
                byte_data=bytes(image.tobytes())
            )
        )
    )
    print(f"Image is anomalous -
 {detect_anomalies_response.detect_anomaly_result.is_anomalous}")
    return detect_anomalies_response.detect_anomaly_result


# Creating stub.
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
```

```
    start_model_if_needed(stub, model_component_name)
    detect_anomalies(stub, model_component_name, "sample.jpg")
```

Détection des anomalies à l'aide d'un segment de mémoire partagée

Vous pouvez détecter des anomalies dans une image en fournissant l'image sous forme d'octets d'image dans le segment de mémoire partagée POSIX. Pour de meilleures performances, nous vous recommandons d'utiliser de la mémoire partagée pour les DetectAnomalies demandes. Pour de plus amples informations, veuillez consulter DetectAnomalies.

Arrêter le modèle

Si vous n'utilisez plus le modèle, l'StopModelAPI pour arrêter l'exécution du modèle.

```
stop_model_response = stub.StopModel(
    pb2.StopModelRequest(
         model_component=model_component_name
    )
)
print(f"New status of the model is {stop_model_response.status}")
```

Répertorier des modèles sur un appareil

Vous pouvez utiliser l'the section called “ListModels”API pour répertorier les modèles déployés sur un appareil.

```
models_list_response = stub.ListModels(
    pb2.ListModelsRequest()
)
for model in models_list_response.models:
    print(f"Model Details {model}")
```

Décrire un modèle

Vous pouvez obtenir des informations sur un modèle déployé sur un appareil en appelant l'DescribeModelAPI. L'utilisation `DescribeModel` est utile pour obtenir l'état actuel d'un modèle. Par exemple, vous devez savoir si un modèle est en cours d'exécution avant de pouvoir appeler`DetectAnomalies`. Pour obtenir un exemple de code, consultez Démarrage du modèle.

Obtenir des informations sur les erreurs

Les codes d'état gRPC sont utilisés pour rapporter les résultats de l'API.

Vous pouvez obtenir des informations d'erreur en détectant l'`RpcError`exception, comme indiqué dans l'exemple suivant. Pour plus d'informations sur les codes d'état d'erreur, consultez la rubrique de référence relative à une API.

```
# Error handling.
try:
    stub.DetectAnomalies(detect_anomalies_request)
except grpc.RpcError as e:
    print(f"Error code: {e.code()}, Status: {e.details()}")
```

Création du composant d'application client

Vous pouvez créer le composant d'application client une fois que vous avez généré vos stubs gRPC et que le code de votre application client est prêt. Le composant que vous créez est un composant personnalisé que vous déployez sur un appareil AWS IoT Greengrass Version 2 principal avec AWS IoT Greengrass V2. Une recette que vous créez décrit votre composant personnalisé. La recette inclut toutes les dépendances qui doivent également être déployées. Dans ce cas, vous spécifiez le composant du modèle dans lequel vous créezEmballage de votre modèle Amazon Lookout for Vision. Pour plus d'informations sur les recettes de composants, consultez la référence des recettes de AWS IoT Greengrass Version 2 composants.

Les procédures décrites dans cette rubrique expliquent comment créer le composant d'application client à partir d'un fichier de recette et le publier en tant que composant AWS IoT Greengrass V2 personnalisé. Vous pouvez utiliser la AWS IoT Greengrass V2 console ou le AWS SDK pour publier le composant.

Pour obtenir des informations détaillées sur la création d'un composant personnalisé, consultez la AWS IoT Greengrass V2documentation suivante.

- Développez et testez un composant sur votre appareil
- Créez des AWS composants IoT Greengrass
- Publiez des composants à déployer sur vos appareils principaux

Rubriques

- Autorisations IAM pour publier un composant d'application client

- Création de la recette
- Publication du composant de l'application client (console)
- Publication du composant d'application client (SDK)

Autorisations IAM pour publier un composant d'application client

Pour créer et publier le composant de votre application client, vous devez disposer des autorisations IAM suivantes :

- `greengrass:CreateComponentVersion`
- `greengrass:DescribeComponent`
- `s3:PutObject`

Création de la recette

Dans cette procédure, vous allez créer la recette d'un composant d'application client simple. Le code contenu `lookoutvision_edge_agent_example.py` répertorie les modèles déployés sur le périphérique et est automatiquement exécuté une fois que vous avez déployé le composant sur le périphérique principal. Pour consulter le résultat, consultez le journal du composant après le déploiement du composant. Pour de plus amples informations, veuillez consulter Déploiement de vos composants sur un appareil. Lorsque vous êtes prêt, utilisez cette procédure pour créer la recette du code qui implémente votre logique métier.

Vous créez la recette sous forme de fichier au format JSON ou YAML. Vous chargez également le code de l'application client dans un compartiment Amazon S3.

Pour créer la recette du composant de l'application client

1. Si ce n'est pas déjà fait, créez les fichiers stub gRPC. Pour de plus amples informations, veuillez consulter Configuration de gRPC.
2. Enregistrez le code suivant dans un fichier nommé `lookoutvision_edge_agent_example.py`

```
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2

# Creating stub.
```

```
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    # Add additional code that works with Edge Agent in this block to prevent
 resources leakage

    models_list_response = stub.ListModels(
        pb2.ListModelsRequest()
    )
    for model in models_list_response.models:
        print(f"Model Details {model}")
```

3. Créez un compartiment Amazon S3 (ou utilisez un compartiment existant) pour stocker les fichiers source du composant de votre application client. Le bucket doit se trouver dans votre AWS compte et dans la même AWS région que celle dans laquelle vous utilisez AWS IoT Greengrass Version 2 Amazon Lookout for Vision.
4. Téléchargez`lookoutvision_edge_agent_example.py`, `edge_agent_pb2_grpc.py` and `edge_agent_pb2.py` dans le compartiment Amazon S3 que vous avez créé à l'étape précédente. Notez le chemin Amazon S3 de chaque fichier. Vous avez créé `edge_agent_pb2_grpc.py` et `edge_agent_pb2.py` dansConfiguration de gRPC.
5. Dans un éditeur, créez le fichier de recette JSON ou YAML suivant.

   - `model_component`au nom du composant de votre modèle. Pour de plus amples informations, veuillez consulter Réglages des composants.
   - Remplacez les entrées d'URI par les chemins S3 de `lookoutvision_edge_agent_example.pyedge_agent_pb2_grpc.py`, `etedge_agent_pb2.py`.

   JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.lookoutvision.EdgeAgentPythonExample",
  "ComponentVersion": "1.0.0",
  "ComponentType": "aws.greengrass.generic",
  "ComponentDescription": "Lookout for Vision Edge Agent Sample Application",
  "ComponentPublisher": "Sample App Publisher",
  "ComponentDependencies": {
    "model_component": {
      "VersionRequirement": ">=1.0.0",
```

```
      "DependencyType": "HARD"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "pip3 install grpcio grpcio-tools protobuf Pillow",
        "run": {
          "script": "python3 {artifacts:path}/
lookoutvision_edge_agent_example.py"
        }
      },
      "Artifacts": [
        {
          "Uri": "S3 path to lookoutvision_edge_agent_example.py"
        },
        {
          "Uri": "S3 path to edge_agent_pb2_grpc.py"
        },
        {
          "Uri": "S3 path to edge_agent_pb2.py"
        }
      ]
    }
  ],
  "Lifecycle": {}
}
```

YAML

```
---
RecipeFormatVersion: 2020-01-25
ComponentName: com.lookoutvison.EdgeAgentPythonExample
ComponentVersion: 1.0.0
ComponentDescription: Lookout for Vision Edge Agent Sample Application
ComponentPublisher: Sample App Publisher
ComponentDependencies:
  model_component:
    VersionRequirement: '>=1.0.0'
    DependencyType: HARD
```

```
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: |-
        pip3 install grpcio
        pip3 install grpcio-tools
        pip3 install protobuf
        pip3 install Pillow
      run:
        script: |-
          python3 {artifacts:path}/lookout_vision_agent_example.py
    Artifacts:
      - URI: S3 path to lookoutvision_edge_agent_example.py
      - URI: S3 path to edge_agent_pb2_grpc.py
      - URI: S3 path to edge_agent_pb2.py
```

6. Enregistrez le fichier JSON ou YAML sur votre ordinateur.
7. Créez le composant de l'application client en effectuant l'une des opérations suivantes :
   - Si vous souhaitez utiliser la AWS IoT Greengrass console, faites-lePublication du composant de l'application client (console).
   - Si vous souhaitez utiliser le AWS SDK, faites-lePublication du composant d'application client (SDK).

Publication du composant de l'application client (console)

Vous pouvez utiliser la AWS IoT Greengrass V2 console pour publier le composant de l'application client.

Pour publier le composant de l'application client

1. Si ce n'est pas déjà fait, créez la recette pour le composant de votre application cliente en procédant ainsiCréation de la recette.
2. Ouvrez la AWS IoT Greengrass console à https://console.aws.amazon.com/iot/
3. Dans le volet de navigation de gauche, sous Greengrass, choisissez Components.
4. Sous Mes composants, sélectionnez Créer un composant.

5. Sur la page Créer un composant, choisissez Enter recipe as JSON si vous souhaitez utiliser une recette au format JSON. Choisissez Entrer la recette au format YAML si vous souhaitez utiliser une recette au format YAML.
6. Sous Recette, remplacez la recette existante par la recette JSON ou YAML que vous avez créée dansCréation de la recette.
7. Choisissez Créer un composant.
8. Déployez ensuite le composant de votre application client.

Publication du composant d'application client (SDK)

Vous pouvez publier le composant de l'application client à l'aide de l'CreateComponentVersionAPI.

Pour publier le composant d'application client (SDK)

1. Si ce n'est pas déjà fait, créez la recette pour le composant de votre application cliente en procédant ainsiCréation de la recette.
2. À l'invite de commande, entrez la commande suivante pour créer le composant d'application client. `recipe-file`Remplacez-le par le nom du fichier de recette dans lequel vous l'avez crééCréation de la recette.

   ```
   aws greengrassv2 create-component-version --inline-recipe fileb://recipe-file
   ```

   Notez l'ARN du composant dans la réponse. Vous en avez besoin à l'étape suivante.
3. Utilisez la commande suivante pour obtenir l'état du composant de l'application cliente. `component-arn`Remplacez-le par l'ARN que vous avez noté à l'étape précédente. Le composant de l'application client est prêt si la valeur de `componentState` est`DEPLOYABLE`.

   ```
   aws greengrassv2 describe-component --arn component-arn
   ```
4. Déployez ensuite le composant de votre application client.

Déploiement de vos composants sur un appareil

Pour déployer le composant modèle et le composant d'application client sur un appareil AWS IoT Greengrass Version 2 principal, vous devez utiliser la AWS IoT Greengrass V2 console ou l'CreateDeploymentAPI. Pour plus d'informations, consultez la section Créer des déploiements ou le

Guide du AWS IoT Greengrass Version 2 développeur. Pour plus d'informations sur la mise à jour d'un composant déployé sur un périphérique principal, consultez la section Réviser les déploiements.

Rubriques

- Autorisations IAM pour le déploiement de composants
- Déploiement de vos composants (console)
- Déploiement des composants (SDK)

Autorisations IAM pour le déploiement de composants

Pour déployer un composant, AWS IoT Greengrass V2 vous devez disposer des autorisations suivantes :

- `greengrass:ListComponents`
- `greengrass:ListComponentVersions`
- `greengrass:ListCoreDevices`
- `greengrass:CreateDeployment`
- `greengrass:GetDeployment`
- `greengrass:ListDeployments`

`CreateDeployment`et `GetDeployment` ont des actions dépendantes. Pour plus d'informations, consultez la section Actions définies par AWS IoT Greengrass V2.

Pour plus d'informations sur la modification des autorisations IAM, consultez la section Modification des autorisations d'un utilisateur.

Déploiement de vos composants (console)

Utilisez la procédure suivante pour déployer le composant d'application client sur un périphérique principal. L'application client dépend du composant du modèle (lui-même dépendant de l'agent Lookout for Vision Edge). Le déploiement du composant d'application client lance également le déploiement du composant modèle et de l'agent Lookout for Vision Edge.

> **Note**
>
> Vous pouvez ajouter vos composants à un déploiement existant. Vous pouvez également déployer des composants dans un groupe d'objets.

Pour exécuter cette procédure, vous devez disposer d'un périphérique AWS IoT Greengrass V2 principal configuré. Pour de plus amples informations, veuillez consulter Configuration de votre appareil AWS IoT Greengrass Version 2 principal.

Pour déployer vos composants sur un appareil

1. Ouvrez la AWS IoT Greengrass console à l'adresse https://console.aws.amazon.com/iot/.
2. Dans le volet de navigation de gauche, sous Greengrass, choisissez Deployments.
3. Sous Déploiements, choisissez Créer.
4. Sur la page Specify target (Spécifier une cible), procédez comme suit :
    1. Sous Deployment information (Informations sur le déploiement), saisissez ou modifiez le nom convivial de votre déploiement.
    2. Sous Cible de déploiement, sélectionnez le périphérique principal et entrez le nom de la cible.
    3. Choisissez Suivant.
5. Sur la page Sélectionner les composants, procédez comme suit :
    1. Sous Mes composants, choisissez le nom du composant de votre application client (`com.lookoutvison.EdgeAgentPythonExample`).
    2. Choisissez Next (Suivant)
6. Sur la page Configurer les composants, conservez la configuration actuelle et choisissez Next.
7. Sur la page Configurer les paramètres avancés, conservez les paramètres actuels et choisissez Next.
8. Sur la page Révision, choisissez Déployer pour commencer à déployer votre composant.

Vérification de l'état du déploiement (console)

Vous pouvez vérifier l'état de votre déploiement depuis la AWS IoT Greengrass V2 console. Si le composant de votre application client utilise l'exemple de recette et de codethe section called “Création du composant d'application client”, consultez le journal des composants de l'application

client une fois le déploiement terminé. En cas de réussite, le journal inclut une liste des modèles Lookout for Vision déployés sur le composant.

Pour plus d'informations sur l'utilisation du AWS SDK pour vérifier l'état du déploiement, voir Vérifier l'état du déploiement.

Pour vérifier l'état du déploiement

1. Ouvrez la AWS IoT Greengrass console à l'adresse https://console.aws.amazon.com/iot/
2. Dans le volet de navigation de gauche, choisissez Core devices.
3. Dans la section Appareils principaux de Greengrass, choisissez votre appareil principal.
4. Choisissez l'onglet Déploiements pour afficher l'état actuel du déploiement.
5. Une fois les déploiements réussis (le statut est terminé), ouvrez une fenêtre de terminal sur le périphérique principal et consultez le journal des composants de l'application client sur. `/greengrass/v2/logs/com.lookoutvison.EdgeAgentPythonExample.log` Si votre déploiement utilise l'exemple de recette et de code, le journal inclut le résultat de `lookoutvision_edge_agent_example.py`. Par exemple :

   ```
   Model Details model_component:"ModelComponent"
   ```

Déploiement des composants (SDK)

Utilisez la procédure suivante pour déployer le composant d'application client, le composant de modèle et l'agent Amazon Lookout for Vision Edge sur votre appareil principal.

1. Créez un `deployment.json` fichier pour définir la configuration de déploiement de vos composants. Ce fichier doit ressembler à l'exemple suivant.

   ```
   {
     "targetArn":"targetArn",
     "components": {
       "com.lookoutvison.EdgeAgentPythonExample": {
         "componentVersion": "1.0.0",
         "configurationUpdate": {
         }
       }
     }
   }
   ```

- Dans le champ `targetArn`, remplacez *`targetArn`* par l'Amazon Resource Name (ARN) de l'objet ou du groupe d'objets à cibler pour le déploiement, au format suivant :
  - Objet : `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
  - Groupe d'objets : `arn:aws:iot:`*`region`*`:`*`account-id:`*`thinggroup/`*`thingGroupName`*

2. Vérifiez si la cible de déploiement comporte un déploiement existant que vous souhaitez réviser. Procédez comme suit :

   a. Exécutez la commande suivante pour répertorier les déploiements pour la cible de déploiement. `targetArn`Remplacez-le par le Amazon Resource Name (ARN) de l'objet ou du groupe d' AWS objets IoT cible. Pour obtenir les éléments ARNs de la région AWS actuelle, utilisez la commande`aws iot list-things`.

   ```
   aws greengrassv2 list-deployments --target-arn targetArn
   ```

   La réponse contient une liste du dernier déploiement pour la cible. Si la réponse est vide, cela signifie que la cible n'a pas de déploiement existant et vous pouvez passer à l'étape 3. Sinon, copiez le contenu `deploymentId` de la réponse pour l'utiliser à l'étape suivante.

   b. Exécutez la commande suivante pour obtenir les détails du déploiement. Ces détails incluent les métadonnées, les composants et la configuration des tâches. `deploymentId`Remplacez-le par l'ID de l'étape précédente.

   ```
   aws greengrassv2 get-deployment --deployment-id deploymentId
   ```

   c. Copiez l'une des paires clé-valeur suivantes de la réponse de la commande précédente dans deployment.json. Vous pouvez modifier ces valeurs pour le nouveau déploiement.

   - `deploymentName`— Le nom du déploiement.
   - `components`— Les composants du déploiement. Pour désinstaller un composant, supprimez-le de cet objet.
   - `deploymentPolicies`— Les politiques du déploiement.
   - `tags`— Les tags du déploiement.

3. Exécutez la commande suivante pour déployer les composants sur l'appareil. Notez la valeur de `deploymentId` dans la réponse.

   ```
   aws greengrassv2 create-deployment \
   ```

```
    --cli-input-json file://path/to/deployment.json
```

4. Exécutez la commande suivante pour obtenir l'état du déploiement. `deployment-id`Remplacez la valeur que vous avez notée à l'étape précédente. Le déploiement s'est terminé avec succès si la valeur est de `deploymentStatus` 1. `COMPLETED`

```
aws greengrassv2 get-deployment --deployment-id deployment-id
```

5. Une fois les déploiements réussis, ouvrez une fenêtre de terminal sur le périphérique principal et consultez le journal des composants de l'application client sur. `/greengrass/v2/logs/com.lookoutvison.EdgeAgentPythonExample.log` Si votre déploiement utilise l'exemple de recette et de code, le journal inclut le résultat `delookoutvision_edge_agent_example.py`. Par exemple :

```
Model Details model_component:"ModelComponent"
```

Référence de l'API Amazon Lookout for Vision Edge Agent

Cette section est la référence de l'API pour l'agent Amazon Lookout for Vision Edge.

Détecter les anomalies à l'aide d'un modèle

Vous utilisez l'DetectAnomaliesAPI pour détecter les anomalies dans les images en utilisant un modèle en cours d'exécution sur un appareil AWS IoT Greengrass Version 2 principal.

Obtenir des informations sur le modèle

APIs qui obtiennent des informations sur les modèles déployés sur un appareil principal.

- ListModels
- DescribeModel

Exécution d'un modèle

APIs pour démarrer et arrêter un modèle Amazon Lookout for Vision déployé sur un appareil principal.

- StartModel

- StopModel

DetectAnomalies

Détecte les anomalies dans l'image fournie.

La réponse de `DetectAnomalies` inclut une prédiction booléenne indiquant que l'image contient une ou plusieurs anomalies et une valeur de confiance pour la prédiction. Si le modèle est un modèle de segmentation, la réponse inclut les éléments suivants :

- Une image de masque qui couvre chaque type d'anomalie dans une couleur unique. Vous pouvez `DetectAnomalies` stocker l'image du masque dans la mémoire partagée ou renvoyer le masque sous forme d'octets d'image.
- Pourcentage de zone de l'image couvert par un type d'anomalie.
- Couleur hexadécimale d'un type d'anomalie sur l'image du masque.

> **Note**
>
> Le modèle que vous utilisez `DetectAnomalies` doit être en cours d'exécution. Vous pouvez obtenir le statut actuel en appelantDescribeModel. Pour commencer à exécuter un modèle, voirStartModel.

`DetectAnomalies`prend en charge les bitmaps compressés (images) au format RGB888 entrelacé. Le premier octet représente le canal rouge, le deuxième octet représente le canal vert et le troisième octet représente le canal bleu. Si vous fournissez l'image dans un format différent, tel que BGR, les prédictions DetectAnomalies sont incorrectes.

Par défaut, OpenCV utilise le format BGR pour les bitmaps d'image. Si vous utilisez OpenCV pour capturer des images à des fins d'analyse`DetectAnomalies`, vous devez convertir l'image au format avant de la transmettre RGB888 à. `DetectAnomalies`

La dimension d'image minimale prise en charge est de 64 x 64 pixels. La dimension d'image maximale prise en charge est de 4 096 x 4 096 pixels.

Vous pouvez envoyer l'image dans le message protobuf ou via un segment de mémoire partagée. La sérialisation d'images de grande taille dans le message protobuf peut augmenter considérablement

la latence des appels à. `DetectAnomalies` Pour une latence minimale, nous vous recommandons d'utiliser de la mémoire partagée.

```
rpc DetectAnomalies(DetectAnomaliesRequest) returns (DetectAnomaliesResponse);
```

DetectAnomaliesRequest

Les paramètres d'entrée pour`DetectAnomalies`.

```
message Bitmap {
  int32 width = 1;
  int32 height = 2;
  oneof data {
    bytes byte_data = 3;
    SharedMemoryHandle shared_memory_handle = 4;
  }
}
```

```
message SharedMemoryHandle {
  string name = 1;
  uint64 size = 2;
  uint64 offset = 3;
}
```

```
message AnomalyMaskParams {
SharedMemoryHandle shared_memory_handle = 2;
}
```

```
message DetectAnomaliesRequest {
string model_component = 1;
Bitmap bitmap = 2;
AnomalyMaskParams anomaly_mask_params = 3;
}
```

Bitmap

L'image avec laquelle vous souhaitez analyser`DetectAnomalies`.

width

Largeur de l'image en pixels.

height

Hauteur de l'image en pixels.

données_octets

Octets d'image transmis dans le message protobuf.

handle de mémoire partagé

Octets d'image transmis dans le segment de mémoire partagée.

SharedMemoryHandle

Représente un segment de mémoire partagée POSIX.

name

Nom du segment de mémoire POSIX. Pour plus d'informations sur la création de mémoire partagée, consultez shm_open.

size

Taille de la mémoire tampon d'image en octets à partir du décalage.

offset

Le décalage, en octets, par rapport au début de la mémoire tampon d'image par rapport au début du segment de mémoire partagée.

AnomalyMaskParams

Paramètres de sortie d'un masque d'anomalie. (Modèle de segmentation).

handle de mémoire partagé

Contient les octets d'image pour le masque, s'ils `shared_memory_handle` n'ont pas été fournis.

DetectAnomaliesRequest

composant_modèle

Nom du AWS IoT Greengrass V2 composant qui contient le modèle que vous souhaitez utiliser.

bitmap

L'image avec laquelle vous souhaitez analyser`DetectAnomalies`.

anomaly_mask_params

Paramètres facultatifs pour la sortie du masque. (Modèle de segmentation).

DetectAnomaliesResponse

La réponse de`DetectAnomalies`.

```
message DetectAnomalyResult {
bool is_anomalous = 1;
float confidence = 2;
Bitmap anomaly_mask = 3;
repeated Anomaly anomalies = 4;
float anomaly_score = 5;
float anomaly_threshold = 6;
}
```

```
message Anomaly {
string name = 1;
PixelAnomaly pixel_anomaly = 2;
```

```
message PixelAnomaly {
float total_percentage_area = 1;
string hex_color = 2;
}
```

```
message DetectAnomaliesResponse {
  DetectAnomalyResult detect_anomaly_result = 1;
}
```

Anomalie

Représente une anomalie détectée sur une image. (Modèle de segmentation).

name

Nom d'un type d'anomalie détecté dans une image. `name`correspond à un type d'anomalie dans le jeu de données d'entraînement. Le service insère automatiquement le type d'anomalie d'arrière-plan dans le formulaire de DetectAnomalies réponse.

pixel_anomalie

Informations sur le masque de pixels qui couvre un type d'anomalie.

PixelAnomaly

Informations sur le masque de pixels qui couvre un type d'anomalie. (Modèle de segmentation).

superficie totale en pourcentage

Pourcentage de zone de l'image couvert par le type d'anomalie.

couleur hexagonale

Valeur de couleur hexadécimale qui représente le type d'anomalie sur l'image. La couleur correspond à la couleur du type d'anomalie utilisé dans le jeu de données d'apprentissage.

DetectAnomalyResult

est anormal

Indique si l'image contient une anomalie. `true`si l'image contient une anomalie. `false`si l'image est normale.

confidence

La confiance que l'on `DetectAnomalies` a dans l'exactitude de la prédiction. `confidence`est une valeur à virgule flottante comprise entre 0 et 1.

masque d'anomalie

si shared_memory_handle n'a pas été fourni, contient les octets d'image pour le masque. (Modèle de segmentation).

anomalies

Liste d'au moins 0 anomalies détectées dans l'image d'entrée. (Modèle de segmentation).

score d'anomalie

Nombre qui quantifie dans quelle mesure les anomalies prédites pour une image s'écartent d'une image sans anomalies. `anomaly_score`est une valeur flottante comprise entre et (écart le plus faible par rapport `0.0` à une image normale) à 1,0 (écart le plus élevé par rapport à une image normale). Amazon Lookout for Vision renvoie une valeur `anomaly_score` pour, même si la prédiction pour une image est normale.

seuil d'anomalie

Nombre (flottant) qui détermine si la classification prédite d'une image est normale ou anormale. Les images `anomaly_score` dont la valeur est égale ou supérieure à la valeur de `anomaly_threshold` sont considérées comme anormales. `anomaly_score`La valeur ci-dessous `anomaly_threshold` indique une image normale. La valeur utilisée par un modèle est calculée par Amazon Lookout for Vision lorsque vous entraînez le modèle. `anomaly_threshold` Vous ne pouvez pas définir ou modifier la valeur de `anomaly_threshold`

Codes d'état

Code	Nombre	Description
OK	0	`DetectAnomalies` a fait une prédiction avec succès
UNKNOWN	2	Une erreur inconnue s'est produite.
ARGUMENT_INVALIDE	3	Un ou plusieurs paramètres d'entrée ne sont pas valides. Consultez le message d'erreur pour plus de détails.
INTROUVABLE	5	Aucun modèle portant le nom spécifié n'a été trouvé.
RESSOURCE_ÉPUISÉE	8	Les ressources sont insuffisantes pour effectuer cette opération. Par exemple, l'agent Lookout for Vision Edge ne parvient pas à suivre le rythme des appels

Code	Nombre	Description
		`DetectAnomalies` adressés à. Consultez le message d'erreur pour plus de détails.
PRÉCONDITION DÉFAILLANTE	9	`DetectAnomalies` a été appelé pour un modèle qui n'est pas dans l'état RUNNING.
INTERNE	13	Une erreur interne s'est produite.

DescribeModel

Décrit un modèle Amazon Lookout for Vision déployé sur AWS IoT Greengrass Version 2 un appareil principal.

```
rpc DescribeModel(DescribeModelRequest) returns (DescribeModelResponse);
```

DescribeModelRequest

```
message DescribeModelRequest {
  string model_component = 1;
}
```

composant_modèle

Nom du AWS IoT Greengrass V2 composant qui contient le modèle que vous souhaitez décrire.

DescribeModelResponse

```
message ModelDescription {
  string model_component = 1;
  string lookout_vision_model_arn = 2;
```

```
  ModelStatus status = 3;
  string status_message = 4;
}
```

```
message DescribeModelResponse {
  ModelDescription model_description = 1;
}
```

ModelDescription

composant_modèle

Nom du AWS IoT Greengrass Version 2 composant qui contient le modèle Amazon Lookout for Vision.

lookout_vision_model_arn

Nom de ressource (ARN) Amazon du modèle Amazon Lookout for Vision utilisé pour générer AWS IoT Greengrass V2 le composant.

status

État actuel du modèle. Pour de plus amples informations, veuillez consulter ModelStatus.

message_statut

Message d'état du modèle.

Codes d’état

Code	Nombre	Description
OK	0	L'appel a été couronné de succès.
UNKNOWN	2	Une erreur inconnue s'est produite.
ARGUMENT_INVALIDE	3	Un ou plusieurs paramètres d'entrée ne sont pas valides.

Code	Nombre	Description
		Consultez le message d'erreur pour plus de détails.
INTROUVABLE	5	Aucun modèle portant le nom fourni n'a été trouvé.
INTERNE	13	Une erreur interne s'est produite.

ListModels

Répertorie les modèles déployés sur un périphérique AWS IoT Greengrass Version 2 principal.

```
rpc ListModels(ListModelsRequest) returns (ListModelsResponse);
```

ListModelsRequest

```
message ListModelsRequest {}
```

ListModelsResponse

```
message ModelMetadata {
  string model_component = 1;
  string lookout_vision_model_arn = 2;
  ModelStatus status = 3;
  string status_message = 4;
}
```

```
message ListModelsResponse {
  repeated ModelMetadata models = 1;
}
```

ModelMetadata

composant_modèle

Nom du AWS IoT Greengrass Version 2 composant qui contient un modèle Amazon Lookout for Vision.

lookout_vision_model_arn

Nom de ressource Amazon (ARN) du modèle Amazon Lookout for Vision utilisé pour générer AWS IoT Greengrass V2 le composant.

status

État actuel du modèle. Pour de plus amples informations, veuillez consulter ModelStatus.

message_statut

Message d'état du modèle.

Codes d’état

Code	Nombre	Description
OK	0	L'appel a été couronné de succès.
UNKNOWN	2	Une erreur inconnue s'est produite.
INTERNE	13	Une erreur interne s'est produite.

StartModel

Démarre un modèle exécuté sur un appareil AWS IoT Greengrass Version 2 principal. Le lancement du modèle peut prendre un certain temps. Pour vérifier le statut actuel, appelezDescribeModel. Le modèle est en cours d'exécution si le `Status` champ l'est`RUNNING`.

Le nombre de modèles que vous pouvez exécuter simultanément dépend des caractéristiques matérielles de votre périphérique principal.

```
rpc StartModel(StartModelRequest) returns (StartModelResponse);
```

StartModelRequest

```
message StartModelRequest {
  string model_component = 1;
}
```

composant_modèle

Nom du AWS IoT Greengrass Version 2 composant qui contient le modèle que vous souhaitez démarrer.

StartModelResponse

```
message StartModelResponse {
  ModelStatus status = 1;
}
```

status

État actuel du modèle. La réponse est `STARTING` si l'appel aboutit. Pour de plus amples informations, veuillez consulter ModelStatus.

Codes d’état

Code	Nombre	Description
OK	0	Le modèle démarre
UNKNOWN	2	Une erreur inconnue s'est produite.
ARGUMENT_INVALIDE	3	Un ou plusieurs paramètres d'entrée ne sont pas valides. Consultez le message d'erreur pour plus de détails.
INTROUVABLE	5	Aucun modèle portant le nom fourni n'a été trouvé.

Code	Nombre	Description
RESSOURCE_ÉPUISÉE	8	Les ressources sont insuffisantes pour effectuer cette opération. Par exemple, la mémoire est insuffisante pour charger le modèle. Consultez le message d'erreur pour plus de détails.
PRÉCONDITION DÉFAILLANTE	9	La méthode a été appelée pour un modèle qui n'est pas dans l'état STOPPED ou FAILED.
INTERNE	13	Une erreur interne s'est produite.

StopModel

Arrête l'exécution d'un modèle sur un appareil AWS IoT Greengrass Version 2 principal. `StopModel`revient après l'arrêt du modèle. Le modèle s'est arrêté correctement si le `Status` champ de la réponse est`STOPPED`.

```
  rpc StopModel(StopModelRequest) returns (StopModelResponse);
```

StopModelRequest

```
message StopModelRequest {
  string model_component = 1;
}
```

composant_modèle

Nom du AWS IoT Greengrass Version 2 composant qui contient le modèle que vous souhaitez arrêter.

StopModelResponse

```
message StopModelResponse {
  ModelStatus status = 1;
}
```

status

État actuel du modèle. La réponse est STOPPED si l'appel aboutit. Pour de plus amples informations, veuillez consulter ModelStatus.

Codes d'état

Code	Nombre	Description
OK	0	Le modèle s'arrête.
UNKNOWN	2	Une erreur inconnue s'est produite.
ARGUMENT_INVALIDE	3	Un ou plusieurs paramètres d'entrée ne sont pas valides. Consultez le message d'erreur pour plus de détails.
INTROUVABLE	5	Aucun modèle portant le nom fourni n'a été trouvé.
PRÉCONDITION DÉFAILLANTE	9	La méthode a été appelée pour un modèle qui n'est pas dans l'état RUNNING.
INTERNE	13	Une erreur interne s'est produite.

ModelStatus

État d'un modèle déployé sur un appareil AWS IoT Greengrass Version 2 principal. Pour connaître le statut actuel, appelezDescribeModel.

```
enum ModelStatus {
  STOPPED = 0;
  STARTING = 1;
  RUNNING = 2;
  FAILED = 3;
  STOPPING = 4;
}
```

Utilisation du tableau de bord Amazon Lookout for Vision

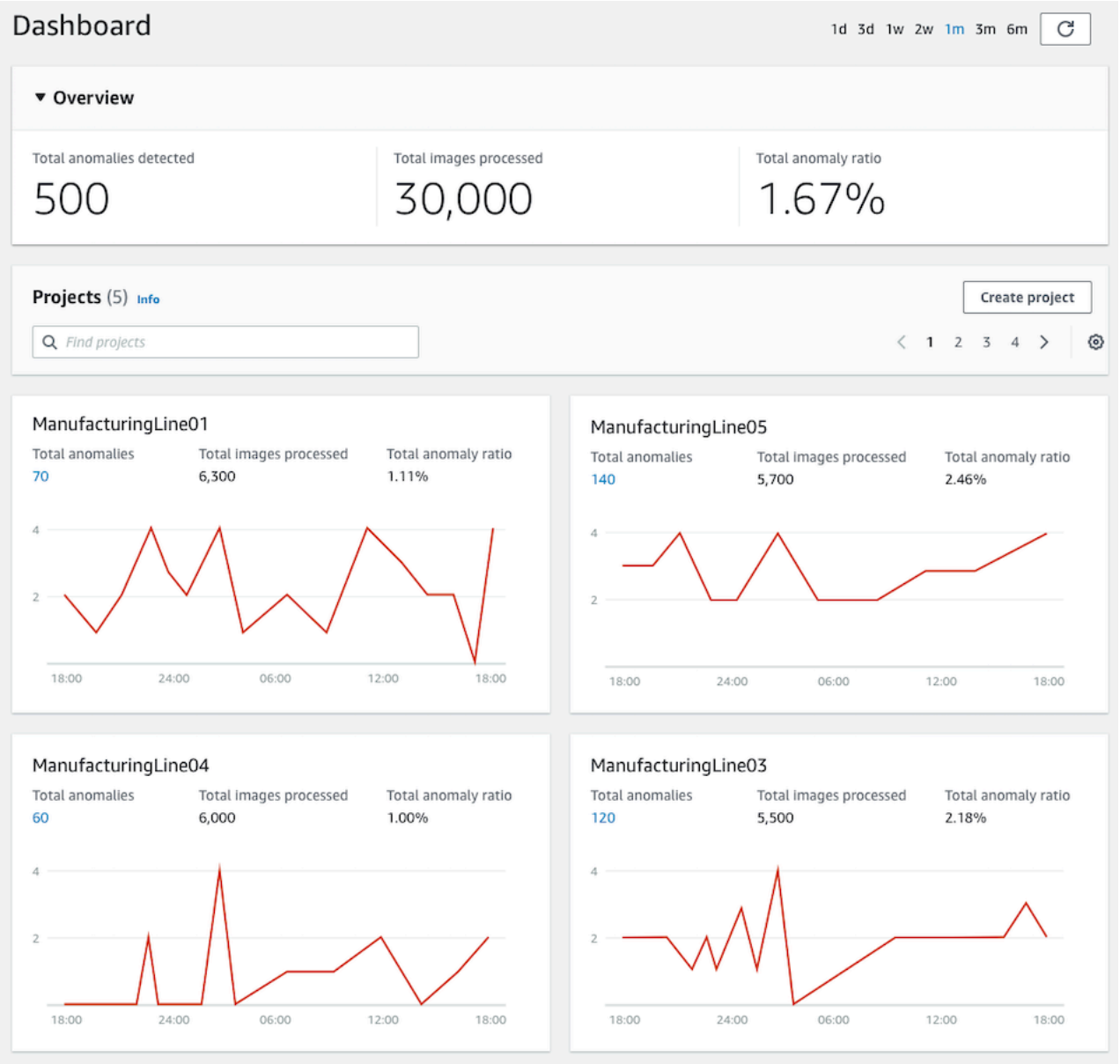
Le tableau de bord fournit un aperçu des indicateurs relatifs à vos projets Amazon Lookout for Vision, tels que le nombre total d'anomalies détectées la semaine dernière. Avec le tableau de bord, vous obtenez une vue d'ensemble de tous vos projets et une vue d'ensemble de chaque projet individuel. Vous pouvez choisir la chronologie sur laquelle les indicateurs sont affichés. Vous pouvez également utiliser le tableau de bord pour créer un nouveau projet.

La section Vue d'ensemble indique le nombre total de projets, le nombre total d'images et le nombre total d'images détectées par tous vos projets.

La section Projets présente les informations générales suivantes pour les projets individuels :

- Nombre total d'anomalies détectées.
- Le nombre total d'images traitées.
- Le taux d'anomalies total (c'est-à-dire le pourcentage d'images détectées présentant une anomalie).
- Un graphique montre les anomalies détectées au cours de la période choisie.

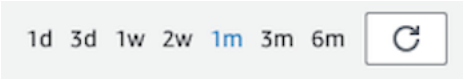
Vous pouvez également obtenir de plus amples informations sur un projet.



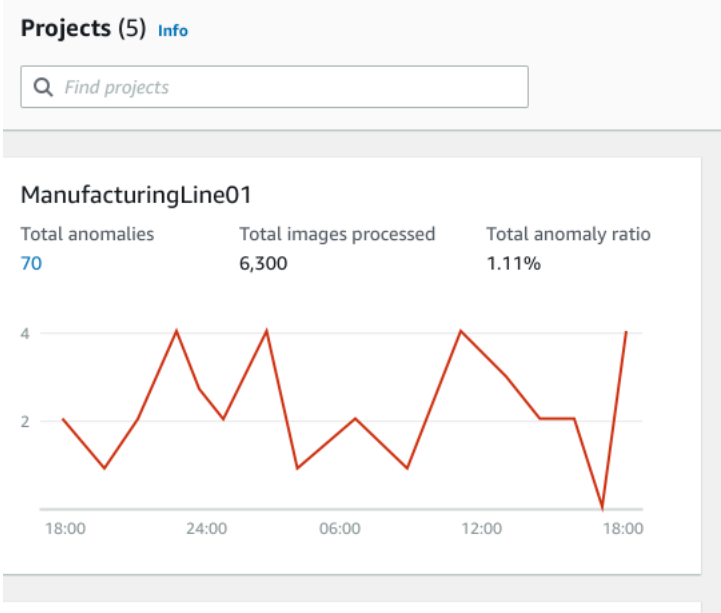
Pour utiliser votre tableau de bord

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Choisissez Démarrer.
3. Dans le panneau de navigation de gauche, choisissez Tableau de bord.
4. Pour consulter les statistiques sur une période donnée, procédez comme suit :
   a. Choisissez la période dans le coin supérieur droit du tableau de bord.

b. Cliquez sur le bouton d'actualisation pour afficher le tableau de bord avec la nouvelle chronologie.



5. Pour obtenir plus de détails sur un projet, choisissez le nom du projet dans la section Projets (par exemple,`ManufacturingLine01`).



6. Pour créer un projet, choisissez Créer un projet dans la section Projets.

Gestion de vos ressources Amazon Lookout for Vision

Vous pouvez gérer vos ressources Amazon Lookout for Vision à l'aide de la console ou AWS du SDK. Amazon Lookout for Vision propose les ressources suivantes :

- Projets
- Jeux de données
- Modèles
- Détections d'essais

> **Note**
>
> Vous ne pouvez pas supprimer une tâche de détection d'essai. De plus, vous ne pouvez pas gérer les détections d'essai à l'aide du AWS SDK.

Rubriques

- Afficher vos projets
- Suppression d’un projet
- Afficher vos ensembles de données
- Ajouter des images à votre jeu de données
- Supprimer des images de votre jeu de données
- Suppression d’un jeu de données
- Exportation de jeux de données depuis un projet (SDK)
- Visualisation de vos modèles
- Suppression d’un modèle
- Modèles de balisage
- Afficher les tâches de détection de vos essais

Afficher vos projets

Vous pouvez obtenir une liste des projets Amazon Lookout for Vision et des informations sur les différents projets depuis la console ou à l'aide AWS du SDK.

> **Note**
>
> La liste des projets est finalement cohérente. Si vous créez ou supprimez un projet, il se peut que vous deviez attendre un moment avant que la liste des projets ne soit mise à jour.

Afficher vos projets (console)

Suivez les étapes de la procédure suivante pour afficher vos projets dans la console.

Pour consulter vos projets

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Choisissez Démarrer.
3. Dans le volet de navigation de gauche, choisissez Projets. La vue des projets s'affiche.
4. Choisissez un nom de projet pour voir les détails du projet.

Afficher vos projets (SDK)

Un projet gère les ensembles de données et les modèles pour un seul cas d'utilisation. Par exemple, détecter des anomalies dans des pièces de machines. L'exemple suivant appelle `ListProjects` à obtenir la liste de vos projets.

Pour consulter vos projets (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 afficher vos projets.

   CLI

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

   ```
   aws lookoutvision list-projects \
     --profile lookoutvision-access
   ```

Utilisez la `describe-project` commande pour obtenir des informations sur un projet.

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

```
aws lookoutvision describe-project --project-name project_name \
  --profile lookoutvision-access
```

Python

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

```
    @staticmethod
    def list_projects(lookoutvision_client):
        """
        Lists information about the projects that are in in your AWS account
        and in the current AWS Region.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        """
        try:
            response = lookoutvision_client.list_projects()
            for project in response["Projects"]:
                print("Project: " + project["ProjectName"])
                print("\tARN: " + project["ProjectArn"])
                print("\tCreated: " + str(["CreationTimestamp"]))
                print("Datasets")
                project_description = lookoutvision_client.describe_project(
                    ProjectName=project["ProjectName"]
                )
                if not project_description["ProjectDescription"]["Datasets"]:
                    print("\tNo datasets")
                else:
                    for dataset in project_description["ProjectDescription"][
                        "Datasets"
                    ]:
                        print(f"\ttype: {dataset['DatasetType']}")
                        print(f"\tStatus: {dataset['StatusMessage']}")

                print("Models")
                response_models = lookoutvision_client.list_models(
                    ProjectName=project["ProjectName"]
```

```
                )
                if not response_models["Models"]:
                    print("\tNo models")
                else:
                    for model in response_models["Models"]:
                        Models.describe_model(
                            lookoutvision_client,
                            project["ProjectName"],
                            model["ModelVersion"],
                        )


 print("-----------------------------------------------------------\n")
            print("Done!")
        except ClientError:
            logger.exception("Problem listing projects.")
            raise
```

Java V2

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

```
/**
 * Lists the Amazon Lookout for Vision projects in the current AWS account and
 AWS
 * Region.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that you want to create.
 * @return List<ProjectMetadata> Metadata for each project.
 */
public static List<ProjectMetadata> listProjects(LookoutVisionClient lfvClient)
                throws LookoutVisionException {

        logger.log(Level.INFO, "Getting projects:");
        ListProjectsRequest listProjectsRequest = ListProjectsRequest.builder()
                        .maxResults(100)
                        .build();

        List<ProjectMetadata> projectMetadata = new ArrayList<>();
```

```
        ListProjectsIterable projects =
 lfvClient.listProjectsPaginator(listProjectsRequest);

        projects.stream().flatMap(r -> r.projects().stream())
                        .forEach(project -> {
                                projectMetadata.add(project);
                                logger.log(Level.INFO, project.projectName());
                        });

        logger.log(Level.INFO, "Finished getting projects.");

        return projectMetadata;

}
```

Suppression d’un projet

Vous pouvez supprimer un projet depuis la page d'affichage des projets de la console ou en utilisant cette `DeleteProject` opération.

Les images référencées par les ensembles de données d'un projet ne sont pas supprimées.

Supprimer un projet (console)

Pour supprimer un projet, procédez comme suit. Si vous utilisez la procédure de console, les versions du modèle et les jeux de données associés sont supprimés pour vous.

Pour supprimer un projet

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Choisissez Démarrer.
3. Dans le volet de navigation de gauche, choisissez Projets.
4. Sur la page Projets, sélectionnez le projet que vous souhaitez supprimer.
5. En haut de la page, choisissez Supprimer.
6. Dans la boîte de dialogue Supprimer, entrez Supprimer pour confirmer que vous souhaitez supprimer le projet.
7. Si nécessaire, choisissez de supprimer tous les ensembles de données et modèles associés.

8. Choisissez Supprimer le projet.

Suppression d’un projet (kit SDK)

Vous supprimez un projet Amazon Lookout for Vision en DeleteProjectappelant et en fournissant le nom du projet que vous souhaitez supprimer.

Avant de pouvoir supprimer un projet, vous devez d'abord supprimer tous les modèles du projet. Pour de plus amples informations, veuillez consulter Suppression d’un modèle (kit SDK). Vous devez également supprimer les jeux de données associés au modèle. Pour de plus amples informations, veuillez consulter 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 `DeleteProject` n'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-name` par le nom du projet que vous souhaitez supprimer.

   ```
   aws lookoutvision delete-project --project-name project_name \
     --profile lookoutvision-access
   ```

   Python

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

   ```
       @staticmethod
       def delete_project(lookoutvision_client, project_name):
           """
           Deletes a Lookout for Vision Model
   ```

```
        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that you want to delete.
        """
        try:
            logger.info("Deleting project: %s", project_name)
            response =
 lookoutvision_client.delete_project(ProjectName=project_name)
            logger.info("Deleted project ARN: %s ", response["ProjectArn"])
        except ClientError as err:
            logger.exception("Couldn't delete project %s.", project_name)
            raise
```

Java V2

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

```
/**
 * Deletes an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that you want to create.
 * @return String The ARN of the deleted project.
 */
public static String deleteProject(LookoutVisionClient lfvClient, String
 projectName)
                throws LookoutVisionException {

        logger.log(Level.INFO, "Deleting project: {0}", projectName);

        DeleteProjectRequest deleteProjectRequest =
 DeleteProjectRequest.builder()
                        .projectName(projectName)
                        .build();

        DeleteProjectResponse response =
 lfvClient.deleteProject(deleteProjectRequest);

        logger.log(Level.INFO, "Deleted project: {0} ARN: {1}",
                        new Object[] { projectName, response.projectArn() });

        return response.projectArn();
```

```
}
```

Afficher vos ensembles de données

Un projet peut comporter un seul jeu de données utilisé pour entraîner et tester votre modèle. Vous pouvez également avoir des ensembles de données d'entraînement et de test distincts. Vous pouvez utiliser la console pour consulter vos ensembles de données. Vous pouvez également utiliser l'`DescribeDataset`opération pour obtenir des informations sur un ensemble de données (entraînement ou test).

Afficher les ensembles de données d'un projet (console)

Suivez les étapes de la procédure suivante pour afficher les ensembles de données de votre projet dans la console.

Pour consulter vos ensembles de données (console)

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Choisissez Démarrer.
3. Dans le volet de navigation de gauche, choisissez Projets.
4. Sur la page Projets, sélectionnez le projet qui contient les ensembles de données que vous souhaitez consulter.
5. Dans le volet de navigation de gauche, choisissez Dataset pour afficher les détails du jeu de données. Si vous disposez d'un ensemble de données d'entraînement et d'un ensemble de données de test, un onglet est affiché pour chaque ensemble de données.

Afficher les ensembles de données d'un projet (SDK)

Vous pouvez utiliser cette `DescribeDataset` opération pour obtenir des informations sur le jeu de données d'entraînement ou de test associé à un projet.

Pour afficher vos ensembles de données (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 afficher un ensemble de données.

CLI

Remplacez les valeurs suivantes :

- `project-name`au nom du projet qui contient le modèle que vous souhaitez visualiser.
- `dataset-type`au type de jeu de données que vous souhaitez afficher (`train`ou`test`).

```
aws lookoutvision describe-dataset --project-name project name\
  --dataset-type train or test \
  --profile lookoutvision-access
```

Python

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

```
    @staticmethod
    def describe_dataset(lookoutvision_client, project_name, dataset_type):
        """
        Gets information about a Lookout for Vision dataset.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the dataset
 that
                             you want to describe.
        :param dataset_type: The type (train or test) of the dataset that you
 want
                             to describe.
        """
        try:
            response = lookoutvision_client.describe_dataset(
                ProjectName=project_name, DatasetType=dataset_type
            )
            print(f"Name: {response['DatasetDescription']['ProjectName']}")
            print(f"Type: {response['DatasetDescription']['DatasetType']}")
            print(f"Status: {response['DatasetDescription']['Status']}")
            print(f"Message: {response['DatasetDescription']['StatusMessage']}")
            print(f"Images: {response['DatasetDescription']['ImageStats']
['Total']}")
```

```
            print(f"Labeled: {response['DatasetDescription']['ImageStats']
['Labeled']}")
            print(f"Normal: {response['DatasetDescription']['ImageStats']
['Normal']}")
            print(f"Anomaly: {response['DatasetDescription']['ImageStats']
['Anomaly']}")
        except ClientError:
            logger.exception("Service error: problem listing datasets.")
            raise
        print("Done.")
```

Java V2

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

```
/**
 * Gets the description for a Amazon Lookout for Vision dataset.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project in which you want to describe a
 *                    dataset.
 * @param datasetType The type of the dataset that you want to describe (train
 *                    or test).
 * @return DatasetDescription A description of the dataset.
 */
public static DatasetDescription describeDataset(LookoutVisionClient lfvClient,
                String projectName,
                String datasetType) throws LookoutVisionException {

        logger.log(Level.INFO, "Describing {0} dataset for project {1}",
                        new Object[] { datasetType, projectName });

        DescribeDatasetRequest describeDatasetRequest =
 DescribeDatasetRequest.builder()
                        .projectName(projectName)
                        .datasetType(datasetType)
                        .build();

        DescribeDatasetResponse describeDatasetResponse =
 lfvClient.describeDataset(describeDatasetRequest);
```

```
        DatasetDescription datasetDescription =
 describeDatasetResponse.datasetDescription();

        logger.log(Level.INFO, "Project: {0}\n"
                        + "Created: {1}\n"
                        + "Type: {2}\n"
                        + "Total: {3}\n"
                        + "Labeled: {4}\n"
                        + "Normal: {5}\n"
                        + "Anomalous: {6}\n",
                        new Object[] {
                                        datasetDescription.projectName(),
                                        datasetDescription.creationTimestamp(),
                                        datasetDescription.datasetType(),

 datasetDescription.imageStats().total().toString(),

 datasetDescription.imageStats().labeled().toString(),

 datasetDescription.imageStats().normal().toString(),

 datasetDescription.imageStats().anomaly().toString(),
                        });

        return datasetDescription;

}
```

Ajouter des images à votre jeu de données

Après avoir créé un jeu de données, vous souhaiterez peut-être y ajouter d'autres images. Par exemple, si l'évaluation du modèle indique un modèle médiocre, vous pouvez améliorer la qualité de votre modèle en ajoutant des images supplémentaires. Si vous avez créé un jeu de données de test, l'ajout d'images supplémentaires peut améliorer la précision des indicateurs de performance de votre modèle.

Réentraînez votre modèle après avoir mis à jour vos ensembles de données.

Rubriques

- Ajout d’autres images
- Ajout d’autres images (kit SDK)

Ajout d’autres images

Vous pouvez ajouter d'autres images à vos ensembles de données en téléchargeant des images depuis votre ordinateur local. Pour ajouter d'autres images étiquetées avec le SDK, utilisez l'UpdateDatasetEntriesopération.

Pour ajouter d’autres images à votre jeu de données (console)

1. Choisissez Actions et sélectionnez le jeu de données auquel vous souhaitez ajouter des images.
2. 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.
3. Choisissez Charger des images.
4. Sélectionnez Enregistrer les modifications.

Lorsque vous avez terminé d'ajouter d'autres images, vous devez les étiqueter afin qu'elles puissent être utilisées pour entraîner le modèle. Pour de plus amples informations, veuillez consulter Classification des images (console).

Ajout d’autres images (kit SDK)

Pour ajouter d'autres images étiquetées avec le SDK, utilisez l'UpdateDatasetEntriesopération. Vous fournissez un fichier manifeste contenant les images que vous souhaitez ajouter. Vous pouvez également mettre à jour les images existantes en spécifiant l'image dans le `source-ref` champ de la ligne JSON du fichier manifeste. Pour de plus amples informations, veuillez consulter Création d’un fichier manifeste.

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 l'exemple de code suivant pour ajouter d'autres images à un ensemble de données.

   CLI

   Remplacez les valeurs suivantes :

- `project-name`au nom du projet qui contient le jeu de données que vous souhaitez mettre à jour.
- `dataset-type`au type de jeu de données que vous souhaitez mettre à jour (`train`ou`test`).
- `changes`à l'emplacement du fichier manifeste contenant les mises à jour de l'ensemble de données.

```
aws lookoutvision update-dataset-entries\
    --project-name project\
    --dataset-type train or test\
    --changes fileb://manifest file \
    --profile lookoutvision-access
```

Python

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

```
   @staticmethod
   def update_dataset_entries(lookoutvision_client, project_name, dataset_type,
updates_file):
       """
       Adds dataset entries to an Amazon Lookout for Vision dataset.
       :param lookoutvision_client: The Amazon Rekognition Custom Labels Boto3
client.
       :param project_name: The project that contains the dataset that you want
to update.
       :param dataset_type: The type of the dataset that you want to update
(train or test).
       :param updates_file: The manifest file of JSON Lines that contains the
updates.
       """

       try:
           status = ""
           status_message = ""
           manifest_file = ""

           # Update dataset entries
```

```
            logger.info(f"""Updating {dataset_type} dataset for project
 {project_name}
with entries from {updates_file}.""")

            with open(updates_file) as f:
                manifest_file = f.read()

            lookoutvision_client.update_dataset_entries(
                ProjectName=project_name,
                DatasetType=dataset_type,
                Changes=manifest_file,
            )

            finished = False
            while finished == False:

                dataset =
 lookoutvision_client.describe_dataset(ProjectName=project_name,

 DatasetType=dataset_type)

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

                if status == "UPDATE_IN_PROGRESS":
                    logger.info(
                        (f"Updating {dataset_type} dataset for project
 {project_name}."))
                    time.sleep(5)
                    continue

                if status == "UPDATE_FAILED_ROLLBACK_IN_PROGRESS":
                    logger.info(
                        (f"Update failed, rolling back {dataset_type} dataset
 for project {project_name}."))
                    time.sleep(5)
                    continue

                if status == "UPDATE_COMPLETE":
                    logger.info(
                        f"Dataset updated: {status} : {status_message} :
 {dataset_type} dataset for project {project_name}.")
                    finished = True
                    continue
```

```
                if status == "UPDATE_FAILED_ROLLBACK_COMPLETE":
                    logger.info(
                        f"Rollback complated after update failure: {status} :
{status_message} : {dataset_type} dataset for project {project_name}.")
                    finished = True
                    continue

                logger.exception(
                    f"Failed. Unexpected state for dataset update: {status} :
{status_message} : {dataset_type} dataset for project {project_name}.")
                raise Exception(
                    f"Failed. Unexpected state for dataset update: {status} :
{status_message} :{dataset_type} dataset for project {project_name}.")

            logger.info(f"Added entries to dataset.")

            return status, status_message

        except ClientError as err:
            logger.exception(
                f"Couldn't update dataset: {err.response['Error']['Message']}")
            raise
```

Java V2

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

```
/**
 * Updates an Amazon Lookout for Vision dataset from a manifest file.
 * Returns after Lookout for Vision updates the dataset.
 *
 * @param lfvClient    An Amazon Lookout for Vision client.
 * @param projectName  The name of the project in which you want to update a
 *                     dataset.
 * @param datasetType  The type of the dataset that you want to update (train or
 *                     test).
 * @param manifestFile The name and location of a local manifest file that you
 want to
 * use to update the dataset.
 * @return DatasetStatus The status of the updated dataset.
 */
```

```
public static DatasetStatus updateDatasetEntries(LookoutVisionClient lfvClient,
 String projectName,
                String datasetType, String updateFile) throws
 FileNotFoundException, LookoutVisionException,
                InterruptedException {

        logger.log(Level.INFO, "Updating {0} dataset for project {1}",
                        new Object[] { datasetType, projectName });

        InputStream sourceStream = new FileInputStream(updateFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        UpdateDatasetEntriesRequest updateDatasetEntriesRequest =
 UpdateDatasetEntriesRequest.builder()
                        .projectName(projectName)
                        .datasetType(datasetType)
                        .changes(sourceBytes)
                        .build();

        lfvClient.updateDatasetEntries(updateDatasetEntriesRequest);

        boolean finished = false;
        DatasetStatus status = null;

        // Wait until update completes.

        do {

                DescribeDatasetRequest describeDatasetRequest =
 DescribeDatasetRequest.builder()
                                .projectName(projectName)
                                .datasetType(datasetType)
                                .build();
                DescribeDatasetResponse describeDatasetResponse = lfvClient
                                .describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
 describeDatasetResponse.datasetDescription();

                status = datasetDescription.status();

                switch (status) {
```

```
                        case UPDATE_COMPLETE:
                                logger.log(Level.INFO, "{0} Dataset updated for
project {1}.",
                                                new Object[] { datasetType,
projectName });
                                finished = true;
                                break;

                        case UPDATE_IN_PROGRESS:

                                logger.log(Level.INFO, "{0} Dataset update for
project {1} in progress.",
                                                new Object[] { datasetType,
projectName });
                                TimeUnit.SECONDS.sleep(5);

                                break;

                        case UPDATE_FAILED_ROLLBACK_IN_PROGRESS:
                                logger.log(Level.SEVERE,
                                                "{0} Dataset update failed for
project {1}. Rolling back",
                                                new Object[] { datasetType,
projectName });

                                TimeUnit.SECONDS.sleep(5);

                                break;

                        case UPDATE_FAILED_ROLLBACK_COMPLETE:
                                logger.log(Level.SEVERE,
                                                "{0} Dataset update failed for
project {1}. Rollback completed.",
                                                new Object[] { datasetType,
projectName });
                                finished = true;
                                break;

                        default:
                                logger.log(Level.SEVERE,
                                                "{0} Dataset update failed for
project {1}. Unexpected error returned.",
                                                new Object[] { datasetType,
projectName });
```

```
                                finished = true;

                }

        } while (!finished);

        return status;

}
```

3. Répétez l'étape précédente et fournissez des valeurs pour l'autre type de jeu de données.

Supprimer des images de votre jeu de données

Vous ne pouvez pas supprimer d'images directement d'un jeu de données. Vous devez plutôt supprimer le jeu de données existant et créer un nouveau jeu de données sans les images que vous souhaitez supprimer. La méthode de suppression des images dépend de la manière dont vous les avez importées dans le jeu de données existant (fichier manifeste, compartiment Amazon S3 ou ordinateur local).

Vous pouvez également utiliser le AWS SDK pour supprimer des images. Cela est utile lors de la création d'un modèle de segmentation d'image sans fichier manifeste de segmentation d'image, ce qui évite de redessiner les masques d'image avec la console Amazon Lookout for Vision.

Rubriques

- Supprimer des images d'un jeu de données (console)
- Supprimer des images d'un jeu de données (SDK)

Supprimer des images d'un jeu de données (console)

Utilisez la procédure suivante pour supprimer des images d'un ensemble de données à l'aide de la console Amazon Lookout for Vision.

Pour supprimer des images d'un jeu de données (console)

1. Ouvrez la galerie de jeux de données du projet.

2. Notez le nom de chaque image que vous souhaitez supprimer.
3. Supprimez le jeu de données existant.
4. Effectuez l'une des actions suivantes :
   - Si vous avez créé l'ensemble de données avec un fichier manifeste, procédez comme suit :
     a. Dans un éditeur de texte, ouvrez le fichier manifeste que vous avez utilisé pour créer le jeu de données.
     b. Supprimez la ligne JSON pour chaque image que vous avez notée à l'étape 2. Vous pouvez identifier la ligne JSON d'une image en cochant le `source-ref` champ.
     c. Enregistrez le fichier manifeste.
     d. Créez un nouvel ensemble de données avec le fichier manifeste mis à jour.
   - Si vous avez créé le jeu de données à partir d'images importées depuis un compartiment Amazon S3, procédez comme suit :
     a. Supprimez les images que vous avez notées à l'étape 2 du compartiment Amazon S3.
     b. Créez un nouvel ensemble de données avec les images restantes dans le compartiment Amazon S3. Si vous classez les images par nom de dossier, il n'est pas nécessaire de les classer à l'étape suivante.
     c. Effectuez l'une des actions suivantes :
        - Si vous créez un modèle de classification d'images, classez chaque image non étiquetée.
        - Si vous créez un modèle de segmentation d'image, classez et segmentez chaque image non étiquetée.
   - Si vous avez créé le jeu de données à partir d'images importées depuis un ordinateur local, procédez comme suit :
     a. Sur votre ordinateur, créez un dossier contenant les images que vous souhaitez utiliser. N'incluez pas les images que vous souhaitez supprimer du jeu de données. Pour de plus amples informations, veuillez consulter Création d'un jeu de données à partir d'images stockées sur votre ordinateur local.
     b. Créez le jeu de données avec les images dans le dossier que vous avez créé à l'étape 4.a.
     c. Effectuez l'une des actions suivantes :

- Si vous créez un modèle de classification d'images, classez chaque image non étiquetée.
- Si vous créez un modèle de segmentation d'image, classez et segmentez chaque image non étiquetée.

5. Entraînez le modèle.

Supprimer des images d'un jeu de données (SDK)

Vous pouvez utiliser le AWS SDK pour supprimer des images d'un jeu de données.

Pour supprimer des images d'un jeu de données (SDK)

1. Ouvrez la galerie de jeux de données du projet.
2. Notez le nom de chaque image que vous souhaitez supprimer.
3. Exportez les lignes JSON de l'ensemble de données à l'aide de l'ListDatasetEntriesopération.
4. Créez un fichier manifeste avec les lignes JSON exportées.
5. Dans un éditeur de texte, ouvrez le fichier manifeste.
6. Supprimez la ligne JSON pour chaque image que vous avez notée à l'étape 2. Vous pouvez identifier la ligne JSON d'une image en cochant le `source-ref` champ.
7. Enregistrez le fichier manifeste.
8. Supprimez le jeu de données existant.
9. Créez un nouvel ensemble de données avec le fichier manifeste mis à jour.
10. Entraînez le modèle.

Suppression d’un jeu de données

Vous pouvez supprimer un ensemble de données d'un projet à l'aide de la console ou de l'`DeleteDataset`opération. Les images référencées par un ensemble de données ne sont pas supprimées. Si vous supprimez le jeu de données de test d'un projet comportant un ensemble de données d'entraînement et un ensemble de données de test, le projet redevient un projet de jeu de données unique ; le jeu de données restant est divisé pendant l'entraînement pour créer un ensemble de données d'entraînement et un ensemble de données de test. Si vous supprimez le jeu de données d'entraînement, vous ne pouvez pas entraîner de modèle dans le projet tant que vous n'avez pas créé un nouveau jeu de données d'entraînement.

Supprimer un ensemble de données (console)

Procédez comme indiqué dans la procédure suivante pour supprimer un ensemble de données. Si vous supprimez tous les ensembles de données d'un projet, la page Créer un jeu de données s'affiche.

Pour supprimer un ensemble de données (console)

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Choisissez Démarrer.
3. Dans le volet de navigation de gauche, choisissez Projets.
4. Sur la page Projets, sélectionnez le projet qui contient le jeu de données que vous souhaitez supprimer.
5. Dans le volet de navigation de gauche, choisissez Dataset.
6. Choisissez Actions, puis sélectionnez le jeu de données que vous souhaitez supprimer.
7. Dans la boîte de dialogue Supprimer, entrez delete pour confirmer que vous souhaitez supprimer le jeu de données.
8. Choisissez Supprimer le jeu de données d'entraînement ou Supprimer le jeu de données de test pour supprimer le jeu de données.

Suppression d’un jeu de données (kit SDK)

Utilisez cette `DeleteDataset` opération pour supprimer un ensemble de données.

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 l'exemple de code suivant pour supprimer un modèle.

   CLI

   Modifiez la valeur de ce qui suit

   - `project-name`au nom du projet qui contient le modèle que vous souhaitez supprimer.

- `dataset-type`à l'un `train` ou l'autre `test`, selon le jeu de données que vous souhaitez supprimer. Si vous n'avez qu'un seul projet de jeu de données, spécifiez `train` de supprimer le jeu de données.

```
aws lookoutvision delete-dataset --project-name project name\
  --dataset-type dataset type \
  --profile lookoutvision-access
```

Python

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

```
    @staticmethod
    def delete_dataset(lookoutvision_client, project_name, dataset_type):
        """
        Deletes a Lookout for Vision dataset

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the dataset
that
                             you want to delete.
        :param dataset_type: The type (train or test) of the dataset that you
                             want to delete.
        """
        try:
            logger.info(
                "Deleting the %s dataset for project %s.", dataset_type,
project_name
            )
            lookoutvision_client.delete_dataset(
                ProjectName=project_name, DatasetType=dataset_type
            )
            logger.info("Dataset deleted.")
        except ClientError:
            logger.exception("Service error: Couldn't delete dataset.")
            raise
```

Java V2

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

```
/**
 * Deletes the train or test dataset in an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project in which you want to delete a
 *                    dataset.
 * @param datasetType The type of the dataset that you want to delete (train or
 *                    test).
 * @return Nothing.
 */
public static void deleteDataset(LookoutVisionClient lfvClient, String
 projectName, String datasetType)
                throws LookoutVisionException {

        logger.log(Level.INFO, "Deleting {0} dataset for project {1}",
                        new Object[] { datasetType, projectName });

        DeleteDatasetRequest deleteDatasetRequest =
 DeleteDatasetRequest.builder()
                        .projectName(projectName)
                        .datasetType(datasetType)
                        .build();

        lfvClient.deleteDataset(deleteDatasetRequest);

        logger.log(Level.INFO, "Deleted {0} dataset for project {1}",
                        new Object[] { datasetType, projectName });

}
```

Exportation de jeux de données depuis un projet (SDK)

Vous pouvez utiliser le AWS SDK pour exporter des ensembles de données d'un projet Amazon Lookout for Vision vers un emplacement de compartiment Amazon S3.

En exportant un jeu de données, vous pouvez effectuer des tâches telles que la création d'un projet Lookout for Vision avec une copie des ensembles de données d'un projet source. Vous pouvez également créer un instantané des jeux de données utilisés pour une version spécifique d'un modèle.

Le code Python de cette procédure exporte le jeu de données d'entraînement (manifeste et images du jeu de données) d'un projet vers un emplacement Amazon S3 de destination que vous spécifiez. S'il est présent dans le projet, le code exporte également le manifeste du jeu de données de test et les images du jeu de données. La destination peut se trouver dans le même compartiment Amazon S3 que le projet source ou dans un autre compartiment Amazon S3. Le code utilise l'ListDatasetEntriesopération pour obtenir les fichiers manifestes de l'ensemble de données. Les opérations Amazon S3 copient les images du jeu de données et les fichiers manifestes mis à jour vers l'emplacement Amazon S3 de destination.

Cette procédure montre comment exporter les ensembles de données d'un projet. Il montre également comment créer un nouveau projet avec les ensembles de données exportés.

Pour exporter les ensembles de données d'un projet (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. Déterminez le chemin Amazon S3 de destination pour l'exportation du jeu de données. Assurez-vous que la destination se trouve dans une AWS région prise en charge par Amazon Lookout for Vision. Pour créer un nouveau compartiment Amazon S3, consultez Création d'un compartiment.
3. Assurez-vous que l'utilisateur dispose des autorisations d'accès au chemin Amazon S3 de destination pour l'exportation du jeu de données et aux emplacements S3 des fichiers image dans les ensembles de données du projet source. Vous pouvez appliquer la politique suivante, qui part du principe que les fichiers images peuvent se trouver n'importe où. *bucket/path*Remplacez-le par le compartiment et le chemin de destination pour l'exportation du jeu de données.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PutExports",
            "Effect": "Allow",
            "Action": [
                "S3:PutObjectTagging",
                "S3:PutObject"
```

```
            ],
            "Resource": "arn:aws:s3:::bucket/path/*"
        },
        {
            "Sid": "GetSourceRefs",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion"
            ],
            "Resource": "*"
        }
    ]
}
```

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.

4. Enregistrez le code suivant dans un fichier nommé `dataset_export.py`.

```
"""
Purpose

Shows how to export the datasets (manifest files and images)
from an Amazon Lookout for Vision project to a new Amazon
```

```
S3 location.
"""

import argparse
import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def copy_file(s3_resource, source_file, destination_file):
    """
    Copies a file from a source Amazon S3 folder to a destination
    Amazon S3 folder.
    The destination can be in a different S3 bucket.
    :param s3: An Amazon S3 Boto3 resource.
    :param source_file: The Amazon S3 path to the source file.
    :param destination_file: The destination Amazon S3 path for
    the copy operation.
    """

    source_bucket, source_key = source_file.replace("s3://", "").split("/", 1)
    destination_bucket, destination_key = destination_file.replace("s3://",
 "").split(
        "/", 1
    )

    try:
        bucket = s3_resource.Bucket(destination_bucket)
        dest_object = bucket.Object(destination_key)
        dest_object.copy_from(CopySource={"Bucket": source_bucket, "Key":
 source_key})
        dest_object.wait_until_exists()
        logger.info("Copied %s to %s", source_file, destination_file)
    except ClientError as error:
        if error.response["Error"]["Code"] == "404":
            error_message = (
                f"Failed to copy {source_file} to "
                f"{destination_file}. : {error.response['Error']['Message']}"
            )
            logger.warning(error_message)
```

```
            error.response["Error"]["Message"] = error_message
        raise


def upload_manifest_file(s3_resource, manifest_file, destination):
    """
    Uploads a manifest file to a destination Amazon S3 folder.
    :param s3: An Amazon S3 Boto3 resource.
    :param manifest_file: The manifest file that you want to upload.
    :destination: The Amazon S3 folder location to upload the manifest
    file to.
    """

    destination_bucket, destination_key = destination.replace("s3://",
 "").split("/", 1)

    bucket = s3_resource.Bucket(destination_bucket)

    put_data = open(manifest_file, "rb")
    obj = bucket.Object(destination_key + manifest_file)

    try:
        obj.put(Body=put_data)
        obj.wait_until_exists()
        logger.info("Put manifest file '%s' to bucket '%s'.", obj.key,
 obj.bucket_name)
    except ClientError:
        logger.exception(
            "Couldn't put manifest file '%s' to bucket '%s'.", obj.key,
 obj.bucket_name
        )
        raise
    finally:
        if getattr(put_data, "close", None):
            put_data.close()


def get_dataset_types(lookoutvision_client, project):
    """
    Determines the types of the datasets (train or test) in an
    Amazon Lookout for Vision project.
    :param lookoutvision_client: A Lookout for Vision Boto3 client.
    :param project: The Lookout for Vision project that you want to check.
    :return: The dataset types in the project.
```

```
    """

    try:
        response = lookoutvision_client.describe_project(ProjectName=project)

        datasets = []

        for dataset in response["ProjectDescription"]["Datasets"]:
            if dataset["Status"] in ("CREATE_COMPLETE", "UPDATE_COMPLETE"):
                datasets.append(dataset["DatasetType"])
        return datasets

    except lookoutvision_client.exceptions.ResourceNotFoundException:
        logger.exception("Project %s not found.", project)
        raise


def process_json_line(s3_resource, entry, dataset_type, destination):
    """
    Creates a JSON line for a new manifest file, copies image and mask to
    destination.
    :param s3_resource: An Amazon S3 Boto3 resource.
    :param entry: A JSON line from the manifest file.
    :param dataset_type: The type (train or test) of the dataset that
    you want to create the manifest file for.
    :param destination: The destination Amazon S3 folder for the manifest
    file and dataset images.
    :return: A JSON line with details for the destination location.
    """
    entry_json = json.loads(entry)

    print(f"source: {entry_json['source-ref']}")

    # Use existing folder paths to ensure console added image names don't clash.
    bucket, key = entry_json["source-ref"].replace("s3://", "").split("/", 1)
    logger.info("Source location: %s/%s", bucket, key)

    destination_image_location = destination + dataset_type + "/images/" + key

    copy_file(s3_resource, entry_json["source-ref"], destination_image_location)

    # Update JSON for writing.
    entry_json["source-ref"] = destination_image_location
```

```
    if "anomaly-mask-ref" in entry_json:
        source_anomaly_ref = entry_json["anomaly-mask-ref"]
        mask_bucket, mask_key = source_anomaly_ref.replace("s3://", "").split("/",
 1)

        destination_mask_location = destination + dataset_type + "/masks/" +
 mask_key
        entry_json["anomaly-mask-ref"] = destination_mask_location

        copy_file(s3_resource, source_anomaly_ref, entry_json["anomaly-mask-ref"])

    return entry_json


def write_manifest_file(
    lookoutvision_client, s3_resource, project, dataset_type, destination
):
    """
    Creates a manifest file for a dataset. Copies the manifest file and
    dataset images (and masks, if present) to the specified Amazon S3 destination.
    :param lookoutvision_client: A Lookout for Vision Boto3 client.
    :param project: The Lookout for Vision project that you want to use.
    :param dataset_type: The type (train or test) of the dataset that
    you want to create the manifest file for.
    :param destination: The destination Amazon S3 folder for the manifest file
    and dataset images.
    """

    try:
        # Create a reusable Paginator
        paginator = lookoutvision_client.get_paginator("list_dataset_entries")

        # Create a PageIterator from the Paginator
        page_iterator = paginator.paginate(
            ProjectName=project,
            DatasetType=dataset_type,
            PaginationConfig={"PageSize": 100},
        )

        output_manifest_file = dataset_type + ".manifest"

        # Create manifest file then upload to Amazon S3 with images.
        with open(output_manifest_file, "w", encoding="utf-8") as manifest_file:
            for page in page_iterator:
```

```
                for entry in page["DatasetEntries"]:
                    try:
                        entry_json = process_json_line(
                            s3_resource, entry, dataset_type, destination
                        )

                        manifest_file.write(json.dumps(entry_json) + "\n")

                    except ClientError as error:
                        if error.response["Error"]["Code"] == "404":
                            print(error.response["Error"]["Message"])
                            print(f"Excluded JSON line: {entry}")
                        else:
                            raise
        upload_manifest_file(
            s3_resource, output_manifest_file, destination + "datasets/"
        )

    except ClientError:
        logger.exception("Problem getting dataset_entries")
        raise


def export_datasets(lookoutvision_client, s3_resource, project, destination):
    """
    Exports the datasets from an Amazon Lookout for Vision project to a specified
    Amazon S3 destination.
    :param project: The Lookout for Vision project that you want to use.
    :param destination: The destination Amazon S3 folder for the exported datasets.
    """
    # Add trailing backslash, if missing.
    destination = destination if destination[-1] == "/" else destination + "/"

    print(f"Exporting project {project} datasets to {destination}.")

    # Get each dataset and export to destination.

    dataset_types = get_dataset_types(lookoutvision_client, project)
    for dataset in dataset_types:
        logger.info("Copying %s dataset to %s.", dataset, destination)

        write_manifest_file(
            lookoutvision_client, s3_resource, project, dataset, destination
        )
```

```
    print("Exported dataset locations")
    for dataset in dataset_types:
        print(f"   {dataset}: {destination}datasets/{dataset}.manifest")

    print("Done.")


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

    parser.add_argument("project", help="The project that contains the dataset.")
    parser.add_argument("destination", help="The destination Amazon S3 folder.")


def main():
    """
    Exports the datasets from an Amazon Lookout for Vision project to a
    destination Amazon S3 location.
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
    add_arguments(parser)

    args = parser.parse_args()

    try:
        session = boto3.Session(profile_name="lookoutvision-access")
        lookoutvision_client = session.client("lookoutvision")
        s3_resource = session.resource("s3")

        export_datasets(
            lookoutvision_client, s3_resource, args.project, args.destination
        )
    except ClientError as err:
        logger.exception(err)
        print(f"Failed: {format(err)}")


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

5. Exécutez le code. Fournissez les arguments de ligne de commande suivants :

   - projet : nom du projet source qui contient les ensembles de données que vous souhaitez exporter.
   - destination — Le chemin Amazon S3 de destination pour les ensembles de données.

   Par exemple, `python dataset_export.py myproject s3://bucket/path/`
6. Notez les emplacements des fichiers manifestes affichés par le code. Vous en aurez besoin à l'étape 8.
7. Créez un nouveau projet Lookout for Vision avec un ensemble de données exporté en suivant les instructions Création de votre projet de.
8. Effectuez l'une des actions suivantes :
   - Utilisez la console Lookout for Vision pour créer des ensembles de données pour votre nouveau projet en suivant les instructions de. Création d'un ensemble de données avec un fichier manifeste (console) Vous n'avez pas besoin de suivre les étapes 1 à 6.

     Pour l'étape 12, procédez comme suit :

     a. Si le projet source possède un ensemble de données de test, choisissez Séparer les ensembles de données d'entraînement et de test, sinon choisissez un ensemble de données unique.

     b. Pour l'emplacement du fichier .manifest, entrez l'emplacement du fichier manifeste approprié (train ou test) que vous avez noté à l'étape 6.
   - Utilisez l'CreateDatasetopération pour créer des ensembles de données pour votre nouveau projet en utilisant le code à Création d'un ensemble de données à l'aide d'un fichier manifeste (SDK) l'adresse. Pour le `manifest_file` paramètre, utilisez l'emplacement du fichier manifeste indiqué à l'étape 6. Si le projet source possède un jeu de données de test, réutilisez le code pour créer l'ensemble de données de test.
9. Si vous êtes prêt, entraînez le modèle en suivant les instructions surEntraînement de votre modèle.

Visualisation de vos modèles

Un projet peut comporter plusieurs versions d'un modèle. Vous pouvez utiliser la console pour visualiser les modèles d'un projet. Vous pouvez également utiliser l'opération `ListModels`.

> **Note**
>
> La liste des modèles est finalement cohérente. Si vous créez un modèle, il se peut que vous deviez attendre un peu avant que la liste des modèles ne soit mise à jour.

Affichage de vos modèles (console)

Suivez les étapes de la procédure suivante pour afficher les modèles de votre projet dans la console.

Pour afficher vos modèles (console)

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Choisissez Démarrer.
3. Dans le volet de navigation de gauche, choisissez Projets.
4. Sur la page Projets, sélectionnez le projet contenant les modèles que vous souhaitez consulter.
5. Dans le volet de navigation de gauche, choisissez Modèles, puis affichez les détails du modèle.

Affichage de vos modèles (SDK)

Pour voir les versions d'un modèle, vous utilisez l'`ListModels`opération. Pour obtenir des informations sur une version de modèle spécifique, utilisez l'`DescribeModel`opération. L'exemple suivant répertorie toutes les versions de modèle d'un projet, puis affiche les informations de performance et de configuration de sortie pour les différentes versions de modèle.

Pour afficher vos modèles (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 vos modèles et obtenir des informations sur un modèle.

CLI

Utilisez la `list-models` commande pour répertorier les modèles d'un projet.

Modifiez la valeur suivante :

- `project-name`au nom du projet qui contient le modèle que vous souhaitez visualiser.

```
aws lookoutvision list-models --project-name project name \
  --profile lookoutvision-access
```

Utilisez la `describe-model` commande pour obtenir des informations sur un modèle. Remplacez les valeurs suivantes :

- `project-name`au nom du projet qui contient le modèle que vous souhaitez visualiser.
- `model-version`à la version du modèle que vous souhaitez décrire.

```
aws lookoutvision describe-model --project-name project name\
    --model-version model version \
   --profile lookoutvision-access
```

Python

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

```
    @staticmethod
    def describe_models(lookoutvision_client, project_name):
        """
        Gets information about all models in a Lookout for Vision project.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that you want to use.
        """
        try:
            response =
 lookoutvision_client.list_models(ProjectName=project_name)
            print("Project: " + project_name)
```

```
            for model in response["Models"]:
                Models.describe_model(
                    lookoutvision_client, project_name, model["ModelVersion"]
                )
                print()
            print("Done...")
        except ClientError:
            logger.exception("Couldn't list models.")
            raise
```

Java V2

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

```
/**
 * Lists the models in an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that contains the models that
 *                    you want to list.
 * @return List <Metadata> A list of models in the project.
 */
public static List<ModelMetadata> listModels(LookoutVisionClient lfvClient,
 String projectName)
                throws LookoutVisionException {

        ListModelsRequest listModelsRequest = ListModelsRequest.builder()
                        .projectName(projectName)
                        .build();

        // Get a list of models in the supplied project.
        ListModelsResponse response = lfvClient.listModels(listModelsRequest);

        for (ModelMetadata model : response.models()) {
                logger.log(Level.INFO, "Model ARN: {0}\nVersion: {1}\nStatus:
 {2}\nMessage: {3}", new Object[] {
                                model.modelArn(),
                                model.modelVersion(),
                                model.statusMessage(),
                                model.statusAsString() });
        }
```

```
        return response.models();

}
```

Suppression d’un modèle

Vous pouvez supprimer une version d'un modèle à l'aide de la console ou à l'aide de l'`DeleteModel`opération. Vous ne pouvez pas supprimer la version du modèle en cours d'exécution ou en cours d'entraînement.

Si le modèle est en cours d'exécution, utilisez d'abord l'`StopModel`opération pour arrêter la version du modèle. Pour de plus amples informations, veuillez consulter Arrêter votre modèle Amazon Lookout for Vision. Si un modèle est en cours d’entraînement, attendez qu’il soit terminé avant de le supprimer.

La suppression d'un modèle peut prendre quelques secondes. Pour déterminer si un modèle a été supprimé, appelez ListProjectset vérifiez si la version du modèle (`ModelVersion`) se trouve dans le `Models` tableau.

Supprimer un modèle (console)

Procédez comme suit pour supprimer un modèle de la console.

Pour supprimer un modèle (console)

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
2. Choisissez Démarrer.
3. Dans le volet de navigation de gauche, choisissez Projets.
4. Sur la page Projets, sélectionnez le projet contenant le modèle que vous souhaitez supprimer.
5. Dans le volet de navigation de gauche, choisissez Models (Modèles).
6. Dans la vue des modèles, sélectionnez le bouton radio du modèle que vous souhaitez supprimer.
7. En haut de la page, choisissez Supprimer.
8. Dans la boîte de dialogue Supprimer, entrez delete pour confirmer que vous souhaitez supprimer le modèle.

9. Choisissez Supprimer le modèle pour supprimer le modèle.

Suppression d’un modèle (kit SDK)

Utilisez la procédure suivante pour supprimer le modèle lors de l'`DeleteModel`opération.

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 l'exemple de code suivant pour supprimer un modèle.

   CLI

   Remplacez les valeurs suivantes :

   - `project-name`au nom du projet qui contient le modèle que vous souhaitez supprimer.
   - `model-version`à la version du modèle que vous souhaitez supprimer.

   ```
   aws lookoutvision delete-model --project-name project name\
     --model-version model version \
     --profile lookoutvision-access
   ```

   Python

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

   ```
       @staticmethod
       def delete_model(lookoutvision_client, project_name, model_version):
           """
           Deletes a Lookout for Vision model. The model must first be stopped and
   can't
           be in training.

           :param lookoutvision_client: A Boto3 Lookout for Vision client.
           :param project_name: The name of the project that contains the desired
   model.
           :param model_version: The version of the model that you want to delete.
           """
   ```

```
        try:
            logger.info("Deleting model: %s", model_version)
            lookoutvision_client.delete_model(
                ProjectName=project_name, ModelVersion=model_version
            )

            model_exists = True
            while model_exists:
                response =
 lookoutvision_client.list_models(ProjectName=project_name)

                model_exists = False
                for model in response["Models"]:
                    if model["ModelVersion"] == model_version:
                        model_exists = True

                if model_exists is False:
                    logger.info("Model deleted")
                else:
                    logger.info("Model is being deleted...")
                    time.sleep(2)

            logger.info("Deleted Model: %s", model_version)
        except ClientError:
            logger.exception("Couldn't delete model.")
            raise
```

Java V2

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

```
/**
* Deletes an Amazon Lookout for Vision model.
*
* @param lfvClient    An Amazon Lookout for Vision client. Returns after the
 model is deleted.
* @param projectName  The name of the project that contains the model that you
 want to delete.
* @param modelVersion The version of the model that you want to delete.
* @return void
```

```
*/
public static void deleteModel(LookoutVisionClient lfvClient,
                String projectName,
                String modelVersion) throws LookoutVisionException,
 InterruptedException {

        DeleteModelRequest deleteModelRequest = DeleteModelRequest.builder()
                        .projectName(projectName)
                        .modelVersion(modelVersion)
                        .build();

        lfvClient.deleteModel(deleteModelRequest);

        boolean deleted = false;

        do {

                ListModelsRequest listModelsRequest =
 ListModelsRequest.builder()
                                .projectName(projectName)
                                .build();

                // Get a list of models in the supplied project.
                ListModelsResponse response =
 lfvClient.listModels(listModelsRequest);

                ModelMetadata modelMetadata = response.models().stream()
                                .filter(model ->
 model.modelVersion().equals(modelVersion)).findFirst()
                                .orElse(null);

                if (modelMetadata == null) {
                        deleted = true;
                        logger.log(Level.INFO, "Deleted: Model version {0} of
 project {1}.",
                                        new Object[] { modelVersion,
 projectName });

                } else {
                        logger.log(Level.INFO, "Not yet deleted: Model version
 {0} of project {1}.",
                                        new Object[] { modelVersion,
 projectName });
                        TimeUnit.SECONDS.sleep(60);
```

```
                }

        } while (!deleted);

}
```

Modèles de balisage

Vous pouvez identifier, organiser, rechercher et filtrer vos modèles Amazon Lookout for Vision 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, vous pouvez les étiqueter à l'aide d'une `Cost center` clé et ajouter 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 de votre modèle Amazon Lookout for Vision entraîné.

Vous pouvez étiqueter des modèles à l'aide de la console Amazon Lookout for Vision ou à l'aide AWS SDKs du.

Rubriques

- Balisage de modèles (console)
- Balisage de modèles (kit SDK)

Balisage de modèles (console)

Vous pouvez utiliser la console Amazon Lookout for Vision pour ajouter des balises aux modèles, afficher les balises associées à un modèle et supprimer des balises.

Ajouter ou supprimer des tags (console)

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 de plus amples informations, veuillez consulter Entraînement de votre modèle.

Pour ajouter ou supprimer des balises à un modèle existant (console)

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
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 du modèle (console)

Vous pouvez utiliser la console Amazon Lookout for Vision pour afficher les balises associées 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 Répertorier les tags des modèles (SDK).

Pour afficher les balises jointes à un modèle

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/
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

Cette section contient des AWS CLI exemples. Si vous n'avez pas installé le AWS CLI, consultezÉtape 4 : Configurez le AWS CLI et AWS SDKs.

Ajouter des balises à un nouveau modèle (SDK)

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

```
aws lookoutvision create-model --project-name "project name"\
  --output-config '{ "S3Location": { "Bucket": "output bucket", "Prefix":  "output
 folder" } }'\
  --tags '[{"Key":"Key","Value":"Value"}]' \
  --profile lookoutvision-access
```

Pour plus d'informations sur la création et la formation d'un modèle, consultez Entraînement d'un modèle (kit SDK).

Ajouter des balises à un modèle existant (SDK)

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.

```
aws lookoutvision tag-resource --resource-arn "resource-arn"\
  --tags '[{"Key":"Key","Value":"Value"}]' \
  --profile lookoutvision-access
```

Par exemple, le code Java, voir TagModel.

Répertorier les tags des modèles (SDK)

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

```
aws lookoutvision list-tags-for-resource --resource-arn resource-arn \
  --profile lookoutvision-access
```

Pour voir quels modèles d'un projet possèdent une balise spécifique, appelez `ListModels` pour obtenir la liste des modèles. Appelez ensuite `ListTagsForResource` pour chaque modèle dans la

réponse de `ListModels`. Examinez la réponse de `ListTagsForResource` pour vérifier si la balise requise est présente.

Par exemple, le code Java, voir ListModelTags. Par exemple, le code Python qui recherche une valeur de balise dans tous les projets, consultez find_tag.py.

Supprimer des balises d'un modèle (SDK)

Pour supprimer une ou plusieurs balises d'un modèle, utilisez l'UntagResourceopération. Spécifiez le nom de ressource Amazon (ARN) (`ResourceArn`) du modèle et les clés de balise (`Tag-Keys`) que vous souhaitez supprimer.

```
aws lookoutvision untag-resource --resource-arn resource-arn\
  --tag-keys '["Key"]' \
  --profile lookoutvision-access
```

Par exemple, le code Java, voir UntagModel.

Afficher les tâches de détection de vos essais

Vous pouvez consulter vos détections d'essai à l'aide de la console. Vous ne pouvez pas utiliser le AWS SDK pour afficher les tâches de détection des versions d'essai.

> **Note**
>
> La liste des détections dans les essais est finalement cohérente. Si vous créez une détection d'essai, il se peut que vous deviez attendre un peu avant que la liste des détections d'essai soit mise à jour.

Afficher les tâches de détection de votre essai (console)

Suivez les procédures suivantes pour consulter les détections de votre essai.

Pour consulter les tâches de détection de vos essais

1. Ouvrez la console Amazon Lookout for Vision à l'adresse. https://console.aws.amazon.com/lookoutvision/

2. Choisissez Démarrer.
3. Dans le volet de navigation de gauche, choisissez Trial detections.
4. Sur la page des détections d'essai, choisissez une tâche de détection d'essai pour en afficher les détails.

Exemples de code et de jeux de données

Vous trouverez ci-dessous des exemples de code et des ensembles de données que vous pouvez utiliser avec Amazon Lookout for Vision.

Rubriques

- Exemple de code
- Exemples de jeux de données

Exemple de code

Les exemples de code suivants pour Amazon Lookout for Vision sont disponibles.

exemple	Description
GitHub	Exemple de code Python qui entraîne et héberge un modèle Amazon Lookout for Vision.
Amazon Lookout for Vision Lab	Un bloc-notes en Python que vous pouvez utiliser pour créer un modèle avec les images d'exemple du circuit imprimé.
Exemple de code Python	Exemples de Python utilisés dans la documentation Amazon Lookout for Vision.
Exemple de code Java	Exemples de Java utilisés dans la documenta tion Amazon Lookout for Vision.

Exemples de jeux de données

Vous trouverez ci-dessous des exemples de jeux de données que vous pouvez utiliser avec Amazon Lookout for Vision.

Rubriques

- Ensembles de données de segmentation d'images
- Ensemble de données de classification d'images

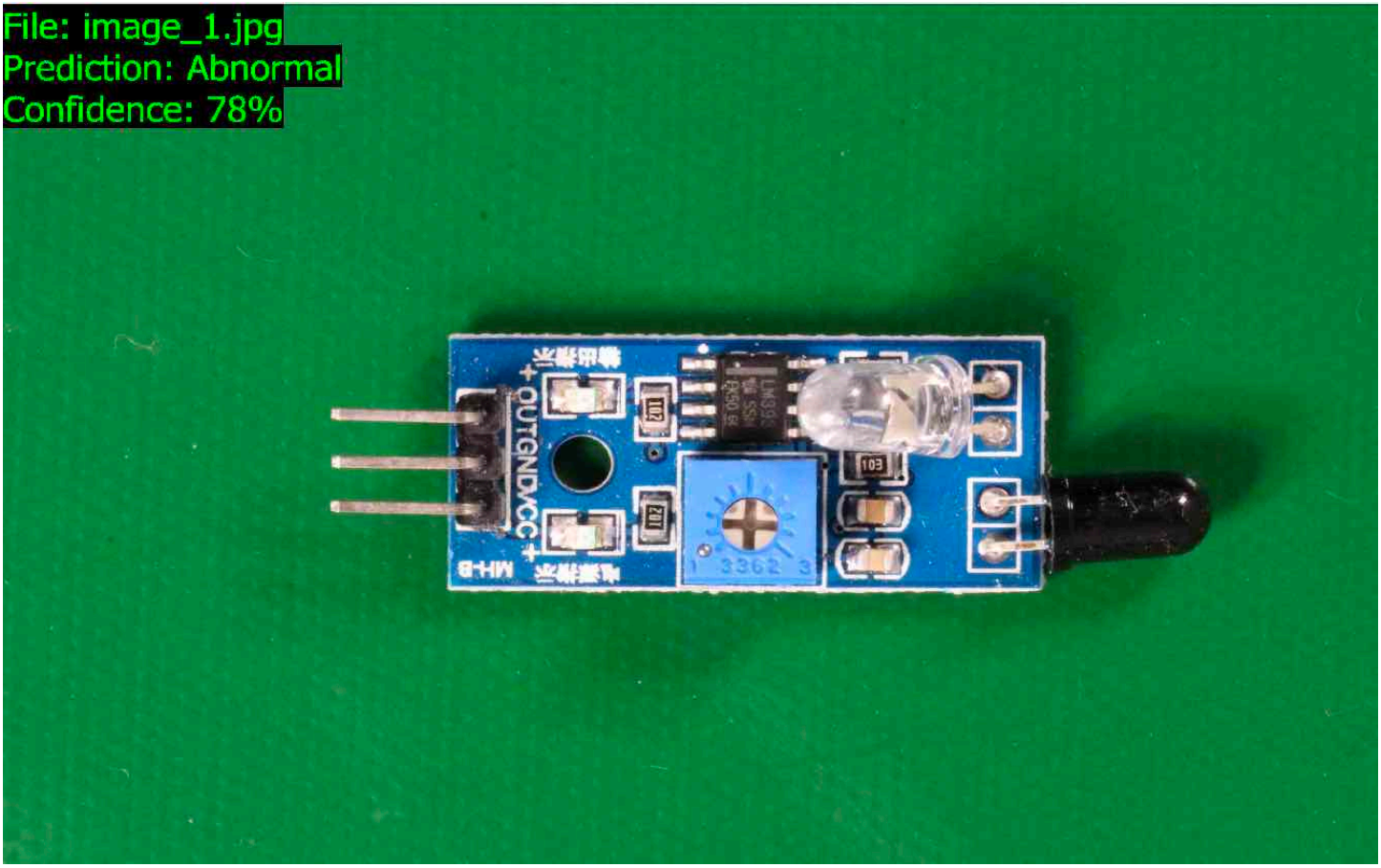
Ensembles de données de segmentation d'images

Commencer à utiliser Amazon Lookout for Visionfournit un ensemble de données de cookies cassés que vous pouvez utiliser pour créer un modèle de segmentation d'image.

Pour un autre jeu de données qui crée un modèle de segmentation d'image, consultez Identifier l'emplacement des anomalies à l'aide d'Amazon Lookout for Vision at the edge sans utiliser de GPU.

Ensemble de données de classification d'images

Amazon Lookout for Vision fournit des exemples d'images de circuits imprimés que vous pouvez utiliser pour créer un modèle de classification d'images.



Vous pouvez copier les images depuis le https://github.com/aws-samples/amazon-lookout-for-vision GitHubréférentiel. Les images se trouvent dans le `circuitboard` dossier.

Le `circuitboard` dossier contient les dossiers suivants.

- `train`— Des images que vous pouvez utiliser dans un jeu de données de formation.

- `test`— Des images que vous pouvez utiliser dans un ensemble de données de test.
- `extra_images`— Des images que vous pouvez utiliser pour effectuer un essai de détection ou pour tester votre modèle entraîné lors de l'DetectAnomaliesopération.

Les `test` dossiers `train` et ont chacun un sous-dossier nommé `normal` (contient des images normales) et un sous-dossier nommé `anomaly` (contient des images présentant des anomalies).

> **Note**
>
> Plus tard, lorsque vous créez un ensemble de données avec la console, Amazon Lookout for Vision peut utiliser les noms des dossiers `normal` (`anomaly`et) pour étiqueter automatiquement les images. Pour de plus amples informations, veuillez consulter the section called “Compartiment Amazon S3”.

Pour préparer les images du jeu de données

1. Clonez le https://github.com/aws-samples/amazon-lookout-for-visiondépôt sur votre ordinateur. Pour plus d'informations, consultez la section Clonage d'un référentiel.
2. Créez un compartiment Amazon S3. Pour plus d'informations, consultez Comment créer un compartiment S3 ? .
3. À l'invite de commande, entrez la commande suivante pour copier les images du jeu de données de votre ordinateur vers votre compartiment Amazon S3.

```
aws s3 cp --recursive your-repository-folder/circuitboard s3://your-bucket/
circuitboard
```

Après avoir chargé les images, vous pouvez créer un modèle. Vous pouvez classer automatiquement les images en ajoutant les images provenant de l'emplacement Amazon S3 où vous avez précédemment chargé les images du circuit imprimé. N'oubliez pas que vous êtes facturé pour chaque formation réussie d'un modèle et pour le temps pendant lequel un modèle est exécuté (hébergé).

Pour créer un modèle de classification

1. FaisCréation d'un projet (console).

2. FaisCréation d'un ensemble de données à partir d'images stockées dans un compartiment Amazon S3.

   - Pour l'étape 6, choisissez l'onglet Séparer les ensembles de données d'entraînement et de test.
   - Pour l'étape 8a, entrez l'URI S3 pour les images d'entraînement que vous avez téléchargées dans Pour préparer les images du jeu de données. Par exemple, `s3://your-bucket/circuitboard/train`. Pour l'étape 8b, entrez l'URI S3 de l'ensemble de données de test. Par exemple, `s3://your-bucket/circuitboard/test`.
   - Assurez-vous de suivre l'étape 9.
3. FaisEntraînement d'un modèle (console).
4. FaisDémarrage de votre modèle (console).
5. FaisDétecter des anomalies dans une image. Vous pouvez utiliser les images du `test_images` dossier.
6. Lorsque vous aurez terminé avec le modèle, faites-leArrêt de votre modèle (console).

La sécurité dans Amazon Lookout for Vision

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez de centres de données et d'architectures réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cela comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de AWS conformité Programmes de de conformité. Pour en savoir plus sur les programmes de conformité qui s'appliquent à Amazon Lookout for Vision, consultez la section Services AWS concernés par programme de conformité .
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris de la sensibilité de vos données, des exigences de votre entreprise, ainsi que de la législation et de la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation de Lookout for Vision. Les rubriques suivantes expliquent comment configurer Lookout for Vision pour répondre à vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres services AWS qui vous aident à surveiller et à sécuriser vos ressources Lookout for Vision.

Rubriques

- Protection des données dans Amazon Lookout for Vision
- Gestion des identités et des accès pour Amazon Lookout for Vision
- Validation de conformité pour Amazon Lookout for Vision
- La résilience dans Amazon Lookout for Vision
- Sécurité de l'infrastructure dans Amazon Lookout for Vision

Protection des données dans Amazon Lookout for Vision

Le modèle de responsabilité AWS partagée s'applique à la protection des données dans Amazon Lookout for Vision. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez Questions fréquentes (FAQ) sur la confidentialité des données. Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog Modèle de responsabilité partagée AWS et RGPD (Règlement général sur la protection des données) sur le Blog de sécuritéAWS .

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- Utilisez le protocole SSL/TLS pour communiquer avec les ressources. AWS Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section Utilisation des CloudTrail sentiers dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-3 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS disponibles, consultez Norme FIPS (Federal Information Processing Standard) 140-3.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec Lookout for Vision ou un Services AWS

autre outil à l'aide de la console, de l'API AWS CLI ou. AWS SDKs Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Chiffrement des données

Les informations suivantes expliquent les domaines dans lesquels Amazon Lookout for Vision utilise le chiffrement des données pour protéger vos données.

Chiffrement au repos

Images

Pour entraîner votre modèle, Amazon Lookout for Vision crée une copie de vos images d'entraînement et de test source. Les images copiées sont chiffrées au repos dans Amazon Simple Storage Service (S3) à l'aide d'un chiffrement côté serveur à l'aide d'une clé ou d' Clé détenue par AWS une clé que vous fournissez. Les clés sont stockées à l'aide d'AWS Key Management Service (SSE-KMS). Vos images sources ne sont pas affectées. Pour de plus amples informations, veuillez consulter Entraînement de votre modèle.

Modèles Amazon Lookout for Vision

Par défaut, les modèles entraînés et les fichiers manifestes sont chiffrés dans Amazon S3 à l'aide d'un chiffrement côté serveur avec des clés KMS stockées dans AWS Key Management Service (SSE-KMS). Lookout for Vision utilise Clé détenue par AWS un. Pour plus d'informations, consultez Protection des données à l'aide du chiffrement côté serveur. Les résultats d'entraînement sont écrits dans le compartiment spécifié dans le paramètre `output_bucket` d'entrée to`CreateModel`. Les résultats de l'entraînement sont chiffrés à l'aide des paramètres de chiffrement configurés pour le compartiment (`output_bucket`).

Compartiment de console Amazon Lookout for Vision

La console Amazon Lookout for Vision crée un compartiment Amazon S3 (compartiment de console) que vous pouvez utiliser pour gérer vos projets. Le compartiment de console est chiffré à l'aide du chiffrement par défaut d'Amazon S3. Pour plus d'informations, veuillez consulter la rubrique Chiffrement par défaut Amazon S3 pour les compartiments S3. Si vous utilisez votre propre clé KMS, configurez le compartiment de console après sa création. Pour plus d'informations, consultez

Protection des données à l’aide du chiffrement côté serveur. Amazon Lookout for Vision bloque l'accès public au bucket de console.

Chiffrement en transit

Les points de terminaison de l'API Amazon Lookout for Vision prennent uniquement en charge les connexions sécurisées via HTTPS. Toutes les communications sont chiffrées avec Transport Layer Security (TLS).

Gestion des clés

Vous pouvez utiliser AWS Key Management Service (KMS) pour gérer le chiffrement des images d'entrée que vous stockez dans des compartiments Amazon S3. Pour de plus amples informations, veuillez consulter Étape 5 : (Facultatif) Utilisation de votre propre clé AWS Key Management Service.

Par défaut, vos images sont chiffrées à l'aide d'une clé détenue et gérée par AWS. Vous pouvez également choisir d'utiliser votre propre clé AWS Key Management Service (KMS). Pour plus d’informations, consultez Concepts d’AWS Key Management Service.

Confidentialité du trafic inter-réseau

Un point de terminaison Amazon Virtual Private Cloud (Amazon VPC) pour Amazon Lookout for Vision est une entité logique au sein d'un VPC qui permet la connectivité uniquement à Amazon Lookout for Vision. Amazon VPC achemine les demandes vers Amazon Lookout for Vision et renvoie les réponses au VPC. Pour plus d’informations, consultez Points de terminaison d’un VPC dans le Guide de l’utilisateur Amazon VPC. Pour plus d'informations sur l'utilisation des points de terminaison Amazon VPC avec Amazon Lookout for Vision, consultez. Accédez à Amazon Lookout for Vision à l'aide d'un point de terminaison AWS PrivateLink d'interface ()

Gestion des identités et des accès pour Amazon Lookout for Vision

AWS Identity and Access Management (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser les ressources Lookout for Vision. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- Public ciblé

- Authentification par des identités
- Gestion des accès à l’aide de politiques
- Comment Amazon Lookout for Vision fonctionne avec IAM
- Exemples de politiques basées sur l'identité Amazon Lookout for Vision
- AWS politiques gérées pour Amazon Lookout for Vision
- Résolution des problèmes d'identité et d'accès à Amazon Lookout for Vision

Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) varie en fonction du travail que vous effectuez dans Lookout for Vision.

Utilisateur du service : si vous utilisez le service Lookout for Vision dans le cadre de votre travail, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Au fur et à mesure que vous utilisez de plus en plus de fonctionnalités de Lookout for Vision dans le cadre de votre travail, vous aurez peut-être besoin d'autorisations supplémentaires. En comprenant bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne parvenez pas à accéder à une fonctionnalité dans Lookout for Vision, Résolution des problèmes d'identité et d'accès à Amazon Lookout for Vision consultez.

Administrateur du service : si vous êtes responsable des ressources Lookout for Vision au sein de votre entreprise, vous avez probablement un accès complet à Lookout for Vision. C'est à vous de déterminer les fonctionnalités et ressources de Lookout for Vision auxquelles les utilisateurs de votre service doivent accéder. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d’IAM. Pour en savoir plus sur la manière dont votre entreprise peut utiliser l'IAM avec Lookout for Vision, consultez. Comment Amazon Lookout for Vision fonctionne avec IAM

Administrateur IAM : si vous êtes administrateur IAM, vous souhaiterez peut-être en savoir plus sur la manière dont vous pouvez rédiger des politiques pour gérer l'accès à Lookout for Vision. Pour consulter des exemples de politiques basées sur l'identité de Lookout for Vision que vous pouvez utiliser dans IAM, consultez. Exemples de politiques basées sur l'identité Amazon Lookout for Vision

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAM Identity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez la section Comment vous connecter à votre compte Compte AWS dans le guide de Connexion à AWS l'utilisateur.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer des demandes vous-même, consultez AWS Signature Version 4 pour les demandes d'API dans le Guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour plus d'informations, consultez Authentification multifactorielle dans le Guide de l'utilisateur AWS IAM Identity Center et Authentification multifactorielle AWS dans IAM dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une identité de connexion unique qui donne un accès complet à toutes Services AWS les ressources du compte. Cette identité est appelée utilisateur Compte AWS root et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas

utiliser l'utilisateur racine pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur racine, consultez Tâches nécessitant des informations d'identification d'utilisateur racine dans le Guide de l'utilisateur IAM.

Identité fédérée

La meilleure pratique consiste à obliger les utilisateurs humains, y compris ceux qui ont besoin d'un accès administrateur, à utiliser la fédération avec un fournisseur d'identité pour accéder à l'aide Services AWS d'informations d'identification temporaires.

Une identité fédérée est un utilisateur de l'annuaire des utilisateurs de votre entreprise, d'un fournisseur d'identité Web AWS Directory Service, du répertoire Identity Center ou de tout utilisateur qui y accède à l'aide des informations d'identification fournies Services AWS par le biais d'une source d'identité. Lorsque des identités fédérées y accèdent Comptes AWS, elles assument des rôles, qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Vous pouvez créer des utilisateurs et des groupes dans IAM Identity Center, ou vous pouvez vous connecter et synchroniser avec un ensemble d'utilisateurs et de groupes dans votre propre source d'identité afin de les utiliser dans toutes vos applications Comptes AWS et applications. Pour obtenir des informations sur IAM Identity Center, consultez Qu'est-ce que IAM Identity Center ? dans le Guide de l'utilisateur AWS IAM Identity Center .

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité au sein de vous Compte AWS qui possède des autorisations spécifiques pour une seule personne ou une seule application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme telles que des mots de passe et des clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons d'effectuer une rotation des clés d'accès. Pour plus d'informations, consultez Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification dans le Guide de l'utilisateur IAM.

Un groupe IAM est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations

pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez nommer un groupe IAMAdminset lui donner les autorisations nécessaires pour administrer les ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour plus d'informations, consultez Cas d'utilisation pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité au sein de vous Compte AWS dotée d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Pour assumer temporairement un rôle IAM dans le AWS Management Console, vous pouvez passer d'un rôle d'utilisateur à un rôle IAM (console). Vous pouvez assumer un rôle en appelant une opération d' AWS API AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez Méthodes pour endosser un rôle dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Accès utilisateur fédéré : pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, consultez Jeux d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .
- Autorisations d'utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, dans certains Services AWS cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les

ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

- Accès multiservices — Certains Services AWS utilisent des fonctionnalités dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, un rôle de service ou un rôle lié au service.
  - Sessions d'accès direct (FAS) : lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez Transmission des sessions d'accès.
  - Rôle de service : il s'agit d'un rôle IAM attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.
  - Rôle lié à un service — Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS fichier et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.
- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une EC2 instance et qui envoient des demandes AWS CLI d' AWS API. Cela est préférable au stockage des clés d'accès dans l' EC2 instance. Pour attribuer un AWS rôle à une EC2 instance et le rendre disponible pour toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes exécutés sur l' EC2 instance d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez Utiliser un rôle IAM pour accorder des autorisations aux applications exécutées sur des EC2 instances Amazon dans le guide de l'utilisateur IAM.

Gestion des accès à l’aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d’informations sur la structure et le contenu des documents de politique JSON, consultez Vue d’ensemble des politiques JSON dans le Guide de l’utilisateur IAM.

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C’est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d’aucune autorisation. Pour octroyer aux utilisateurs des autorisations d’effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L’administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Les politiques IAM définissent les autorisations d’une action, quelle que soit la méthode que vous utilisez pour exécuter l’opération. Par exemple, supposons que vous disposiez d’une politique qui autorise l’action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le rôle à partir de AWS Management Console AWS CLI, de ou de l' AWS API.

Politiques basées sur l’identité

Les politiques basées sur l’identité sont des documents de politique d’autorisations JSON que vous pouvez attacher à une identité telle qu’un utilisateur, un groupe d’utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d’actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l’identité, consultez Définition d’autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l’utilisateur IAM.

Les politiques basées sur l’identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles au sein de votre Compte AWS. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour découvrir comment choisir entre

une politique gérée et une politique en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

Politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACLs)

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et AWS WAF Amazon VPC sont des exemples de services compatibles. ACLs Pour en savoir plus ACLs, consultez la présentation de la liste de contrôle d'accès (ACL) dans le guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- Limite d'autorisations : une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le nombre maximal d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d'autorisations pour une entité. Les autorisations en résultant représentent la combinaison des politiques basées sur

l'identité d'une entité et de ses limites d'autorisation. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.

- Politiques de contrôle des services (SCPs) : SCPs politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée Comptes AWS les multiples propriétés de votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer des politiques de contrôle des services (SCPs) à l'un ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités figurant dans les comptes des membres, y compris chacune Utilisateur racine d'un compte AWS d'entre elles. Pour plus d'informations sur les Organizations SCPs, voir Politiques de contrôle des services dans le Guide de AWS Organizations l'utilisateur.
- Politiques de contrôle des ressources (RCPs) : RCPs politiques JSON que vous pouvez utiliser pour définir le maximum d'autorisations disponibles pour les ressources de vos comptes sans mettre à jour les politiques IAM associées à chaque ressource que vous possédez. Le RCP limite les autorisations pour les ressources des comptes membres et peut avoir un impact sur les autorisations effectives pour les identités, y compris Utilisateur racine d'un compte AWS, qu'elles appartiennent ou non à votre organisation. Pour plus d'informations sur les Organizations RCPs, y compris une liste de ces Services AWS supports RCPs, consultez la section Resource control policies (RCPs) dans le guide de AWS Organizations l'utilisateur.
- Politiques de séance : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de séance en résultant sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Comment Amazon Lookout for Vision fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à Lookout for Vision, découvrez quelles fonctionnalités IAM peuvent être utilisées avec Lookout for Vision.

Fonctionnalités IAM que vous pouvez utiliser avec Amazon Lookout for Vision

Fonctionnalité IAM	Assistance Lookout for Vision
Politiques basées sur l’identité	Oui
Politiques basées sur les ressources	Non
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition de politique (spécifiques au service)	Oui
ACLs	Non
ABAC (identifications dans les politiques)	Partielle
Informations d’identification temporaires	Oui
Transmission des sessions d’accès (FAS)	Oui
Fonctions du service	Non
Rôles liés à un service	Non

Pour obtenir une vue d'ensemble de la façon dont Lookout for Vision et les AWS autres services fonctionnent avec la plupart des fonctionnalités IAM, AWS consultez la section Services compatibles avec IAM dans le guide de l'utilisateur IAM.

Politiques basées sur l'identité pour Lookout for Vision

Prend en charge les politiques basées sur l’identité : oui

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Définition d'autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l'utilisateur IAM.

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Vous ne pouvez pas spécifier le principal dans une politique basée sur une identité, car celle-ci s'applique à l'utilisateur ou au rôle auquel elle est attachée. Pour découvrir tous les éléments que vous utilisez dans une politique JSON, consultez Références des éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

Exemples de politiques basées sur l'identité pour Lookout for Vision

Pour consulter des exemples de politiques basées sur l'identité de Lookout for Vision, consultez. Exemples de politiques basées sur l'identité Amazon Lookout for Vision

Politiques basées sur les ressources dans Lookout for Vision

Prend en charge les politiques basées sur les ressources : non

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Pour permettre un accès intercompte, vous pouvez spécifier un compte entier ou des entités IAM dans un autre compte en tant que principal dans une politique basée sur les ressources. L'ajout d'un principal intercompte à une politique basée sur les ressources ne représente qu'une partie de l'instauration de la relation d'approbation. Lorsque le principal et la ressource sont différents Comptes AWS, un administrateur IAM du compte sécurisé doit également accorder à l'entité principale (utilisateur ou rôle) l'autorisation d'accéder à la ressource. Pour ce faire, il attache une

politique basée sur une identité à l'entité. Toutefois, si une politique basée sur des ressources accorde l'accès à un principal dans le même compte, aucune autre politique basée sur l'identité n'est requise. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Actions politiques pour Lookout for Vision

Prend en charge les actions de politique : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Les actions de stratégie portent généralement le même nom que l'opération AWS d'API associée. Il existe quelques exceptions, telles que les actions avec autorisations uniquement qui n'ont pas d'opération API correspondante. Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Pour consulter la liste des actions Lookout for Vision, consultez la section Actions définies par Amazon Lookout for Vision dans le Service Authorization Reference.

Les actions de stratégie dans Lookout for Vision utilisent le préfixe suivant avant l'action :

```
lookoutvision
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

```
"Action": [
      "lookoutvision:action1",
      "lookoutvision:action2"
         ]
```

Pour consulter des exemples de politiques basées sur l'identité de Lookout for Vision, consultez. Exemples de politiques basées sur l'identité Amazon Lookout for Vision

Ressources relatives aux politiques pour Lookout for Vision

Prend en charge les ressources de politique : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Les instructions doivent inclure un élément `Resource` ou `NotResource`. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Vous pouvez le faire pour des actions qui prennent en charge un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, telles que les opérations de liste, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

Pour consulter la liste des types de ressources Lookout for Vision et ARNs leurs caractéristiques, consultez la section Ressources définies par Amazon Lookout for Vision dans le Service Authorization Reference. Pour savoir avec quelles actions vous pouvez spécifier l'ARN de chaque ressource, consultez la section Actions définies par Amazon Lookout for Vision.

Pour consulter des exemples de politiques basées sur l'identité de Lookout for Vision, consultez. Exemples de politiques basées sur l'identité Amazon Lookout for Vision

Clés de conditions générales pour Lookout for Vision

Prend en charge les clés de condition de politique spécifiques au service : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Condition` (ou le bloc `Condition`) vous permet de spécifier des conditions lorsqu'une instruction est appliquée. L'élément `Condition` est facultatif. Vous pouvez créer des expressions conditionnelles qui utilisent des opérateurs de condition, tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande.

Si vous spécifiez plusieurs éléments `Condition` dans une instruction, ou plusieurs clés dans un seul élément `Condition`, AWS les évalue à l'aide d'une opération `AND` logique. Si vous spécifiez plusieurs valeurs pour une seule clé de condition, AWS évalue la condition à l'aide d'une `OR` opération logique. Toutes les conditions doivent être remplies avant que les autorisations associées à l'instruction ne soient accordées.

Vous pouvez aussi utiliser des variables d'espace réservé quand vous spécifiez des conditions. Par exemple, vous pouvez accorder à un utilisateur IAM l'autorisation d'accéder à une ressource uniquement si elle est balisée avec son nom d'utilisateur IAM. Pour plus d'informations, consultez Éléments d'une politique IAM : variables et identifications dans le Guide de l'utilisateur IAM.

AWS prend en charge les clés de condition globales et les clés de condition spécifiques au service. Pour voir toutes les clés de condition AWS globales, voir les clés de contexte de condition AWS globales dans le guide de l'utilisateur IAM.

Pour consulter la liste des clés de condition de Lookout for Vision, consultez la section Clés de condition d'Amazon Lookout for Vision dans le manuel de référence d'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez la section Actions définies par Amazon Lookout for Vision.

Pour consulter des exemples de politiques basées sur l'identité de Lookout for Vision, consultez. Exemples de politiques basées sur l'identité Amazon Lookout for Vision

ACLs dans Lookout for Vision

Supports ACLs : Non

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

ABAC avec Lookout for Vision

Prend en charge ABAC (identifications dans les politiques) : partiellement

Le contrôle d'accès par attributs (ABAC) est une stratégie d'autorisation qui définit des autorisations en fonction des attributs. Dans AWS, ces attributs sont appelés balises. Vous pouvez associer des balises aux entités IAM (utilisateurs ou rôles) et à de nombreuses AWS ressources. L'étiquetage des entités et des ressources est la première étape d'ABAC. Vous concevez ensuite des politiques ABAC pour autoriser des opérations quand l'identification du principal correspond à celle de la ressource à laquelle il tente d'accéder.

L'ABAC est utile dans les environnements qui connaissent une croissance rapide et pour les cas où la gestion des politiques devient fastidieuse.

Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans l'élément de condition d'une politique utilisant les clés de condition `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d'informations sur ABAC, consultez Définition d'autorisations avec l'autorisation ABAC dans le Guide de l'utilisateur IAM. Pour accéder à un didacticiel décrivant les étapes de configuration de l'ABAC, consultez Utilisation du contrôle d'accès par attributs (ABAC) dans le Guide de l'utilisateur IAM.

Utilisation d'informations d'identification temporaires avec Lookout for Vision

Prend en charge les informations d'identification temporaires : oui

Certains Services AWS ne fonctionnent pas lorsque vous vous connectez à l'aide d'informations d'identification temporaires. Pour plus d'informations, y compris celles qui Services AWS fonctionnent avec des informations d'identification temporaires, consultez Services AWS la section relative à l'utilisation d'IAM dans le guide de l'utilisateur d'IAM.

Vous utilisez des informations d'identification temporaires si vous vous connectez à l' AWS Management Console aide d'une méthode autre qu'un nom d'utilisateur et un mot de passe. Par exemple, lorsque vous accédez à AWS l'aide du lien d'authentification unique (SSO) de votre entreprise, ce processus crée automatiquement des informations d'identification temporaires. Vous créez également automatiquement des informations d'identification temporaires lorsque vous vous connectez à la console en tant qu'utilisateur, puis changez de rôle. Pour plus d'informations sur le changement de rôle, consultez Passage d'un rôle utilisateur à un rôle IAM (console) dans le Guide de l'utilisateur IAM.

Vous pouvez créer manuellement des informations d'identification temporaires à l'aide de l' AWS API AWS CLI or. Vous pouvez ensuite utiliser ces informations d'identification temporaires pour y accéder AWS. AWS recommande de générer dynamiquement des informations d'identification temporaires au lieu d'utiliser des clés d'accès à long terme. Pour plus d'informations, consultez Informations d'identification de sécurité temporaires dans IAM.

Sessions d'accès direct pour Lookout for Vision

Prend en charge les sessions d’accès direct (FAS) : oui

Lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d’autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez Transmission des sessions d’accès.

Rôles de service pour Lookout for Vision

Prend en charge les rôles de service : Non

Un rôle de service est un rôle IAM qu’un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d’IAM. Pour plus d’informations, consultez Création d’un rôle pour la délégation d’autorisations à un Service AWS dans le Guide de l’utilisateur IAM.

> **Warning**
>
> La modification des autorisations associées à un rôle de service peut perturber les fonctionnalités de Lookout for Vision. Modifiez les rôles de service uniquement lorsque Lookout for Vision fournit des instructions à cet effet.

Rôles liés à un service pour Lookout for Vision

Prend en charge les rôles liés à un service : non

Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d’effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS fichier et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Pour plus d'informations sur la création ou la gestion des rôles liés à un service, consultez Services AWS qui fonctionnent avec IAM. Recherchez un service dans le tableau qui inclut un `Yes` dans la colonne Rôle lié à un service. Choisissez le lien Oui pour consulter la documentation du rôle lié à ce service.

Exemples de politiques basées sur l'identité Amazon Lookout for Vision

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou à modifier les ressources Lookout for Vision. Ils ne peuvent pas non plus effectuer de tâches à l'aide de l'API AWS Management Console, AWS Command Line Interface (AWS CLI) ou de AWS l'API. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, consultez Création de politiques IAM (console) dans le Guide de l'utilisateur IAM.

Pour plus de détails sur les actions et les types de ressources définis par Lookout for Vision, y compris le format du ARNs pour chacun des types de ressources, consultez la section Actions, ressources et clés de condition pour Amazon Lookout for Vision dans le Service Authorization Reference.

Rubriques

- Bonnes pratiques en matière de politiques
- Accès à un seul projet Amazon Lookout for Vision
- Exemples de politique basée sur des balises

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si quelqu'un peut créer, accéder ou supprimer des ressources Lookout for Vision dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire

davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez politiques gérées par AWS ou politiques gérées par AWS pour les activités professionnelles dans le Guide de l'utilisateur IAM.

- Accordez les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez politiques et autorisations dans IAM dans le Guide de l'utilisateur IAM.
- Utilisez des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que AWS CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.
- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez Validation de politiques avec IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Sécurisation de l'accès aux API avec MFA dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.

Accès à un seul projet Amazon Lookout for Vision

Dans cet exemple, vous souhaitez autoriser un utilisateur de votre AWS compte à accéder à l'un de vos projets Amazon Lookout for Vision.

```
{
    "Sid": "SpecificProjectOnly",
    "Effect": "Allow",
    "Action": [
        "lookoutvision:DetectAnomalies"
    ],
    "Resource": "arn:aws:lookoutvision:us-east-1:123456789012:model/myproject/*"
}
```

Exemples de politique basée sur des balises

Les politiques basées sur des balises sont des documents de politique JSON qui spécifient les actions qu'un principal peut effectuer sur des ressources balisées.

Utiliser un tag pour accéder à une ressource

Cet exemple de politique accorde à un utilisateur ou à un rôle de votre compte AWS l'autorisation d'utiliser l'`DetectAnomalies`opération avec n'importe quel modèle étiqueté avec la clé `stage` et la valeur`production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "LookoutVision:DetectAnomalies"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "production"
                }
            }
        }
   ]
}
```

Utiliser un tag pour refuser l'accès à des opérations Amazon Lookout for Vision spécifiques

Cet exemple de politique refuse à un utilisateur ou à un rôle de votre compte AWS l'autorisation d'appeler les `StopModel` opérations `DeleteModel` or avec un modèle étiqueté avec la clé `stage` et la valeur`production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "LookoutVision:DeleteModel",
                "LookoutVision:StopModel"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "production"
                }
            }
        }
   ]
}
```

AWS politiques gérées pour Amazon Lookout for Vision

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte

toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle politique Service AWS est lancée ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez Politiques gérées par AWS dans le Guide de l'utilisateur IAM.

Politique gérée par AWS : AmazonLookoutVisionReadOnlyAccess

Utilisez cette `AmazonLookoutVisionReadOnlyAccess` politique pour autoriser les utilisateurs à accéder en lecture seule à Amazon Lookout for Vision (et à ses dépendances) avec les actions Amazon Lookout for Vision suivantes (opérations du SDK). Par exemple, vous pouvez l'utiliser `DescribeModel` pour obtenir des informations sur un modèle existant.

- DescribeDataset
- DescribeModel
- DescribeModelPackagingJob
- DescribeProject
- ListDatasetEntries
- ListModelPackagingJobs
- ListModels
- ListProjects
- ListTagsForResource

Pour appeler des actions en lecture seule, les utilisateurs n'ont pas besoin d'autorisations de compartiment Amazon S3. Toutefois, les réponses aux opérations peuvent inclure des références à des compartiments Amazon S3. Par exemple, l'`source-ref`entrée du formulaire de réponse `ListDatasetEntries` fait référence à une image dans un compartiment Amazon S3. Ajoutez des autorisations de compartiment Amazon S3 si vos utilisateurs ont besoin d'accéder à des compartiments référencés. Par exemple, un utilisateur peut vouloir télécharger une image référencée par un `source-ref` champ. Pour de plus amples informations, veuillez consulter Octroi d'autorisations pour Amazon S3 Bucket.

Vous pouvez associer la politique `AmazonLookoutVisionReadOnlyAccess` à vos identités IAM.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:DescribeDataset",
                "lookoutvision:DescribeModel",
                "lookoutvision:DescribeProject",
                "lookoutvision:DescribeModelPackagingJob",
                "lookoutvision:ListDatasetEntries",
                "lookoutvision:ListModels",
                "lookoutvision:ListProjects",
                "lookoutvision:ListTagsForResource",
                "lookoutvision:ListModelPackagingJobs"
            ],
            "Resource": "*"
        }
    ]
}
```

Politique gérée par AWS : AmazonLookoutVisionFullAccess

Utilisez cette `AmazonLookoutVisionFullAccess` politique pour autoriser les utilisateurs à accéder pleinement à Amazon Lookout for Vision (et à ses dépendances) grâce aux actions Amazon Lookout for Vision (opérations du SDK). Par exemple, vous pouvez entraîner un modèle sans avoir à utiliser la console Amazon Lookout for Vision. Pour plus d'informations, consultez Actions.

Pour créer un ensemble de données (`CreateDataset`) ou créer un modèle (`CreateModel`), vos utilisateurs doivent disposer d'autorisations d'accès complètes au compartiment Amazon S3 qui stocke les images du jeu de données, les fichiers manifestes Amazon SageMaker AI Ground Truth et les résultats de formation. Pour de plus amples informations, veuillez consulter Étape 2 : configurer les autorisations.

Vous pouvez également autoriser les actions du SDK Amazon Lookout for Vision en utilisant `AmazonLookoutVisionConsoleFullAccess` cette politique.

Vous pouvez associer la politique `AmazonLookoutVisionFullAccess` à vos identités IAM.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionFullAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Politique gérée par AWS : AmazonLookoutVisionConsoleFullAccess

Utilisez cette `AmazonLookoutVisionFullAccess` politique pour autoriser les utilisateurs à accéder pleinement à la console Amazon Lookout for Vision, aux actions (opérations du SDK) et à toutes les dépendances du service. Pour de plus amples informations, veuillez consulter Commencer à utiliser Amazon Lookout for Vision.

La `LookoutVisionConsoleFullAccess` politique inclut les autorisations d'accès à votre bucket de console Amazon Lookout for Vision. Pour plus d'informations sur le bucket de console, consultezÉtape 3 : Créer le compartiment de la console. Pour stocker des ensembles de données, des images et des fichiers manifestes Amazon SageMaker AI Ground Truth dans un autre compartiment Amazon S3, vos utilisateurs ont besoin d'autorisations supplémentaires. Pour de plus amples informations, veuillez consulter the section called “Configuration des autorisations du compartiment Amazon S3”.

Vous pouvez associer la politique `AmazonLookoutVisionConsoleFullAccess` à vos identités IAM.

Groupes d'autorisations

Cette politique est regroupée en instructions en fonction de l'ensemble des autorisations fournies :

- `LookoutVisionFullAccess`— Permet d'accéder à toutes les actions de Lookout for Vision.
- `LookoutVisionConsoleS3BucketSearchAccess`— Permet de répertorier tous les compartiments Amazon S3 appartenant à l'appelant. Lookout for Vision utilise cette action pour identifier le bucket de console Lookout for Vision spécifique à la région AWS, s'il en existe un dans le compte de l'appelant.
- `LookoutVisionConsoleS3BucketFirstUseSetupAccessPermissions`— Permet de créer et de configurer des compartiments Amazon S3 qui correspondent au modèle de nom de compartiment de la console Lookout for Vision. Lookout for Vision utilise ces actions pour créer et configurer un bucket de console Lookout for Vision spécifique à une région lorsqu'il n'en trouve aucun.
- `LookoutVisionConsoleS3BucketAccess`— Autorise les actions Amazon S3 dépendantes sur les compartiments qui correspondent au modèle de nom de compartiment de la console Lookout for Vision. Lookout for Vision permet de rechercher `s3:ListBucket` des objets d'image lors de la création d'un ensemble de données à partir d'un compartiment Amazon S3 et lors du lancement d'une tâche de détection d'essai. Lookout for Vision `s3:GetBucketLocation` utilise `s3:GetBucketVersioning` et pour valider la région, le propriétaire et la configuration AWS du bucket dans le cadre des opérations suivantes :
  - Création d'un jeu de données
  - Entraînement d’un modèle
  - Démarrage d'une tâche de détection d'essai
  - Réaliser des commentaires sur la détection des essais

  `LookoutVisionConsoleS3ObjectAccess`— Permet de lire et d'écrire des objets Amazon S3 dans des compartiments qui correspondent au modèle de nom de compartiment de la console Lookout for Vision. Lookout for Vision utilise ces actions pour afficher des images dans les vues de galerie de la console et pour télécharger de nouvelles images destinées à être utilisées dans des ensembles de données. En outre, ces autorisations permettent à Lookout for Vision d'écrire des métadonnées lors de la création d'un ensemble de données, de l'entraînement d'un modèle,

du démarrage d'une tâche de détection d'essai et de la réception de commentaires sur la détection d'un essai.

- `LookoutVisionConsoleDatasetLabelingToolsAccess`— Autorise les actions d' GroundTruth étiquetage Amazon SageMaker AI dépendantes. Lookout for Vision utilise ces actions pour scanner les compartiments S3 à la recherche d'images, GroundTruth créer des fichiers manifestes et annoter les résultats des tâches de détection des essais à l'aide d'étiquettes de validation.
- `LookoutVisionConsoleDashboardAccess`- Permet de lire les CloudWatch statistiques d'Amazon. Lookout for Vision utilise ces actions pour remplir les graphiques du tableau de bord et les statistiques relatives aux anomalies détectées.
- `LookoutVisionConsoleTagSelectorAccess`— Permet de lire les suggestions de clé et de valeur de balise spécifiques au compte. Lookout for Vision utilise ces autorisations pour fournir des recommandations concernant les clés de balise et les valeurs de balise dans les pages de la console de gestion des balises.
- `LookoutVisionConsoleKmsKeySelectorAccess`— Permet de répertorier AWS Key Management Service (KMS) les clés et les alias. Amazon Lookout for Vision utilise cette autorisation pour renseigner les clés KMS dans la sélection de balises suggérée pour certaines actions Lookout for Vision qui prennent en charge les clés KMS gérées par le client à des fins de chiffrement.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionFullAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketSearchAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
```

```
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketFirstUseSetupAccess",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketVersioning",
                "s3:PutLifecycleConfiguration",
                "s3:PutEncryptionConfiguration",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:GetBucketAcl",
                "s3:GetBucketVersioning"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*"
        },
        {
            "Sid": "LookoutVisionConsoleS3ObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*/*"
        },
        {
            "Sid": "LookoutVisionConsoleDatasetLabelingToolsAccess",
            "Effect": "Allow",
            "Action": [
                "groundtruthlabeling:RunGenerateManifestByCrawlingJob",
                "groundtruthlabeling:AssociatePatchToManifestJob",
                "groundtruthlabeling:DescribeConsoleJob"
            ],
```

```
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleDashboardAccess",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleTagSelectorAccess",
            "Effect": "Allow",
            "Action": [
                "tag:GetTagKeys",
                "tag:GetTagValues"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleKmsKeySelectorAccess",
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases"
            ],
            "Resource": "*"
        }
    ]
}
```

Politique gérée par AWS : AmazonLookoutVisionConsoleReadOnlyAccess

Utilisez cette `AmazonLookoutVisionConsoleReadOnlyAccess` politique pour autoriser les utilisateurs à accéder en lecture seule à la console Amazon Lookout for Vision, aux actions (opérations du SDK) et à toutes les dépendances du service.

La `AmazonLookoutVisionConsoleReadOnlyAccess` politique inclut les autorisations Amazon S3 pour le bucket de console Amazon Lookout for Vision. Si les images de votre jeu de données ou les fichiers manifestes Amazon SageMaker AI Ground Truth se trouvent dans un autre compartiment Amazon S3, vos utilisateurs ont besoin d'autorisations supplémentaires. Pour de plus amples

informations, veuillez consulter the section called “Configuration des autorisations du compartiment Amazon S3”.

Vous pouvez associer la politique `AmazonLookoutVisionConsoleReadOnlyAccess` à vos identités IAM.

Groupes d'autorisations

Cette politique est regroupée en instructions en fonction de l'ensemble des autorisations fournies :

- `LookoutVisionReadOnlyAccess`— Permet d'accéder à des actions Lookout for Vision en lecture seule.
- `LookoutVisionConsoleS3BucketSearchAccess`— Permet de répertorier tous les compartiments S3 appartenant à l'appelant. Lookout for Vision utilise cette action pour identifier le bucket de console Lookout for Vision spécifique à la région AWS, s'il en existe un dans le compte de l'appelant.
- `LookoutVisionConsoleS3ObjectReadAccess`— Permet de lire les objets Amazon S3 et les versions des objets Amazon S3 dans les compartiments de console Lookout for Vision. Lookout for Vision utilise ces actions pour afficher les images dans des ensembles de données, des modèles et des détections d'essai.
- `LookoutVisionConsoleDashboardAccess`— Permet de lire CloudWatch les statistiques d'Amazon. Lookout for Vision utilise ces actions pour générer des statistiques relatives aux graphiques du tableau de bord et aux anomalies détectées.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:DescribeDataset",
                "lookoutvision:DescribeModel",
                "lookoutvision:DescribeProject",
                "lookoutvision:DescribeTrialDetection",
                "lookoutvision:DescribeModelPackagingJob",
                "lookoutvision:ListDatasetEntries",
```

```
                "lookoutvision:ListModels",
                "lookoutvision:ListProjects",
                "lookoutvision:ListTagsForResource",
                "lookoutvision:ListTrialDetections",
                "lookoutvision:ListModelPackagingJobs"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketSearchAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3ObjectReadAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*/*"
        },
        {
            "Sid": "LookoutVisionConsoleDashboardAccess",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        }
    ]
}
```

Mises AWS à jour des politiques gérées par Lookout for Vision

Consultez les informations relatives aux mises à jour apportées aux politiques AWS gérées pour Lookout for Vision depuis que ce service a commencé à suivre ces modifications. Pour recevoir des

alertes automatiques concernant les modifications apportées à cette page, abonnez-vous au flux RSS sur la page d'historique du document Lookout for Vision.

d'emballage ajoutées	ajouté les modèles d'opérations de packaging suivants AmazonLookoutVisionFullAccessaux politiques AmazonLookoutVisionConsoleFullAccessset : • DescribeModelPackagingJob • ListModelPackagingJobs • StartModelPackagingJob Amazon Lookout for Vision a ajouté les modèles d'opérations de packaging suivants AmazonLookoutVisionReadOnlyAccessaux politiques AmazonLookoutVisionConsoleReadOnlyAccessset : • DescribeModelPackagingJob • ListModelPackagingJobs	
Nouvelles politiques ajoutées	Amazon Lookout for Vision a ajouté les politiques suivantes. • AmazonLookoutVisionReadOnlyAccess • AmazonLookoutVisionFullAccess • AmazonLookoutVisionConsoleFullAccess • AmazonLookoutVisionConsoleReadOnlyAccess	11 mai 2021
Lookout for Vision a commencé à suivre les modifications	Amazon Lookout for Vision a commencé à suivre les	1er mars 2021

Modification	Description	Date
Modélisation des opérations d'emballage ajoutées	Amazon Lookout for Vision a ajouté les modèles d'opérations de packaging suivants AmazonLookoutVisionFullAccessaux politiques AmazonLookoutVisionConsoleFullAccesset : • DescribeModelPackagingJob • ListModelPackagingJobs • StartModelPackagingJob Amazon Lookout for Vision a ajouté les modèles d'opérations de packaging suivants AmazonLookoutVisionReadOnlyAccessaux politiques AmazonLookoutVisionConsoleReadOnlyAccesset : • DescribeModelPackagingJob • ListModelPackagingJobs	7 décembre 2021
Nouvelles politiques ajoutées	Amazon Lookout for Vision a ajouté les politiques suivantes. • AmazonLookoutVisionReadOnlyAccess • AmazonLookoutVisionFullAccess • AmazonLookoutVisionConsoleFullAccess • AmazonLookoutVisionConsoleReadOnlyAccess modifications apportées à AWS ses politiques gérées.	11 mai 2021

Résolution des problèmes d'identité et d'accès à Amazon Lookout for Vision

Utilisez les informations suivantes pour diagnostiquer et résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec Lookout for Vision et IAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans Lookout for Vision
- Je ne suis pas autorisé à effectuer iam : PassRole
- Je souhaite autoriser des personnes extérieures Compte AWS à moi à accéder à mes ressources Lookout for Vision

Je ne suis pas autorisé à effectuer une action dans Lookout for Vision

Si vous recevez une erreur qui indique que vous n'êtes pas autorisé à effectuer une action, vos politiques doivent être mises à jour afin de vous permettre d'effectuer l'action.

L'exemple d'erreur suivant se produit quand l'utilisateur IAM `mateojackson` tente d'utiliser la console pour afficher des informations détaillées sur une ressource *`my-example-widget`* fictive, mais ne dispose pas des autorisations `lookoutvision:`*`GetWidget`* fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lookoutvision:GetWidget on resource: my-example-widget
```

Dans ce cas, la politique qui s'applique à l'utilisateur `mateojackson` doit être mise à jour pour autoriser l'accès à la ressource *`my-example-widget`* à l'aide de l'action `lookoutvision:`*`GetWidget`*.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez un message d'erreur indiquant que vous n'êtes pas autorisé à effectuer cette `iam:PassRole` action, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à Lookout for Vision.

Certains vous Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, un utilisateur doit disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour effectuer une action dans Lookout for Vision. Toutefois, l'action nécessite que le service ait des autorisations accordées par un rôle de service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je souhaite autoriser des personnes extérieures Compte AWS à moi à accéder à mes ressources Lookout for Vision

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si Lookout for Vision prend en charge ces fonctionnalités, Comment Amazon Lookout for Vision fonctionne avec IAM consultez.
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues par des tiers dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.

- Pour en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Validation de conformité pour Amazon Lookout for Vision

Des auditeurs tiers évaluent la sécurité et la conformité d'Amazon Lookout for Vision dans le cadre de AWS plusieurs programmes de conformité. Amazon Lookout for Vision est conforme au règlement général sur la protection des données (RGPD).

Pour savoir si un programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- Conformité et gouvernance de la sécurité : ces guides de mise en œuvre de solutions traitent des considérations architecturales et fournissent les étapes à suivre afin de déployer des fonctionnalités de sécurité et de conformité.
- Référence des services éligibles HIPAA : liste les services éligibles HIPAA. Tous ne Services AWS sont pas éligibles à la loi HIPAA.
- AWS Ressources de https://aws.amazon.com/compliance/resources/ de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- AWS Guides de conformité destinés aux clients — Comprenez le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résument les meilleures pratiques en matière de sécurisation Services AWS et décrivent les directives relatives aux contrôles de sécurité dans plusieurs cadres (notamment le National Institute of Standards and Technology (NIST), le Payment Card Industry Security Standards Council (PCI) et l'Organisation internationale de normalisation (ISO)).

- Évaluation des ressources à l'aide des règles du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- AWS Security Hub— Cela Service AWS fournit une vue complète de votre état de sécurité interne AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos ressources AWS et vérifier votre conformité par rapport aux normes et aux bonnes pratiques du secteur de la sécurité. Pour obtenir la liste des services et des contrôles pris en charge, consultez Référence des contrôles Security Hub.
- Amazon GuardDuty — Cela Service AWS détecte les menaces potentielles qui pèsent sur vos charges de travail Comptes AWS, vos conteneurs et vos données en surveillant votre environnement pour détecter toute activité suspecte et malveillante. GuardDuty peut vous aider à répondre à diverses exigences de conformité, telles que la norme PCI DSS, en répondant aux exigences de détection des intrusions imposées par certains cadres de conformité.
- AWS Audit Manager— Cela vous Service AWS permet d'auditer en permanence votre AWS utilisation afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

La résilience dans Amazon Lookout for Vision

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section Infrastructure AWS mondiale.

Sécurité de l'infrastructure dans Amazon Lookout for Vision

En tant que service géré, Amazon Lookout for Vision est protégé AWS par la sécurité du réseau mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section Sécurité du AWS cloud. Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure,

consultez la section Protection de l'infrastructure dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à Lookout for Vision via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Ses suites de chiffrement PFS (Perfect Forward Secrecy) comme DHE (Ephemeral Diffie-Hellman) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l’aide d’un ID de clé d’accès et d’une clé d’accès secrète associée à un principal IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d’identification de sécurité temporaires et signer les demandes.

Surveillance d'Amazon Lookout for Vision

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité et des performances d'Amazon Lookout for Vision et de vos autres solutions AWS. AWS fournit les outils de surveillance suivants pour surveiller Lookout for Vision, signaler un problème et prendre des mesures automatiques le cas échéant :

- Amazon CloudWatch surveille vos AWS ressources et les applications que vous utilisez AWS en temps réel. Vous pouvez collecter et suivre les métriques, créer des tableaux de bord personnalisés, et définir des alarmes qui vous informent ou prennent des mesures lorsqu'une métrique spécifique atteint un seuil que vous spécifiez. Par exemple, vous pouvez CloudWatch suivre l'utilisation du processeur ou d'autres indicateurs de vos EC2 instances Amazon et lancer automatiquement de nouvelles instances en cas de besoin. Pour plus d'informations, consultez le guide de CloudWatch l'utilisateur Amazon.
- Amazon CloudWatch Logs vous permet de surveiller, de stocker et d'accéder à vos fichiers journaux à partir d' EC2 instances Amazon et d'autres sources. CloudTrail CloudWatch Les journaux peuvent surveiller les informations contenues dans les fichiers journaux et vous avertir lorsque certains seuils sont atteints. Vous pouvez également archiver vos données de journaux dans une solution de stockage hautement durable. Pour plus d'informations, consultez le guide de l'utilisateur d'Amazon CloudWatch Logs.
- Amazon EventBridge peut être utilisé pour automatiser vos AWS services et répondre automatiquement aux événements du système, tels que les problèmes de disponibilité des applications ou les modifications des ressources. Les événements AWS liés aux services sont diffusés EventBridge en temps quasi réel. Vous pouvez écrire des règles simples pour préciser les événements qui vous intéressent et les actions automatisées à effectuer quand un événement correspond à une règle. Pour plus d'informations, consultez le guide de EventBridge l'utilisateur Amazon.
- AWS CloudTrailcapture les appels d'API et les événements associés effectués par ou pour le compte de votre AWS compte et envoie les fichiers journaux dans un compartiment Amazon S3 que vous spécifiez. Vous pouvez identifier les utilisateurs et les comptes appelés AWS, l'adresse IP source à partir de laquelle les appels ont été effectués et la date des appels. Pour plus d'informations, consultez le AWS CloudTrail Guide de l'utilisateur .

Surveillance de Lookout for Vision avec Amazon CloudWatch

Vous pouvez surveiller Lookout for Vision à CloudWatch l'aide de Lookout for Vision, qui collecte des données brutes et les transforme en indicateurs lisibles en temps quasi réel. Ces statistiques sont enregistrées pour une durée de 15 mois ; par conséquent, vous pouvez accéder aux informations historiques et acquérir un meilleur point de vue de la façon dont votre service ou application web s'exécute. Vous pouvez également définir des alarmes qui surveillent certains seuils et envoient des notifications ou prennent des mesures lorsque ces seuils sont atteints. Pour plus d'informations, consultez le guide de CloudWatch l'utilisateur Amazon.

Le service Lookout for Vision indique les métriques suivantes dans l'espace `AWS/LookoutVision` de noms.

Métrique	Description
`DetectedAnomalyCount`	Le nombre d'anomalies détectées dans un projet Statistiques valides : `Sum,Average` Unité : nombre
`ProcessedImageCount`	Le nombre total d'images traitées par le biais de la détection d'anomalies Statistiques valides : `Sum,Average` Unité : nombre
`InvalidImageCount`	Le nombre d'images non valides qui n'ont pas pu renvoyer de résultat Statistiques valides : `Sum,Average` Unité : nombre
`SuccessfulRequestCount`	Le nombre d'appels d'API réussis Statistiques valides : `Sum,Average` Unité : nombre

Métrique	Description
`ErrorCount`	Le nombre d'erreurs d'API Statistiques valides : `Sum,Average` Unité : nombre
`ThrottledCount`	Le nombre d'erreurs d'API dues à la régulation Statistiques valides : `Sum,Average` Unité : nombre
`Time`	Durée en millisecondes nécessaire à Lookout for Vision pour calculer la détection des anomalies Statistiques valides : `Data Samples,Average` Unités : millisecondes pour les `Average` statistiques et nombre pour les statistiques `Data Samples`
`MinInferenceUnits`	Le nombre minimum d'unités d'inférence spécifié lors de la demande `StartModel` . Statistiques valides : `Average` Unité : nombre
`MaxInferenceUnits`	Le nombre maximum d'unités d'inférence spécifiées lors de la demande `StartModel` . Statistiques valides : `Average` Unité : nombre
`DesiredInferenceUnits`	Le nombre d'unités d'inférence par rapport auxquelle s Lookout for Vision augmente ou diminue. Statistiques valides : `Average` Unité : nombre

Métrique	Description
`InServiceInferenceUnits`	Le nombre d'unités d'inférence utilisées par le modèle. Statistiques valides : `Average` Il est recommandé d'utiliser la statistique Average pour obtenir la moyenne sur 1 minute du nombre d'instances utilisées. Unité : nombre

Les dimensions suivantes sont prises en charge pour les métriques Lookout for Vision.

Dimension	Description
`ProjectName`	Vous pouvez diviser les indicateurs par projet pour voir quels projets rencontrent des problèmes ou doivent être mis à jour.
`ModelVersion`	Vous pouvez diviser les métriques par version de modèle pour voir quels modèles rencontrent des problèmes ou doivent être mis à jour.

Journalisation des appels d'API Lookout for Vision avec AWS CloudTrail

Amazon Lookout for Vision est intégré AWS CloudTrailà un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou AWS un service dans Lookout for Vision. CloudTrail capture tous les appels d'API pour Lookout for Vision sous forme d'événements. Les appels capturés incluent des appels provenant de la console Lookout for Vision et des appels de code vers les opérations de l'API Lookout for Vision. Si vous créez un suivi, vous pouvez activer la diffusion continue d' CloudTrail événements vers un compartiment Amazon S3, y compris des événements pour Lookout for Vision. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents dans la CloudTrail console dans Historique des événements. À l'aide

des informations collectées par CloudTrail, vous pouvez déterminer la demande envoyée à Lookout for Vision, l'adresse IP à partir de laquelle la demande a été faite, l'auteur de la demande, la date à laquelle elle a été faite, ainsi que des informations supplémentaires.

Pour en savoir plus CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur.

Informations sur Lookout for Vision dans CloudTrail

CloudTrail est activé sur votre AWS compte lorsque vous le créez. Lorsqu'une activité a lieu dans Lookout for Vision, elle est enregistrée dans CloudTrail un événement avec d' AWS autres événements de service dans l'historique des événements. Vous pouvez consulter, rechercher et télécharger les événements récents dans votre AWS compte. Pour plus d'informations, consultez la section Affichage des événements avec l'historique des CloudTrail événements.

Pour un enregistrement permanent des événements enregistrés sur votre AWS compte, y compris ceux de Lookout for Vision, créez un parcours. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un journal d'activité dans la console, il s’applique à toutes les régions AWS. Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez configurer d'autres AWS services pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence. Pour plus d’informations, consultez les ressources suivantes :

- Vue d’ensemble de la création d’un journal d’activité
- CloudTrail Services et intégrations pris en charge
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers CloudTrail journaux de plusieurs régions et réception de fichiers CloudTrail journaux de plusieurs comptes

Toutes les actions de Lookout for Vision sont enregistrées CloudTrail et documentées dans la documentation de référence de l'API Lookout for Vision. Par exemple, les appels au `CreateProject` `DetectAnomalies` et les `StartModel` actions génèrent des entrées dans les fichiers CloudTrail journaux.

Si vous surveillez les appels d'API Amazon Lookout for Vision, il est possible que des appels aux personnes suivantes s'affichent. APIs

- vision de l'œil : StartTriallDetection

- vision de l'œil : ListTriallDetection
- vision de l'œil : DescribeTrialDetection

Ces appels spéciaux sont utilisés par Amazon Lookout for Vision pour prendre en charge diverses opérations liées à la détection des essais. Pour de plus amples informations, veuillez consulter Vérification de votre modèle à l'aide d'une tâche de détection d'essai.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été faite avec les informations AWS Identity and Access Management d'identification root ou utilisateur.
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la demande a été faite par un autre AWS service.

Pour plus d'informations, consultez la section Élément userIdentity CloudTrail .

Comprendre les entrées du fichier journal Lookout for Vision

Un suivi est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre une entrée de CloudTrail journal illustrant l'`CreateDataset`action.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAYN4CJAYDEXAMPLE:user",
    "arn": "arn:aws:sts::123456789012:assumed-role/Admin/MyUser",
    "accountId": "123456789012",
    "accessKeyId": "ASIAYN4CJAYEXAMPLE",
```

```
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAYN4CJAYDCGEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/Admin",
        "accountId": "123456789012",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-11-20T13:15:09Z"
      }
    }
  },
  "eventTime": "2020-11-20T13:15:43Z",
  "eventSource": "lookoutvision.amazonaws.com",
  "eventName": "CreateDataset",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "128.0.0.1",
  "userAgent": "aws-cli/3",
  "requestParameters": {
    "projectName": "P1",
    "datasetType": "train",
    "datasetSource": {
      "groundTruthManifest": {
        "s3Object": {
          "bucket": "myuser-bucketname",
          "key": "training.manifest"
        }
      }
    },
    "clientToken": "EXAMPLE-0526-47dd-a5d3-2ca975820a34"
  },
  "responseElements": {
    "status": "CREATE_IN_PROGRESS"
  },
  "requestID": "EXAMPLE-15e1-4bc9-be38-cda2537c75bf",
  "eventID": "EXAMPLE-c5e7-43e0-8449-8d9b87e15acb",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "123456789012"
```

```
}
```

Création de ressources Amazon Lookout for Vision avec AWS CloudFormation

Amazon Lookout for Vision est intégré AWS CloudFormationà un service qui vous aide à modéliser et à configurer AWS 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 Lookout for Vision.

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

Lookout for Vision et modèles AWS CloudFormation

Pour mettre en service et configurer des projets pour Lookout for Vision et les services associés, vous devez AWS CloudFormation comprendre les modèles. 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 les projets Lookout for Vision, notamment des exemples de modèles JSON et YAML, LookoutVision consultez la référence des types de ressources.

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

Accédez à Amazon Lookout for Vision à l'aide d'un point de terminaison AWS PrivateLink d'interface ()

Vous pouvez l'utiliser AWS PrivateLink pour créer une connexion privée entre votre VPC et Amazon Lookout for Vision. Vous pouvez accéder à Lookout for Vision comme s'il se trouvait dans votre VPC, sans passer par une passerelle Internet, un appareil NAT, une connexion VPN ou une connexion. AWS Direct Connect Les instances de votre VPC n'ont pas besoin d'adresses IP publiques pour accéder à Lookout for Vision.

Vous établissez cette connexion privée en créant un point de terminaison d'interface optimisé par AWS PrivateLink. Nous créons une interface réseau de point de terminaison dans chaque sous-réseau que vous activez pour le point de terminaison d'interface. Il s'agit d'interfaces réseau gérées par les demandeurs qui servent de point d'entrée pour le trafic destiné à Lookout for Vision.

Pour plus d'informations, consultez la section Accès Services AWS par AWS PrivateLink le biais du AWS PrivateLink guide.

Considérations relatives aux points de terminaison VPC Lookout for Vision

Avant de configurer un point de terminaison d'interface pour Lookout for Vision, consultez les considérations AWS PrivateLink du guide.

Lookout for Vision permet d'appeler toutes ses actions d'API via le point de terminaison de l'interface.

Les politiques relatives aux points de terminaison VPC ne sont pas prises en charge pour Lookout for Vision. Par défaut, l'accès complet à Lookout for Vision est autorisé via le point de terminaison de l'interface. Vous pouvez également associer un groupe de sécurité aux interfaces réseau du point de terminaison afin de contrôler le trafic vers Lookout for Vision via le point de terminaison de l'interface.

Création d'un point de terminaison VPC d'interface pour Lookout for Vision

Vous pouvez créer un point de terminaison d'interface pour Lookout for Vision à l'aide de la console Amazon VPC ou AWS Command Line Interface du ().AWS CLI Pour plus d'informations, consultez Création d'un point de terminaison d'interface dans le Guide AWS PrivateLink .

Créez un point de terminaison d'interface pour Lookout for Vision en utilisant le nom de service suivant :

```
com.amazonaws.region.lookoutvision
```

Si vous activez le DNS privé pour le point de terminaison de l'interface, vous pouvez envoyer des demandes d'API à Lookout for Vision en utilisant son nom DNS régional par défaut. Par exemple, `lookoutvision.us-east-1.amazonaws.com`.

Création d'une politique de point de terminaison VPC pour Lookout for Vision

Une politique de point de terminaison est une ressource IAM que vous pouvez attacher à votre point de terminaison d'interface. La politique de point de terminaison par défaut autorise un accès complet à Lookout for Vision via le point de terminaison de l'interface. Pour contrôler l'accès autorisé à Lookout for Vision depuis votre VPC, associez une politique de point de terminaison personnalisée au point de terminaison de l'interface.

Une politique de point de terminaison spécifie les informations suivantes :

- Les principaux qui peuvent effectuer des actions (Comptes AWS, utilisateurs IAM et rôles IAM).
- Les actions qui peuvent être effectuées.
- La ressource sur laquelle les actions peuvent être effectuées.

Pour plus d'informations, consultez Contrôle de l'accès aux services à l'aide de politiques de point de terminaison dans le Guide AWS PrivateLink .

Exemple : politique de point de terminaison VPC pour les actions Lookout for Vision

Voici un exemple de politique de point de terminaison personnalisée pour Lookout for Vision. Lorsque vous attachez cette politique à votre point de terminaison d'interface, elle indique que tous les utilisateurs ayant accès au point de terminaison d'interface VPC sont autorisés à appeler l'opération d'`DetectAnomalies`API pour le modèle `myModel` Lookout for Vision associé au projet. `myProject`

```
{
   "Statement":[
      {
```

```
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "lookoutvision:DetectAnomalies"
         ],
         "Resource": "arn:aws:lookoutvision:us-west-2:123456789012:model/myProject/
myModel"
      }
   ]
}
```

Quotas dans Amazon Lookout for Vision

Les tableaux suivants décrivent les quotas actuels au sein d'Amazon Lookout for Vision. Pour plus d'informations sur les quotas qui peuvent être modifiés, consultez la section Quotas de service AWS.

Quotas de modèle

Les quotas suivants s'appliquent aux tests, à la formation et aux fonctionnalités d'un modèle.

Ressource	Quota
Format de fichier pris en charge	Formats d'image PNG et JPEG
Dimension d'image minimale d'un fichier image dans un compartiment Amazon S3	64 pixels x 64 pixels
Dimension d'image maximale d'un fichier image dans un compartiment Amazon S3	Le maximum est de 4096 pixels X 4096 pixels. Les petites dimensions peuvent être télécharg ées plus rapidement.
Dimensions d'image différentes des fichiers image utilisés dans un projet	Toutes les images du jeu de données doivent avoir les mêmes dimensions
Taille de fichier maximale pour une image dans un compartiment Amazon S3	8 Mo
Absence d'étiquettes	Les images doivent être étiquetées comme normales ou anormales avant l'entraînement. Les images sans étiquette sont ignorées pendant l'entraînement.
Nombre minimum d'images étiquetées « normales » dans le jeu de données d'entraîn ement	10 pour un projet avec des ensembles de données d'entraînement et de test séparés. 20 pour un projet avec un seul ensemble de données.
Nombre minimal d'images étiquetées anomalie dans un jeu de données d'entraînement	0 pour un projet avec des ensembles de données d'entraînement et de test distincts.

Ressource	Quota
	10 pour un projet avec un seul ensemble de données.
Nombre maximum d'images dans le jeu de données d'entraînement à la classification	16,000
Nombre maximal d'images dans un jeu de données de test de classification	4 000
Nombre minimum d'images étiquetées normales dans l'ensemble de données de test	10
Nombre minimum d'images étiquetées anomalie dans l'ensemble de données de test	10
Nombre maximal d'images dans un jeu de données de formation à la localisation des anomalies	8000
Nombre maximal d'images dans un jeu de données de test de localisation d'anomalies	800
Nombre maximal d'images dans l'ensemble de données de détection des essais	2 000
Taille maximale du fichier manifeste du jeu de données	1 Go
Nombre maximal de jeux de données d'entraînement dans un modèle	1
Durée d'entraînement maximale	24 heures
Durée maximale du test	24 heures
Nombre maximum d'étiquettes d'anomalies dans un projet	100

Ressource	Quota
Nombre maximal d'étiquettes d'anomalie sur une image de masque	20
Nombre minimum d'images pour une étiquette d'anomalie. Pour compter, l'image ne doit contenir qu'un seul type d'étiquette d'anomalie.	20 pour un projet de jeu de données unique. 10 pour chaque ensemble de données dans un projet avec des ensembles de données d'entraînement et de test distincts.

Historique du document pour Amazon Lookout for Vision

Le tableau suivant décrit les modifications importantes apportées à chaque version du guide du développeur Amazon Lookout for Vision. Pour recevoir les notifications de mise à jour de cette documentation, abonnez-vous à un flux RSS.

- Dernière mise à jour de la documentation : 20 février 2023

Modification	Description	Date
Avis de fin de support	Avis de fin de support : le 31 octobre 2025, le support d'Amazon Lookout for Vision AWS sera interrompu. Après le 31 octobre 2025, vous ne pourrez plus accéder à la console Lookout for Vision ni aux ressources Lookout for Vision. Pour plus d'informations, consultez ce billet de blog.	10 octobre 2024
Ajout d'un exemple de fonction Lambda	Exemple montrant comment détecter des anomalies avec une AWS Lambda fonction. Pour plus d'informations, consultez la section Recherche d'anomalies avec une fonction AWS Lambda.	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.	8 février 2023

Exemple d'exportation de jeu de données ajouté	Ajout d'un exemple Python montrant comment utiliser l'`ListDatasetEntries` opération pour exporter les ensembles de données depuis un projet Amazon Lookout for Vision. Pour plus d'informations, consultez Exporter des ensembles de données à partir d'un projet (SDK).	2 décembre 2022
Rubrique de démarrage mise à jour	Mise à jour du mode de démarrage pour montrer la création d'un modèle de segmentation d'image avec un exemple de jeu de données. Pour plus d'informations, consultez Getting started with Amazon Lookout for Vision.	20 octobre 2022
Rubrique de résolution des problèmes ajoutée	Ajout d'une rubrique de formation sur le dépannage des modèles. Pour plus d'informations, consultez la section Résolution des problèmes liés aux modèles de formation.	17 octobre 2022

Ajout d'un sujet sur l'utilisation des jobs Amazon SageMaker AI Ground Truth	Au lieu d'étiqueter vous-même les images, vous pouvez utiliser les tâches Amazon SageMaker AI Ground Truth pour étiqueter les images à des fins de classification et de segmentation d'images. Pour plus d'informations, consultez Utiliser une tâche Amazon SageMaker AI Ground Truth.	19 août 2022
Amazon Lookout for Vision propose désormais la localisation des anomalies.	Vous pouvez créer un modèle de segmentation qui identifie les emplacements sur une image présentant différents types d'anomalies (telles qu'une égratignure, une bosse ou une déchirure), ainsi que l'étiquette et la taille de l'anomalie. Pour plus d'informations, consultez Exécuter votre modèle Amazon Lookout for Vision entraîné.	16 août 2022
Amazon Lookout for Vision fournit désormais une inférence du processeur sur les appareils périphériques.	Les modèles Amazon Lookout for Vision peuvent désormais être déployés pour exécuter des inférences localement sur une plate-forme de calcul x86 exécutant Linux avec un simple processeur, sans avoir besoin d'un accélérateur GPU. Pour plus d'informations, consultez la section Accélérateur de processeur.	16 août 2022

Amazon Lookout for Vision peut désormais redimensionner automatiquement les unités d'inférence.	Pour faire face aux pics de demande, Amazon Lookout for Vision peut désormais ajuster le nombre d'unités d'inférence utilisées par votre modèle. Pour plus d'informations, consultez la section Exécution de votre modèle Amazon Lookout for Vision entraîné.	16 août 2022
Exemples Java ajoutés	Le guide du développeur Amazon Lookout for Vision inclut désormais des exemples Java. Pour plus d'informations, consultez la section Mise en route avec le AWS SDK.	2 mai 2022
Disponibilité générale du déploiement du modèle sur un appareil périphérique	Le déploiement du modèle sur un appareil périphérique géré par AWS IoT Greengrass Version 2 est désormais généralement disponible. Pour plus d'informations, consultez la section Utilisation de votre modèle Amazon Lookout for Vision sur un appareil périphérique.	14 mars 2022
Informations mises à jour sur le bucket de console	Informations mises à jour sur le contenu du compartiment de console et autres approches pour créer le compartiment de console. Pour plus d'informations, consultez Étape 4 : Création du bucket de console.	7 mars 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 à partir d'un fichier CSV. Pour plus d’informations, consultez Création d’un fichier manifeste à partir d’un fichier CSV.	10 février 2022
Version préliminaire du déploiement du modèle sur un appareil périphérique	La version préliminaire du déploiement du modèle sur un appareil périphérique géré par AWS IoT Greengrass Version 2 est désormais disponible. Pour plus d'informations, consultez la section Utilisation de votre modèle Amazon Lookout for Vision sur un appareil périphérique.	7 décembre 2021
Nouveaux exemples de Python et Java 2 ajoutés	Ajout d'exemples Python et Java 2 pour analyser des images avec`DetectAnomalies` . Pour plus d'informations, consultez la section Détection d'anomalies dans une image.	7 septembre 2021
De nouvelles politiques AWS gérées ont été ajoutées.	Amazon Lookout for Vision ajoute la prise en charge AWS des politiques gérées. Pour plus d'informations, consultez les politiques AWS gérées pour Amazon Lookout for Vision.	11 mai 2021

Informations sur l'unité d'inférence mises à jour.	Ajout d'informations décrivant les unités d'inférence et leur mode de facturation. Pour plus d'informations, consultez la section Exécution de votre modèle Amazon Lookout for Vision entraîné.	15 mars 2021
Disponibilité générale d'Amazon Lookout for Vision.	Amazon Lookout for Vision est désormais disponible pour tous. Exemples de code Python mis à jour pour gérer des tâches asynchrones telles que l'entraînement d'un modèle.	17 février 2021
Marquage et AWS CloudFormation support ajoutés.	Vous pouvez désormais baliser les modèles Amazon Lookout for Vision et créer des projets AWS CloudFormation avec. Pour plus d'informations, consultez les sections Marquage de modèles et Création de projets Amazon Lookout for Vision avec AWS CloudFormation.	31 janvier 2021
Nouvelle fonctionnalité et nouveau guide	Il s'agit de la version initiale du service Amazon Lookout for Vision, Amazon Lookout for Vision Developer Guide.	1er décembre 2020

AWS Glossaire

Pour la AWS terminologie la plus récente, consultez le AWS glossaire dans la Glossaire AWS référence.